BAB 4 IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan dilakukan implementasi dan pengujian terhadap aplikasi yang dibangun. Tahapan ini dilakukan setelah analisis dan perancangan selesai dilakukan dan selanjutnya akan diimplementasikan kedalam bahasa pemograman. Setelah implementasi maka dilakukan pengujian terhadap aplikasi. Aplikasi yang telah dibangun akan diimplementasikan untuk mengetahui apakah sistem tersebut dapat berjalan sesuai dengan tujuannya atau tidak.

4.1 Implementasi

Tujuan implementasi adalah untuk menerapkan perancangan yang telah dilakukan terhadap sistem sehingga *user* dapat memberi masukan demi berkembangnya sistem yang telah dibangun sebagai simulasi dari aplikasi penerapan algoritma BFS dan DFS untuk pencarian jalan pada *game* onet *fruit*. Adapun Perangkat keras dan perangkat lunak yang digunakan untuk melakukan implementasi dan pengujian aplikasi yaitu sebagai berikut:

a. Perangkat keras

Perangkat keras yang digunakan untuk mengimplementasikan sistem dapat menggunakan 2 cara yakni *mobile* android atau komputer dengan spesifikasi dijelaskan pada tabel 4.1 berikut:

Tabel 4.1 Perangkat keras untuk implementasi sistem

Perangkat Keras	Spesifikasi
Processor	AMD 1.6 GHz
Monitor	LCD 11.1 Inc
Memori	DDR3 4GB
Harddisk	320GB
Keyboard dan Mouse	(*OnBoard*)

b. Perangkat lunak

Perangkat lunak yang digunakan untuk mengimplementasikan sistem dapat menggunakan 2 cara yakni:

1. *Mobile* android

Copy file onet.apk ke *mobile* android, lalu buka file manager dan cari lokasi tempat file apk disimpan lalu klik file tersebut dan klik install. Setelah selesai instalasi, maka secara otomatis *game* onet *fruit* dapat dimainkan.

2. Menggunakan komputer

Untuk dapat memainkan *game* onet *fruit* di komputer maka langkah-langkah yang harus dilakukan adalah:

- *Download* dan *install* java JDK
- *Download* dan *install* android SDK dan AVD manager
- Jalankan emulator android.

- *Copy* file apk ke folder 'SDK_ROOT/platform-tools/'
- Ketikan perintah : adb install onet.apk
- Lihat di emulator aplikasi sudah terinstall dan siap dimainkan.

4.1.1 Implementasi Antarmuka

Implementasi antarmuka untuk perancangan aplikasi tampilan utama dapat dilihat sebagai berikut:

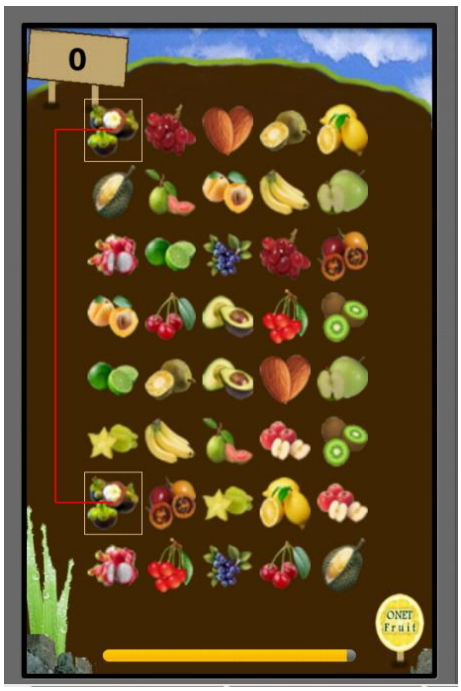


Gambar 4.1 Tampilan Awal Onet *Fruit*

Layar ini merupakan tampilan awal pada *game* onet *fruit*, dimana terdapat beberapa menu yaitu *Play*, *Score*, *Credits* dan *Instruction* dan ada 2 *button* merah yang terletak di pojok kanan atas.

- Menu *Play* merupakan awal permainan *game* onet *fruit.*
- Menu *Score* merupakan informasi nilai 10 teratas yang di dapat oleh pemain.
- Menu *Credits* merupakan informasi mengenai pembuat aplikasi onet *fruit*.
- Menu *Instruction* merupakan panduan untuk pengguna bagaimana cara memainkan *game* onet fruit ini.
- Button , untuk mengaktifkan atau menonaktifkan suara.
- Button , untuk keluar dari permainan.

Adapun gambar untuk setiap menu dapat di lihat sebagai berikut:



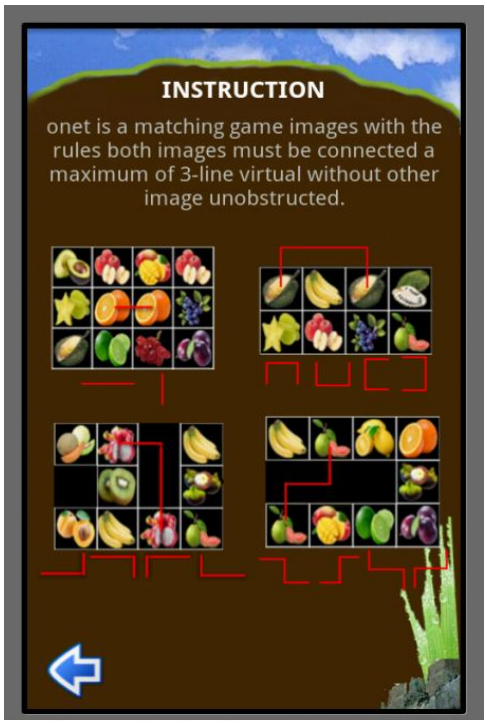
Gambar 4.2 Tampilan Awal Permainan Onet *Fruit*



Gambar 4.3 Tampilan *Score* Permainan Onet *Fruit*



Gambar 4.4 Tampilan *Credits* Permainan Onet *Fruit*



Gambar 4.5 Tampilan *Instruction* Permainan Onet *Fruit*

4.2 Pengujian

Pengujian pada sistem yang baru dibangun dilakukan melalui tahap pengujian *white-box*, pengujian *alpha* (*black-box*) dan pengujian *betha*.

4.2.1 Pengujian *White-Box*

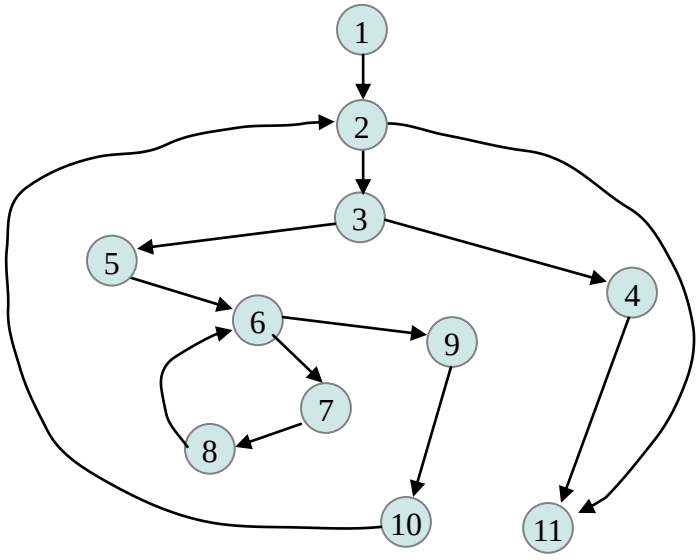
Pengujian *white-box* atau *white-box testing* merupakan metode perancangan *test case* yang menggunakan struktur kontrol dari perancangan prosedural dalam mendapatkan *test case*. Adapun metode yang digunakan dalam pengujian *white-box* ini adalah metode *Basis Path*. Metode *Basis Path* mengijinkan pendesain kasus uji untuk membuat perkiraan lojik yang kompleks dari desain prosedural dan menggunakan perkiraan ini untuk mendefinisikan aliran eksekusi .

a. Pengujian *White-box* pencarian jalan dengan algoritma BFS

Berikut ini *pseudocode* algoritma BFS:

```
1:   baca node awal dan node akhir
     simpan node awal pada record
2:   Do while record masih ada
        baca record
3:      if (record = node akhir)
4:              return record;
                break;
5:      else
                simpan record child (left, top, right, bottom) pada childNode
6:              Do while childNode masih ada
7:                      proses childNode
                        simpan childNode pada record
8:              endDo
9:      endif
10: endDo
    return null
11: end
```

Berdasarkan pseudocode di atas maka diagram alirnya sebagai berikut:



Gambar 4.6 Diagram alir algoritma BFS

Dari diagram alir di atas, dapat dihitung *cyclomatic complexity*, yakni:

1. Flowgraph mempunyai 4 region
2. V(G) = 13 edge – 11 node + 2 = 4
3. V(G) = 3 predicate node + 1 = 4

Dari hasil perhitungan *cyclomatic complexity* terdapat 4 independent path yaitu:

Path 1 = 1 – 2 – 11

Path 2 = 1 – 2 – 3 – 4 – 11

Path 3 = 1 – 2 – 3 – 5 – 6 – 9 – 10 – 2 – 11

Path 4 = 1 – 2 – 3 – 5 – 6 – 7 – 8 – 6 – 9 – 10 – 2 – 11

Untuk melakukan uji coba basis path di atas digunakan graph matrik. Graph matrik merupakan matrik empat persegi yang mempunyai ukuran yang sama dengan jumlah node pada flowgraph. Adapun graph matrik pada flowgraph BFS di atas adalah sebagai berikut:

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**n(E)-1**
1		**1**										**1-1 = 0**
2			**1**								**1**	**2-1 = 1**
3				**1**	**1**							**2-1 = 1**
4											**1**	**1-1 = 0**
5						**1**						**1-1 = 0**
6							**1**		**1**			**2-1 = 1**
7								**1**				**1-1 = 0**
8						**1**						**1-1 = 0**
9										**1**		**1-1 = 0**
10		**1**										**1-1 = 0**
11												**0**
											Sum(E)+1	**3+1=4**

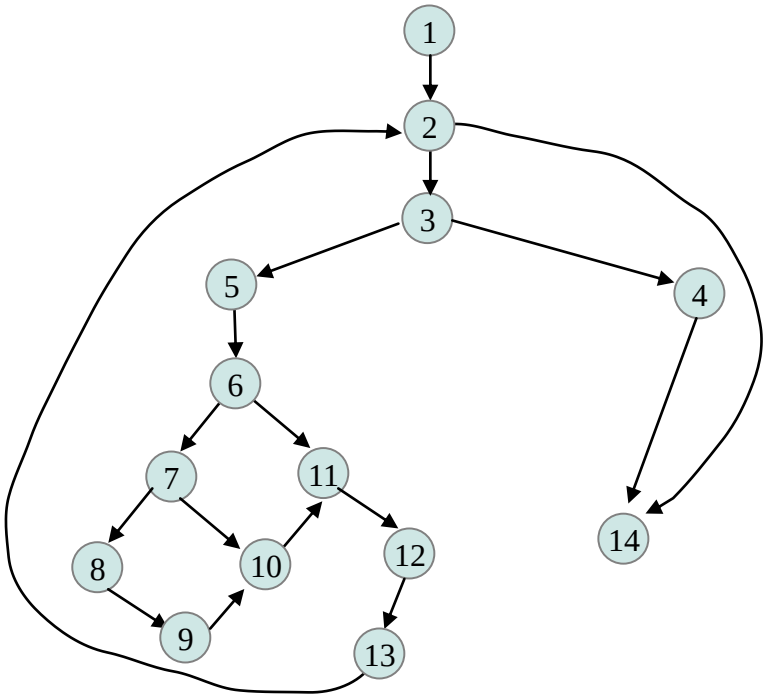
Gambar 4.7 Graph Matrik BFS

b. Pengujian *White-box* pencarian jalan dengan algoritma DFS

Berikut ini *pseudocode* algoritma DFS:

```
1:   baca node awal dan node akhir
     node = node awal
2:   Do While (node != null)
        baca node
3:      if (node = node akhir)
4:              return node
                break
5:      else
                hasil = proses node child (left)
6:              if (hasil = null)
                        hasil = proses node child (top)
7:                      if (hasil = null)
                                hasil = proses node child (right)
8:                              if (hasil = null)
                                        hasil = proses node child (bottom)
9:                              endif
10:                     endif
11:             endif
                node = hasil
12:     endif
13: endDo
    return null
14: end
```

Berdasarkan pseudocode di atas maka diagram alirnya sebagai berikut:



Gambar 4.8 Diagram alir algoritma DFS

Dari diagram alir di atas, dapat dihitung *cyclomatic complexity*, yakni:

4. Flowgraph mempunyai 5 region
5. V(G) = 17 edge – 14 node + 2 = 5
6. V(G) = 4 predicate node + 1 = 5

Dari hasil perhitungan *cyclomatic complexity* terdapat 5 independent path yaitu:

Path 1 = 1 – 2 – 14

Path 2 = 1 – 2 – 3 – 4 – 14

Path 3 = 1 – 2 – 3 – 5 – 6 – 11 – 12 – 13 – 2 – 14

Path 4 = 1 – 2 – 3 – 5 – 6 – 7 – 10 – 11 – 12 – 13 – 2 – 14

Path 5 = 1 – 2 – 3 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 2 – 14

Untuk melakukan uji coba basis path di atas digunakan graph matrik. Graph matrik merupakan matrik empat persegi yang mempunyai ukuran yang sama dengan jumlah node pada flowgraph. Adapun graph matrik pada flowgraph DFS di atas adalah sebagai berikut:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	n(E)-1
1		**1**													**1-1 = 0**
2			**1**											**1**	**2-1 = 1**
3				**1**	**1**										**2-1 = 1**
4														**1**	**1-1 = 0**
5						**1**									**1-1 = 0**
6							**1**				**1**				**2-1 = 1**
7								**1**		**1**					**2-1 = 1**
8									**1**						**1-1 = 0**
9										**1**					**1-1 = 0**
10											**1**				**1-1 = 0**
11												**1**			**1-1 = 0**
12													**1**		**1-1 = 0**
13		**1**													**1-1 = 0**
14															**0**
														Sum(E)+1	**4+1 = 5**

Gambar 4.9 Graph Matrik DFS

4.2.2 Pengujian *Alpha* (*Black box*)

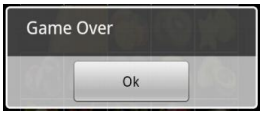
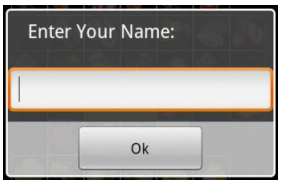
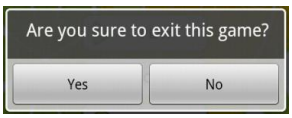
Metode yang digunakan dalam pengujian *alpha* adalah metode *black box* yang fokus pada persyaratan fungsional dari perangkat lunak yang dibangun.

a. Kasus dan hasil pengujian permainan

Berdasarkan rencana pengujian, maka dapat dilakukan pengujian alpha dengan blackbox pada aplikasi game onet fruit yang dijelaskan pada tabel 4.2.

Tabel 4.2 Pengujian Permainan

Kasus dan Hasil Uji (Data Normal)			
Aktivitas Pengujian	**Realisasi yang diharapkan**	**Hasil Pengujian**	**Kesimpulan**
Klik tombol Play 	Untuk memulai permainan	Permainan dapat dimulai	[v] Diterima [ ] Ditolak
Klik tombol *Score* 	Untuk menampilkan *score* pemain 10 teratas	Tombol dapat berfungsi sesuai yang diharapkan	[v] Diterima [ ] Ditolak
Klik tombol Credits 	Untuk menampilkan informasi pembuat	Tombol dapat berfungsi sesuai yang diharapkan	[v] Diterima [ ] Ditolak
Klik tombol *Instruction* 	Untuk menampilkan cara bermain onet *fruit*	Tombol dapat berfungsi sesuai yang diharapkan	[v] Diterima [ ] Ditolak
Klik tombol keluar 	Untuk keluar dari aplikasi	Tombol dapat berfungsi sesuai yang diharapkan	[v] Diterima [ ] Ditolak
Klik tombol *sound* 	Untuk mengeluarkan dan mematikan suara	Tombol dapat berfungsi sesuai yang diharapkan	[v] Diterima [ ] Ditolak
Klik tombol titik awal	Menampilkan titik awal pada papan onet	Tombol dapat berfungsi sesuai yang diharapkan	[v] Diterima [ ] Ditolak
Klik tombol titik tujuan	Menampilkan titik tujuan pada papan onet	Tombol dapat berfungsi sesuai yang diharapkan	[v] Diterima [ ] Ditolak

Klik tombol back 	Untuk kembali ke halaman utama	Tombol dapat berfungsi sesuai yang diharapkan	[v] Diterima [ ] Ditolak
Klik tombol Ok untuk masuk level berikutnya 	Masuk ke level selanjutnya	Tombol dapat berfungsi sesuai yang diharapkan	[v] Diterima [ ] Ditolak
Klik tombol Ok pada saat game over 	Permainan berakhir dan kembali ke tampilan awal	Tombol dapat berfungsi sesuai yang diharapkan	[v] Diterima [ ] Ditolak
Klik tombol Ok saat permainan selesai 	Permainan berakhir dan menampilkan pesan inputkan nama	Tombol dapat berfungsi sesuai yang diharapkan	[v] Diterima [ ] Ditolak
Klik tombol Yes/No saat konfirmasi untuk keluar dari permainan 	Permainan berakhir dan menampilkan pesan inputkan nama	Tombol dapat berfungsi sesuai yang diharapkan	[v] Diterima [ ] Ditolak
Kasus dan Hasil Uji (Data Tidak Normal)			
Aktivitas Pengujian	**Realisasi yang diharapkan**	**Hasil Pengujian**	**Kesimpulan**
Data input nama kosong	Muncul pesan nama tidak boleh kosong	Hasil sesuai dengan yang diharapkan	[v] Diterima [ ] Ditolak

b. Pengujian parameter aplikasi

Performansi yang akan diuji pada aplikasi yang telah dibangun yaitu jumlah garis / *lines*, *cells*, dan node yang diperiksa dari titik awal sampai titik tujuan. Adapun prioritas yang akan diambil dari perfomansi yang diujikan tersebut pada permainan *onet fruit* untuk pencarian jalan dari algoritma BFS dan DFS adalah pertama dilihat dari jumlah *lines* yang paling sedikit, karena semakin kecil jumlah lines yang diperoleh maka semakin dekat jarak yang ditempuh / memperoleh jarak terpendek, setelah itu dilihat dari jumlah cells. Biasanya semakin kecil jumlah lines akan berbanding lurus dengan semakin kecil jumlah cells. Yang terakhir dilihat dari jumlah nodenya, yang merupakan proses pencarian node yang akan dilewati dari titik awal samapi titik akhir. Berikut ini terdapat pengujian performansi pada papan onet *fruit* ordo 6x7.



Gambar 4.10 Pengujian Performansi ordo 6x7

Hasil pengujian performansi pada gambar 4.10 dapat dilihat pada tabel 4.3 di bawah ini.

Tabel 4.3 Hasil Pengujian Performansi ordo 6x7

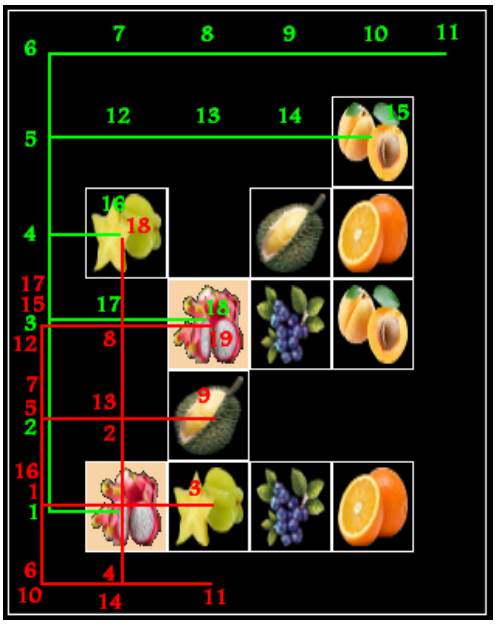
Lang-kah	Titik Awal (x1,y1)	Titik Akhir (x2,y2)	Performansi dengan BFS				Performansi dengan DFS			
			Lines	Cells	Node	Time (ms)	Lines	Cells	Node	Time
1	(0,3)	(0,2)	1	1	2	0.06	3	3	13	0.12
2	(1,0)	(0,0)	1	1	1	0.04	1	1	1	0.04
3	(3,3)	(2,3)	1	1	1	0.04	1	1	1	0.04
4	(1,1)	(2,0)	2	2	7	0.16	3	4	34	0.71
5	(0,4)	(1,2)	2	3	19	0.43	3	5	18	0.16
6	(1,3)	(2,1)	2	3	27	0.6	3	7	91	0.89
7	(3,2)	(3,0)	3	4	20	0.45	3	4	7	0.07
8	(3,4)	(3,1)	1	3	16	0.31	1	3	26	0.23
9	(1,4)	(0,1)	2	4	37	0.71	3	6	18	0.16
10	(2,4)	(2,2)	1	2	9	0.22	3	8	26	0.23

Untuk mengetahui langkah-langkah dari setiap node pada tabel 4.3 di atas, maka diambil beberapa contoh untuk menggambarkan node yang akan dilewati dari titik awal ke titik akhir yakni langkah 1, langkah 5 dan langkah 7.

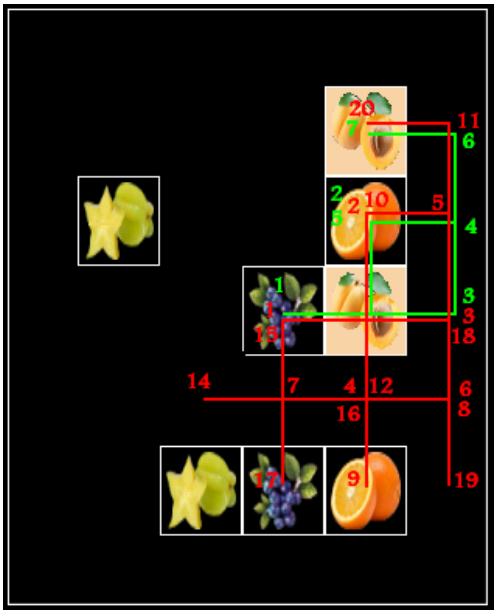
Langkah 1



Langkah 5



Langkah 7



Gambar 4.11 Node yang dilewati pada langkah 1, 5 dan 7

Berdasarkan tabel 4.3 hasil pengujian performansi di atas dapat dilihat bahwa performansi menggunakan BFS lebih dominan menghasilkan jumlah *line* dan jumlah *cell* lebih sedikit dari pada performansi menggunakan DFS, meskipun jika dilihat dari jumlah nodenya tidak selalu lebih sedikit hasilnya. Hal ini dapat disimpulkan bahwa pencarian jalan menggunakan BFS merupakan solusi optimal yang di peroleh meskipun mungkin jumlah node yang perlu diperiksa lebih banyak dibandingkan menggunakan algoritma DFS.

Adapun jika dilihat berdasarkan posisi titik awal (x1, y1) menuju titik akhir (x2,y2). Pencarian akan optimal menggunakan BFS jika posisi titik awal dan titik akhir berdekatan baik secara horizontal (y1=y2), vertikal (x1=x2) maupun diagonal (x2 – x1 <= 2 atau y2 – y1 <= 2 ataupun sebaliknya).

Sedangkan pencarian menggunakan DFS akan optimal jika:

1. Posisi titik akhir berada di sebelah kiri atau atas titik awal, karena proses pencarian dilakukan mulai dari kiri, atas, kanan lalu bawah.

2. Posisi titik awal dan titik akhir berjauhan baik secara horizontal (y1=y2), vertikal (x1=x2) maupun diagonal (x2 – x1 >= 2 atau y2 – y1 >= 2 ataupun sebaliknya).

c. Kesimpulan Pengujian Alpha

Hasil pengujian dari pengujian Aplha yang telah dilakukan, menunjukan bahwa aplikasi yang dibangun sudah memenuhi persyaratan fungsional dan sudah

dapat menghasilkan keluaran yang diharapkan, begitu juga secara perhitungan yang dilakukan dengan algoritma BFS dan DFS telah dihasilkan jalan (*rute*) dari titik awal ke titik tujuan.

4.2.3 Pengujian Betha

Pengujian betha merupakan pengujian yang dilakukan secara objektif, pengujian ini dilakukan oleh user yang akan menggunakan aplikasi yang dibangun. Pengujian dilakukan terhadap 10 orang yang akan berhubungan dengan aplikasi yang dibangun. Dari hasil kuesioner tersebut akan dilakukan perhitungan untuk dapat diambil kesimpulannya terhadap penilaian aplikasi yang dibangun.

Berikut adalah pertanyaan dan hasil kuesioner yang telah dibagikan dengan menggunakan rumus:

$$Y = P / Q * 100\%$$

Keterangan :

Y = Nilai prosentase

P = Banyaknya jawaban responden tiap soal

Q = Jumlah responden

Hasil Pengujian Beta adalah sebagai berikut:

1. Apakah informasi yang disajikan dalam pencarian jalan onet *fruit* ini mudah dimengerti?

☐ Ya ☐ Cukup ☐ Tidak

Kategori Jawaban	Ya	Cukup	Tidak
Frekuensi Jawaban	6	3	1
Presentase Nilai	60%	30%	10%

2. Apakah aplikasi *game* onet *fruit* mudah digunakan?

☐ Ya ☐ Cukup ☐ Tidak

Kategori Jawaban	Ya	Cukup	Tidak
Frekuensi Jawaban	6	3	1
Presentase Nilai	60%	30%	10%

3. Apakah *game* onet *fruit* ini menarik bagi anda?

☐ Ya ☐ Cukup ☐ Tidak

Kategori Jawaban	Ya	Cukup	Tidak
Frekuensi Jawaban	7	3	-
Presentase Nilai	70%	30%	-

4. Apakah anda terpacu untuk memainkan *game* ini kembali?

☐ Ya ☐ Cukup ☐ Tidak

Kategori Jawaban	Ya	Cukup	Tidak
Frekuensi Jawaban	7	3	-
Presentase Nilai	70%	30%	-

5. Bagaimana tampilan dari *game* yang dibangun?

☐ Ya ☐ Cukup ☐ Tidak

Kategori Jawaban	Ya	Cukup	Tidak
Frekuensi Jawaban	6	4	-
Presentase Nilai	60%	40%	-

4.2.3.1 Kesimpulan Pengujian Betha

Berdasarkan dari jawaban yang diberikan oleh responden, bisa ditarik kesimpulan bahwa "Perbandingan algoritma dalam Pencarian Jalan pada *Game* Onet *Fruit*" ini mudah dipahami, aplikasi sudah sesuai dengan tujuan, dan pengoperasian aplikasinya mudah digunakan.