BAB IV
IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1. Lingkungan 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 sistem 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 unblockme* dalam menerapkan algoritma *backtracking.*

4.2. Spesifikasi Kebutuhan Perangkat Keras

Adapun spesifikasi perangkat keras yang direkomedasikan untuk implementasi algoritma backtracking pada game ini adalah sebagai berikut :

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

4.3. Spesifikasi Kebutuhan Perangkat Lunak

Adapun spesifikasi perangkat lunak yang direkomendasikan untuk implementasi algoritma backtracking pada game ini adalah sebagai berikut :

Tabel 4.2 Spesifikasi perangkat lunak

Spesifikasi Perangkat Lunak
Windows 7
Visual Studio

4.4. Tampilan Antar Muka

Pada tampilan antar muka ini diperlihatkan permainan dalam bentuk gambar , Algoritma yang telah dirancang pada sistem, diimplementasikan kedalam program dengan antarmuka sebagai berikut :

4.4.1. Tampilan *Splash Screen*



Gambar 4.1 Tampilan Splash Screen

Gambar tersebut merupakan tampilan awal atau loading pada saat user membuka permainan.

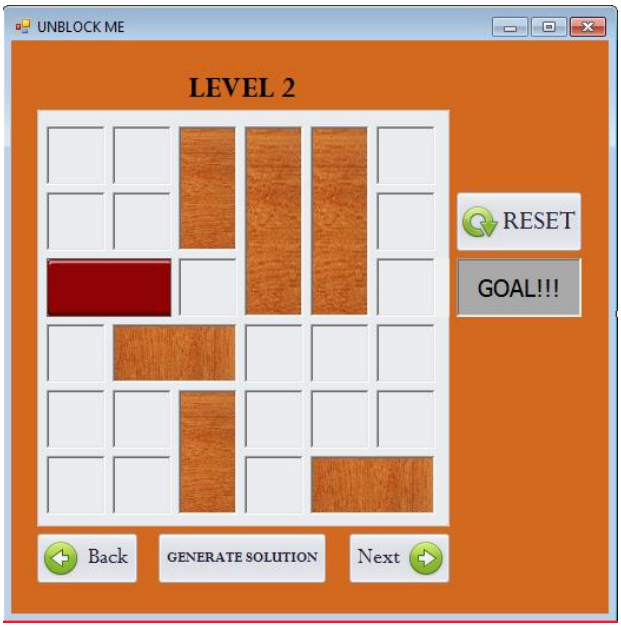
4.4.2. Tampilan Level 1



Gambar 4.2 Tampilan Level 1

Gambar tersebut merupakan tampilan level 1 yang harus diselesaikan oleh pemain.

4.4.3. Tampilan Level 2



Gambar 4.3 Tampilan Level 2

Gambar tersebut merupakan tampilan level 2 yang harus diselesaikan oleh pemain.

4.4.4. Tampilan Level 3



Gambar 4.4 Tampilan Level 3

Gambar tersebut merupakan tampilan level 3 yang harus diselesaikan oleh pemain.

4.4.5. Tampilan Berhasil

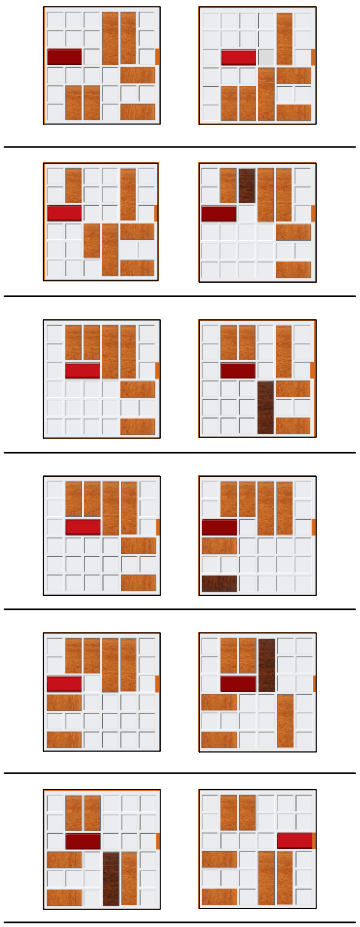


Gambar 4.5 Tampilan Berhasil

Gambar tersebut merupakan tampilan ketika pemain berhasil menyelesaikan sebuah level.

4.5. Pengujian Simulasi

Pengujian dilakukan bertujuan untuk mendapatkan jalur solusi dari susunan balok yang ada pada game *UnblockMe*. Pencarian dilakukan berdasarkan jumlah balok yang masing-masing balok memiliki pola langkah pencarian yang berbeda.



Gambar 4.6 Pola Penyelesaian pada Level 1

4.5.1 Pengujian Simulasi Game pada Level 1

Pengujian simulasi game *Unblock Me* dilakukan pada level 1 yang memiliki tingkat kesulitan rendah agar dapat dengan mudah untuk dijelaskan secara lebih terperinci.



Gambar 4.7 Tampilan Level 1

4.5.2. Pengujian Matrix

Untuk mempermudah lajur pencarian solusi, terlebih dahulu dilakukan perubahan bentuk dari sebuah balok kedalam sebuah matrix yang kemudian diberikan inisialisasi berupa angka sesuai dengan banyaknya balok yang tersedia pada level tersebut. Berikut ini adalah hasil dari tampilan matrix saat digenerate :

```
0 0 0 3 0 0
0 0 2 3 0 9
1 1 2 3 0 9
0 0 0 0 0 8
0 4 6 6 0 8
0 4 5 5 7 7
---------simpul utama ---------
0 0 0 3 0 9
0 0 2 3 0 9
1 1 2 3 0 0
0 0 0 0 0 8
0 4 6 6 0 8
0 4 5 5 7 7
---------simpul utama ---------
0 0 0 3 0 9
0 0 2 3 0 9
1 1 2 3 0 8
0 0 0 0 0 8
0 4 6 6 0 0
0 4 5 5 7 7
---------simpul utama ---------
0 0 0 3 0 9
0 0 2 3 0 9
1 1 2 3 0 8
0 0 0 0 0 8
0 4 0 0 6 6
0 4 5 5 7 7
---------simpul utama ---------
0 0 0 3 0 9
0 0 2 3 0 9
1 1 2 3 0 8
0 0 0 0 0 8
0 4 0 6 6 0
0 4 5 5 7 7
---------simpul utama ---------
0 0 0 3 0 9
0 0 2 3 0 9
1 1 2 3 0 0
0 0 0 0 0 8
0 4 0 6 6 8
0 4 5 5 7 7
---------simpul utama ---------
0 0 0 3 0 0
0 0 2 3 0 9
1 1 2 3 0 9
0 0 0 0 0 8
0 4 0 6 6 8
0 4 5 5 7 7
---------simpul utama ---------
0 0 0 3 0 0
0 0 2 3 0 9
1 1 2 3 0 9
0 4 0 0 0 8
0 4 0 6 6 8
0 0 5 5 7 7
```

```
0 4 2 3 0 9
0 4 2 3 0 9
1 1 0 3 0 8
0 0 0 0 0 8
6 6 0 0 0 0
0 0 5 0 7 7
---------simpul utama ---------
0 4 2 3 0 9
0 4 2 3 0 9
1 1 0 3 0 8
0 0 0 0 0 8
6 6 0 0 0 0
0 0 5 7 7 0
---------simpul utama ---------
0 4 2 3 0 9
0 4 2 3 0 9
1 1 0 3 0 8
0 0 0 0 0 8
0 0 0 0 6 6
0 0 5 7 7 0
---------simpul utama ---------
0 4 2 3 0 9
0 4 2 3 0 9
1 1 0 3 0 8
0 0 0 0 0 8
0 0 0 0 6 6
0 0 5 0 7 7
---------simpul utama ---------
0 4 2 3 0 9
0 4 2 3 0 9
1 1 0 3 0 8
0 0 0 0 0 8
0 6 6 0 0 0
0 0 5 0 7 7
---------simpul utama ---------
0 4 2 0 0 9
0 4 2 0 0 9
1 1 0 0 0 8
0 0 0 3 0 8
0 6 6 3 0 0
0 0 5 3 7 7
---------simpul utama ---------
0 4 2 0 0 9
0 4 2 0 0 9
1 1 0 0 0 0
0 0 0 3 0 8
0 6 6 3 0 8
0 0 5 3 7 7
---------simpul utama ---------
0 4 2 0 0 9
0 4 2 0 0 9
0 0 0 0 1 1
0 0 0 3 0 8
0 6 6 3 0 8
0 0 5 3 7 7
```

Gambar 4.8 Tampilan Level 3 bentuk Matrix

Keterangan :

- Balok merah ditandai dengan angka 1
- Selebihnya dari inisialisasi angka 2 s/d 6, merupakan balok biasa

Matriks di atas menggambarkan alur pencarian solusi yang dilakukan oleh aplikasi setelah tombol *generate solution* di klik. Adapun aturan yang diterapkan pada balok yaitu :

1. Balok merah merupakan balok user yang harus dikeluarkan dari petak papan game menuju petak goal disebelah kanan. Untuk balok user ditandai dengan angka 1 pada matriks.
2. Balok kayu merupakan balok pendukung aplikasi yang berguna untuk memberikan halangan dan rintangan terhadap balok user. Untuk balok kayu ditandai dengan angka > 1 pada matriks.

4.6. Pengujian Sistem

Pengujian system 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.

4.6.1 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 *backtracking* untuk mengetahui kinerja logika berdasarkan *flowchart* yang telah dibuat pada tahap analisis. Pengujian *white box* meliputi beberapa tahap, yaitu :

1. Pengujian Algoritma *Backtracking* Pada *Game Unblock Me.*

Pada pengujian ini, algoritma *backtracking* akan diuji dalam pembangkitan soal pada *game Unblock Me*. Berikut merupakan pengujian algoritma *backtracking* pada pembangkit soal dengan menggunakan *white box.*

Mengubah *source code* menjadi *flowchart,* kemudian diubah kedalam *flowgraph.* Berikut *source code* algoritma *backtracking* dalam generate solution permainan :

Tabel 4.3 *Source code* algoritma *backtracking* pada generate solution

1	`#region inisialisasi matrix`
	`for (int a = 0; a <= 5; a++)`
	`{`
	`for (int b = 0; b <= 5; b++)`
	`{`
	`matrix[a, b] = 0;`
	
	`}`
	`#endregion`

2	`antrian.Add(matrix);`
3	`front = antrian[antrian.Count - 1];`
4	`for (int var = 0; var < kunjung.Count; var++)`

5	`//cetak front`
	`for (int a = 0; a <= 5; a++)`
	`{`
	`for (int b = 0; b <= 5; b++)`
	`{`
	`Console.Write(front[a, b] + " ");`
	`// Console.Write(antrian[init][a, b] + " ");`
	
	`}`
	`Console.WriteLine();`
	`}`
	`Console.WriteLine("---------simpul utama ---------");`

6	`#region move balok auto mode`

7	`if (finish == true)`
	`{`
	`jawaban.Add(front);`

```
            marker = false;
            //posisi di kotak goal
            balokUser.Left = 389;
            blkUser = "GOAL";
            bgGoal.BackColor = Color.Firebrick;
              labelControl1.BackColor = Color.Firebrick;

              MessageBox.Show("selesai");
                            // return true;
                        }

               // jika tidak
                            else{
              //catat simpul yang sedang d kunjungi

                        kunjung.Add(front);

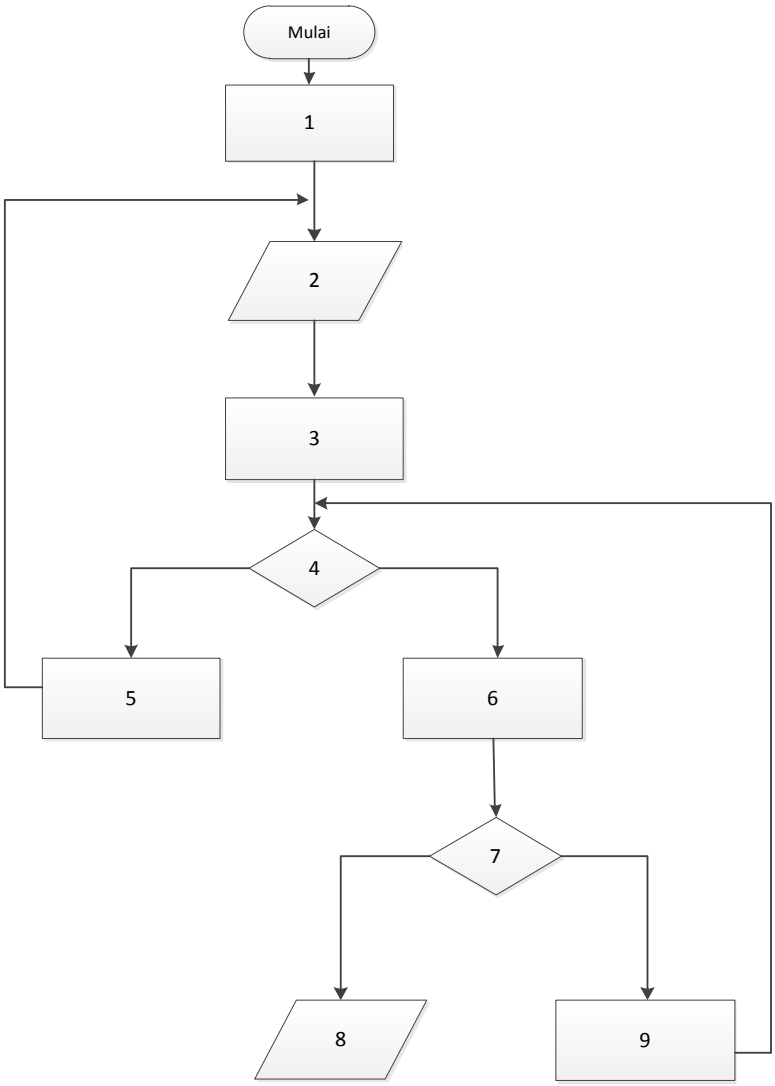
 string[,] step = new string[25, 2];
                            int i = 0;
                            int j = 0;
                            // balok user
                            #region balok user

                            }
                            #endregion
                            // end pencarian langkah

for (int val = 0; val <= (i - 1); val++)
     {
                                int[,] matrik = new int[6, 6];
                                for (int bts = 0; bts <= 5; bts++)
                                {
                                    for (int bts2 = 0; bts2 <= 5; bts2++)
                                    {
                                        matrik[bts, bts2] = front[bts, bts2];
                                    }
                                }

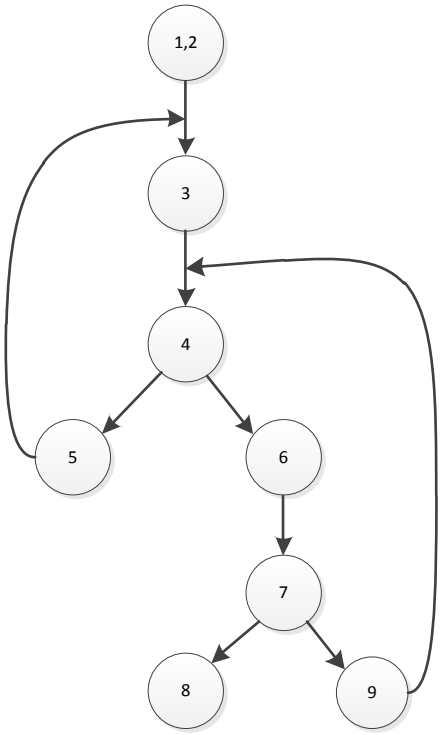
                                if (step[val, 0].Substring(0, 1) == "1")
                                {
                                    #region balok user
                                    if (step[val, 1] == "2,0-2,1")
                                    {
                                        for (int x = 0; x < 6; x++)
                                        {
                                            if (x == 0 || x == 1)
                                            {
                                                matrik[2, x] = 1;
                                            }
                                            else
                                            {
                                                if (matrik[2, x] == 1)
```

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



Gambar 4.9 *Source code* menjadi *flowchart*

Mengubah *flowchart* menjadi *flowgraph*, dapat dilihat pada gambar 4.3.



Gambar 4.10 *Flowchart* menjadi *flowgraph*

Cyclomatic complexity

Dari diagram alir pada gambar 4.3 dapat dihitung *cyclomatic complexity*, yakni :

V(G) = 9 busur - 8 simpul + 2

V(G) = 3

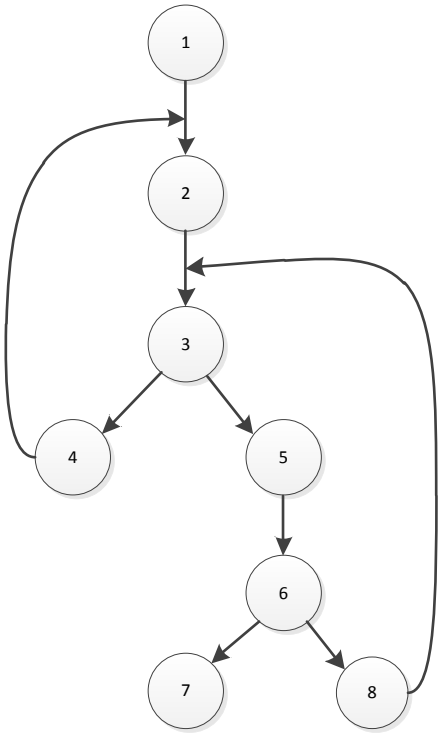
Predicate node

V(G) = Jumlah *node* yang memiliki lebih dari 1 lajur + 1

V(G) = 2 *predicate node* + 1

V(G) = 3

Penomoran ulang *flowgraph*, dapat dilihat pada gambar 4.4.



Gambar 4.11 Penomoran ulang *flowgraph*

Graph matrix, dapat dilihat pada tabel 4.4.

Tabel 4.4*Graph matrix*

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	Perhitungan
1		1												0
2			1											0
3				1	1									1
4		1												0
5						1								0
6							1	1						1
7														0
8			1											0
Jumlah (x)														2

V(G) = x + 1

V(G) = 2 + 1

V(G) = 3

4.6.2 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 . 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.

4.6.2.1 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*
	Menggeser kotak	*Black box*
	Menekan tombol *solve* / *Generate*	*Black box*

4.6.2.2 Kasus Dan Hasil Pengujian *(BlackBox)*

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
	Menggeser Kotak	Sistem dapat menggeser kotak secara horizontal atau vertical.	[√] Berhasil [ ] Tidak
	Menekan tombol *solve*	Sistem memberikan jawaban penyelesaian untuk menggeser balok-balok sehingga balok user ada pada posisi goal.	[√] Berhasil [ ] Tidak

4.7 Kesimpulan

Dari hasil pengujian dapat disimpulkan bahwa algoritma *backtracking* dapat berfungsi dalam generate solution pada game Unblock Me. Pada generate solution, algoritma *backtracking* dapat bekerja dengan baik dalam mengenerate solution permainan pada *game Unblockme* ini. Jawaban Penyelesaian yang dihasilkan menggunakan algoritma *backtracking* disesuaikan dengan syarat permainan *Unblock me* yaitu balok hanya bisa digeser secara horizontal atau vertical sesuai posisi balok dan sesuai dengan inisialisasi balok di setiap levelnya.