BAB V

IMPLEMENTASI DAN PENGUJIAN APLIKASI

5.1. Spesifikasi Kebutuhan Perangkat

Pada bab ini akan dilakukan implementasi dan pengujian terhadap *game* aplikasi sebagai terapan dari algoritma minimax yang telah dijelaskan sebelumnya. Berikut ini spesifikasi perangkat keras dan perangkat lunak yang dibutuhkan baik dari perangkat PC yang digunakan untuk pengembangan aplikasi, maupun dari perangkat *Smart Phone* yang dipakai *user*.

5.1.1. Spesifikasi Perangkat PC (*Personal Computer*)

1. 1.6 GHz Processor
2. 4 GB RAM
3. 10 GB (NTFS) *of avaiable of* Hard Disk *space*
4. 5400 RPM Hard Drive
5. DirectX 10 *video card running at* 1366 x 768
6. *Operating System* Windows 8 Pro 64-bit (x64) *client versions*

5.1.2. Spesifikasi Perangkat *Smart Phone*

1. Layar 4-point *multi-touch* WVGA dengan resolusi (480x800)
2. Processor Qualcomm Snapdragon™ S4
3. DirectX9 *rendering-capable* GPU
4. 512 MB *of* RAM

5. Accelerometer dengan kompas, ambient light sensor, proximity sensor, Assisted GPS, dan Gyroscope
6. 5-megapixel kamera dengan flash
7. *Six* (6) *dedicated hardware buttons – back, start, search*, 2-*stage camera*, *power/sleep* dan Volume *Up - Down*

5.2. Implementasi

5.2.1. Implementasi Antarmuka

Tabel 5.1 berikut ini merupakan implementasi dari setiap halaman aplikasi permainan Reversi :

Tabel 5.1 Implementasi Antarmuka

Sub Menu	Deskripsi	Nama File
Menu Utama	File program untuk menampilkan halaman utama	MainPage.xaml
Play	File program untuk menampilkan halaman pemilihan level sebelum bermain	LevelSelection.xaml
Beginner	File program untuk memulai permainan dengan level mudah dengan memilih warna bidak terlebih dahulu	ChooseColor.xaml?depth=3

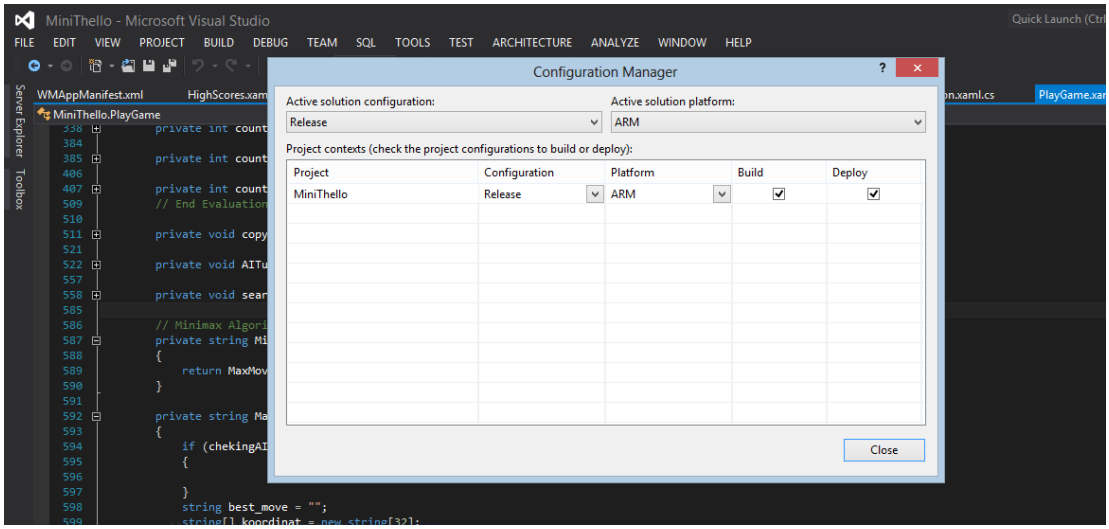
Intermediate	File program untuk memulai permainan dengan level sedang dengan memilih warna bidak terlebih dahulu	ChooseColor.xaml?depth=6
Expert	File program untuk memulai permainan dengan level sulit dengan memilih warna bidak terlebih dahulu	ChooseColor.xaml?depth=9
Black	File program untuk memulai permainan dengan pilihan bidak hitam	PlayGame.xaml
White	File program untuk memulai permainan dengan pilihan bidak putih	PlayGameWhite.xaml
About	File program untuk menampilkan info program dan developer	About2.xaml
High Scores	File program untuk menampilkan informasi score yang tersimpan	HighScores.xaml
Help	File program untuk menampilkan info penggunaan program	Help.xaml

New High Score	File program untuk save nama pemain	NewHighScore.xaml

5.2.2. Implementasi Instalasi (*Release*) Program

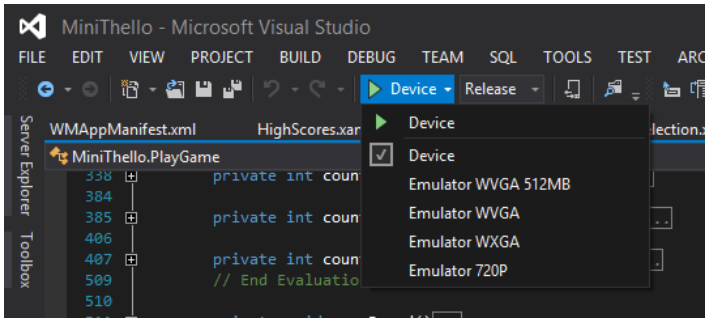
Untuk implementasi instalasi program, penulis menggunakan *Smart Phone* Nokia Lumia 620 dengan spesifikasi seperti yang telah dijelaskan diatas, kemudian sebelum proses instalasi atau *debuging*, *smart phone* tersebut sudah ter-*unlock* terlebih dahulu dan terdaftar di Microsoft sebagai *developer* Windows Phone. Setelah semua kode program berhasil ter-*debug* pada *emulator*, maka berikut implementasi instalasi program ke *Smart Phone* :

1. Ubah terlebih dahulu Platform pada Configuration Manager pada Microsoft Visual Studio (**BUILD -> Configuration Manager**) menjadi ARM (Advanced RISC Machines).



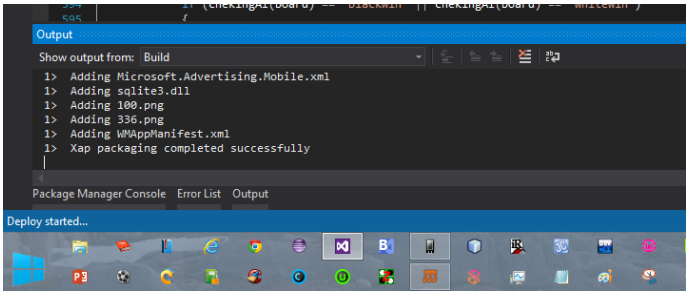
Gambar 5.1. Configurasi *Platform*

2. Kemudian ubah *debuging with Emulator* menjadi *Device*.



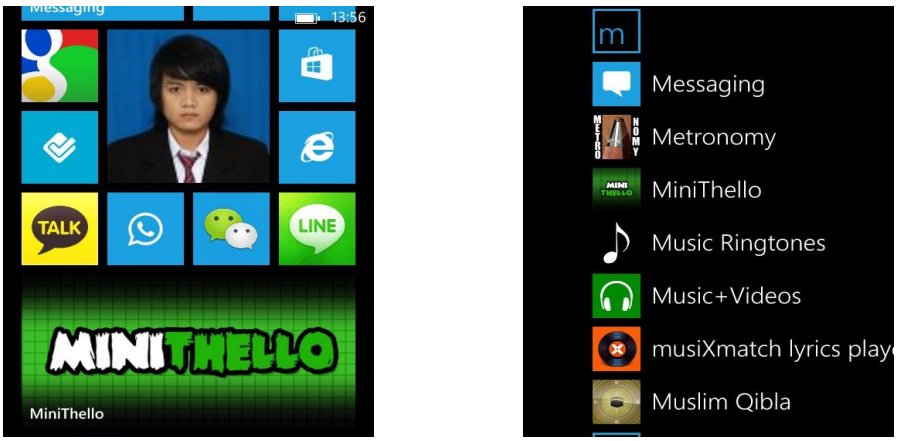
Gambar 5.2 *Debuging* Perangkat

3. Tunggu hingga proses instalasi pada Device selesai.



Gambar 5.3 Proses Instalasi

4. Aplikasi telah terinstal.



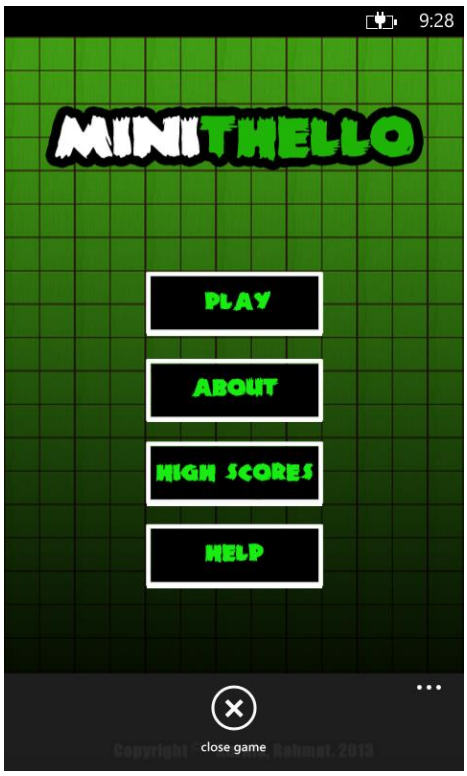
Gambar 5.4 Aplikasi terinstal

File instalasi yang tercipta setelah melakukan langkah diatas yaitu memiliki ekstensi *.xap dengan size “mentahan” berukuran 2.20 MB (2.312.472 bytes).

5.2.3. Implementasi Penggunaan Program

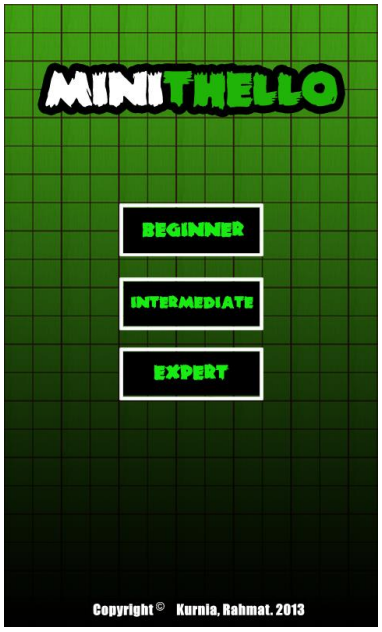
Setelah aplikasi terpasang maka user sudah dapat menggunakan atau memainkan aplikasi tersebut. Berikut implementasi penggunaan programnya :

1. Jalankan aplikasi Reversi (MiniThello) yang sudah ter-*install* kemudian akan muncul halaman utama sebagai berikut :



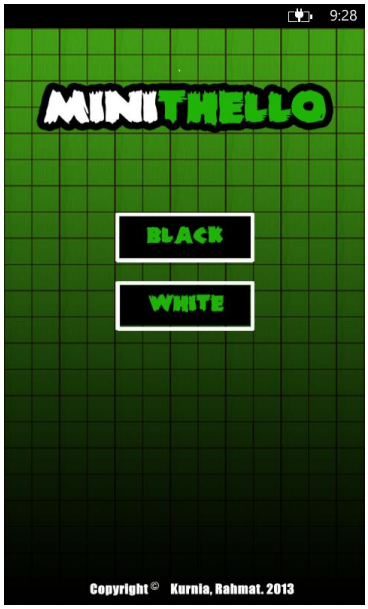
Gambar 5.5 Halaman Utama Program Aplikasi

2. Selanjutnya jika ingin langsung memainkan permainannya klik / tap *button* Play maka akan muncul halaman pemilihan *level* yang diinginkan.



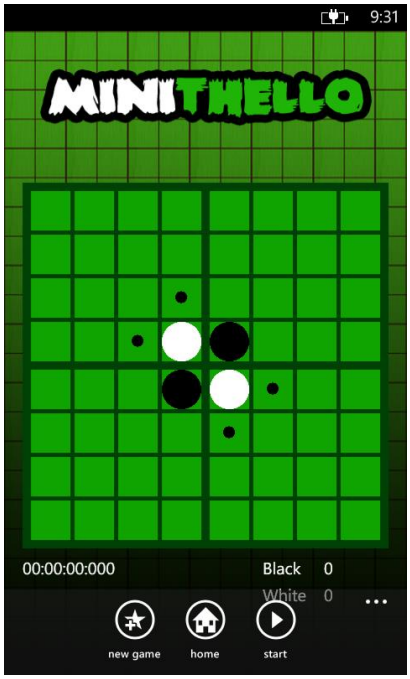
Gambar 5.6 Halaman Pemilihan *Level* Permainan

3. Jika telah memilih *level* yang diinginkan maka akan muncul pilihan untuk memilih warna bidak yang ingin digunakan yaitu hitam atau putih (dengan catatan warna bidak hitam akan bermain terlebih dahulu).



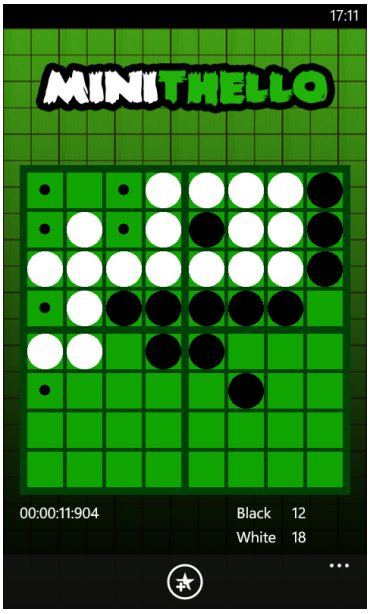
Gambar 5.7 Halaman Pemilihan warna bidak

4. Jika telah memilih warna bidak / koin yang akan digunakan maka akan muncul papan permainan berukuran 8x8. *Player* hanya dapat melangkah pada kotak yang memiliki *mobility* saja yang artinya hanya dapat melangkah pada kotak yang terdapat bulatan hitam atau putih yang kecil saja. Jika memilih warna bidak putih maka untuk memulai permainan harus diawali dengan *button* "start" pada Application Bar.



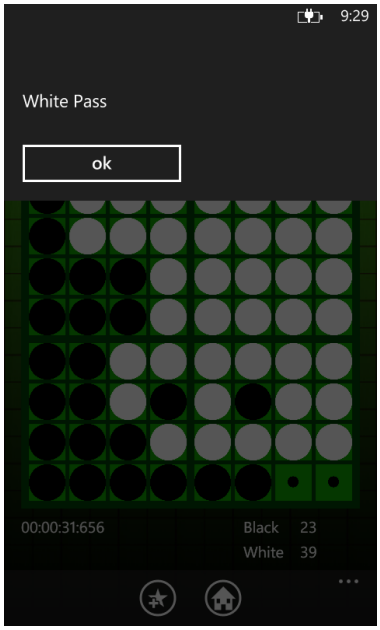
Gambar 5.8 Kondisi awal (*default*) papan main dengan giliran hitam

5. Setelah itu giliran komputer (AI) yang bermain dengan bidak (koin) warna putih.



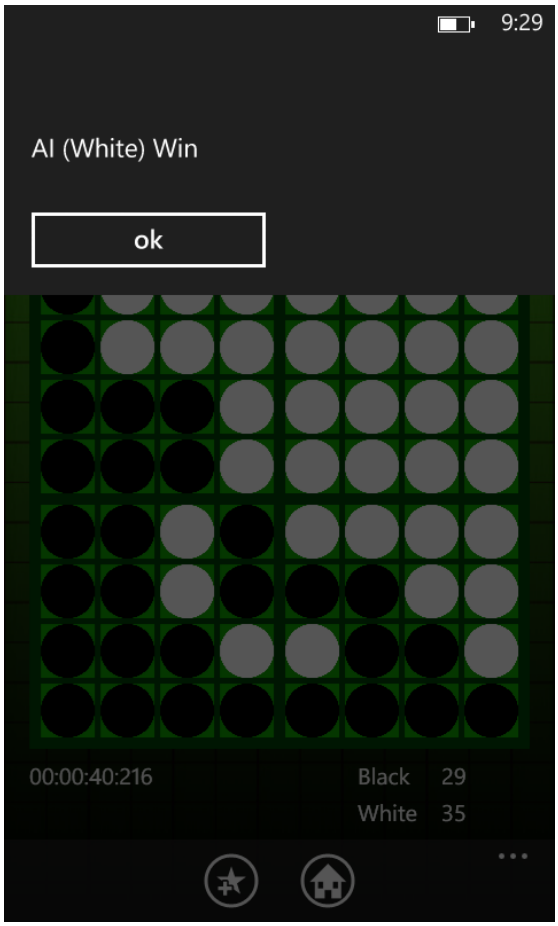
Gambar 5.9 Contoh Langkah Komputer (AI)

6. Jika didapatkan sudah tidak ada lagi *mobility* bagi pemain baik *user* atau komputer maka akan muncul peringatan "pass" yang berarti permainan dilanjutkan dengan masih di pemain sebelumnya (bisa hitam atau putih).



Gambar 5.10 Peringatan "pass" bagi pemain (*user* atau komputer)

7. Sampai pada kondisi dimana seluruh papan main sudah dipenuhi bidak (koin), maka permainan berakhir dan dilakukan perhitungan koin yang didapat dan mememtukan pemenang dari permainan ini.



Gambar 5.11 Penentuan Pemenang

5.3. Pengujian Aplikasi

Pengujian merupakan tahapan penting dalam pembangunan aplikasi atau perangkat lunak. Pengujian dilakukan untuk mengetahui kelayakan suatu perangkat lunak sehingga perangkat lunak tersebut dapat berjalan sesuai dengan yang diharapkan dan mempunyai kualitas yang handal.

5.3.1. Pengujian *White Box*

Pengujian *White Box* merupakan pengujian yang berkaitan dengan struktur kontrol prosedural pada perangkat lunak atau aplikasi, dengan pengujian *White Box logical path* (jalur logika) perangkat lunak akan diuji dengan menyediakan *test case* yang akan mengerjakan sekumpulan kondisi atau perulangan secara spesifik. 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. Dalam hal ini pengujian secara *White Box* dibatasi, penulis hanya menguji alur algoritma prosedur minimax saja (**MaxMove** *function* dan **MinMove** *function*).

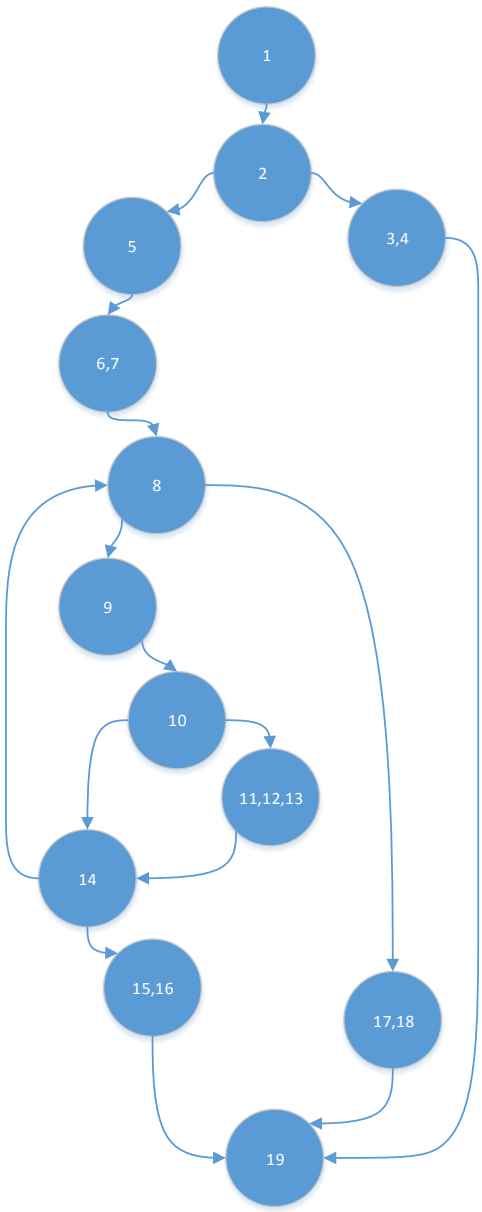
Berikut *pseudocode* fungsi **MaxMove** :

```
1  : MaxMove (Integer[] board , Integer alpha, Integer beta) {
2  :  if (GameEnded(board) || DepthLimit()) {
3  :    return EvalGameState(board, MAX);
4  :  }
5  :  else {
6  :    best_koordinat ← {};
7  :    moves ← GenerateMoves(board);
8  :    ForEach move in moves {
9  :      koordinat ← MinMove(ApplyMove(move,board), alpha, beta);
10 :      if (valueEval(koordinat) > valueEval(best_koordinat)) {
11 :        best_koordinat ← koordinat;
12 :        alpha ← valueEval(koordinat);
13 :      }
14 :      if (beta > alpha)
15 :         return best_koordinat;
16 :    }
17 :    return best_koordinat;
18 :  }
19 : }
```

Sedangkan untuk *pseudocode* fungsi **MinMove** yaitu :

```
1  : MinMove (Integer[] board , Integer alpha, Integer beta) {
2  :  if (GameEnded(board) || DepthLimit()) {
3  :    return EvalGameState(board, MIN);
4  :  }
5  :  else {
6  :    best_koordinat ← {};
7  :    moves ← GenerateMoves(board);
8  :    ForEach move in moves {
9  :      koordinat ← MaxMove(ApplyMove(move,board), alpha, beta);
10 :      if (valueEval(koordinat) > valueEval(best_koordinat)) {
11 :        best_koordinat ← koordinat;
12 :        beta ← valueEval(koordinat);
13 :      }
14 :      if (beta < alpha)
15 :         return best_koordinat;
16 :    }
17 :    return best_koordinat;
18 :  }
19 : }
```

Pada kedua *pseudocode* tersebut tidak terlihat perbedaan yang signifikan, jadi jika kedua *pseudocode* tersebut diubah menjadi bentuk *flow graph*, akan seperti gambar berikut :



Gambar 5.12 *Flow Graph* fungsi MaxMove dan MinMove

Dari *flow graph* diatas dapat dihitung *cyclomatic complexity* yaitu mempunyai 5 *independent path*. Dihitung menggunakan rumus V(G) = E – N + 2 = 16 – 13 + 2 = 5, dimana E merupakan jumlah *edge* pada *flow graph* dan N merupakan jumlah *node* pada *flow graph*. Berikut kelima *independent path* pada *flow graph* diatas :

Path 1 : 1 – 2 – 3,4 – 19

Path 2 : 1 – 2 – 5 – 6,7 – 8 – 9 – 10 – 11,12,13 – 14 – 15,16 – 19

Path 3 : 1 – 2 – 5 – 6,7 – 8 – 9 – 10 – 14 – 15,16 – 19

Path 4 : 1 – 2 – 5 – 6,7 – 8 – 9 – 10 – 11,12,13 – 14 – 8 – 17,18 – 19

Path 5 : 1 – 2 – 5 – 6,7 – 8 – 9 – 10 – 14 – 8 – 17,18 – 19

5.3.2. Pengujian *Black Box*

Pengujian *Black Box* merupakan pengujian yang berfokus pada fungsionalitas perangkat lunak atau aplikasi, fungsionalitas perangkat lunak ini berkaitan dengan masukan-masukan *user* dan kesesuaiannya dengan keluaran yang diinginkan. Dalam hal ini pengujian *Black Box* yang dilakukan terhadap aplikasi permainan Othello atau Reversi ini berkaitan dengan kesesuaian setiap menu dan mobilitas jalannya permainan. Berikut tabel pengujian alpha (*Black Box*) :

Tabel 5.2 Pengujian Aplikasi

Kasus dan Hasil Uji (Data Normal)			
Item Pengujian	**Realisasi yang diharapkan**	**Hasil Pengujian**	**Kesimpulan**
Klik Button Play	Masuk ke Page Memilih Level Permainan	Memilih Level Permainan	[X] Diterima [ ] Ditolak

Klik Button About	Untuk menampilkan informasi Aplikasi dan Pembuat	Tombol berfungsi sesuai yang diharapkan	[X] Diterima [ ] Ditolak
Klik Button High Scores	Untuk Menampilkan *score* pemain 15 teratas	Tombol berfungsi sesuai yang diharapkan	[X] Diterima [ ] Ditolak
Klik Button Help	Untuk Menampilkan cara bermain atau petunjuk permainan	Tombol berfungsi sesuai yang diharapkan	[X] Diterima [ ] Ditolak
Klik Button Beginner	Untuk memulai permainan dengan *depth* 3	Permainan dimulai dengan *depth* 3	[X] Diterima [ ] Ditolak
Klik Button Intermediate	Untuk memulai permainan dengan *depth* 6	Permainan dimulai dengan *depth* 6	[X] Diterima [ ] Ditolak
Klik Button Expert	Untuk memulai permainan dengan *depth* 9	Permainan dimulai dengan *depth* 9	[X] Diterima [ ] Ditolak
Klik Application Bar New Game	Untuk Mengawali Permainan Kembali	Permainan Dimulai Kembali	[X] Diterima [ ] Ditolak

Klik Application Bar Play Again	Untuk memulai permainan kembali dengan memilih level terlebih dahulu	Application Bar berfungsi sesuai dengan harapan	[X] Diterima [ ] Ditolak
Tap mobility pada board	Untuk menyimpan koin hitam pada papan main	Berfungsi sesuai dengan harapan	[X] Diterima [ ] Ditolak
Klik Oke pada saat Player Pass	Giliran kembali player sebelumnya	Berfungsi sesuai dengan harapan	[X] Diterima [ ] Ditolak
Klik Oke pada saat permainan berakhir	Menampilkan kondisi terakhir permainan (menang,kalah,seri)	Berfungsi sesuai dengan harapan	[X] Diterima [ ] Ditolak
Klik Button Save pada saat nama pemain terisi	Menampilkan halaman Highscore	Berfungsi sesuai dengan harapan	[X] Diterima [ ] Ditolak
Kasus dan Hasil Uji (Data tidak Normal)			
Aktivitas Pengujian	**Realisasi yang diharapkan**	**Hasil Pengujian**	**Kesimpulan**
Klik Button Save pada saat nama pemain kosong	Menampilkan alert nama tidak boleh kosong	Hasil sesuai dengan harapan	[X] Diterima [ ] Ditolak