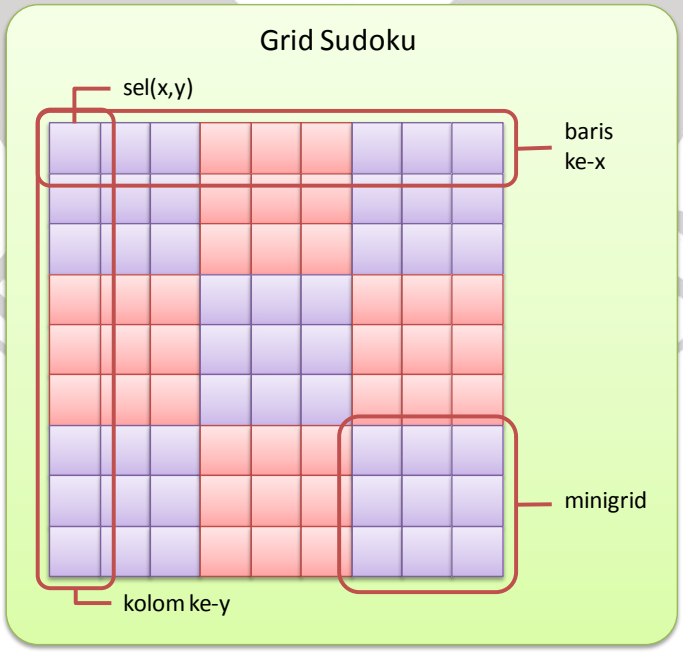
BAB III
ANALISIS DAN PERANCANGAN

Pada bab ini akan dibahas mengenai analisis dan perancangan pada sistem yang dibangun, yaitu penerapan algoritma *Backtrack* dalam membangkitkan elemen awal permainan Sudoku.

3.1 Analisis

3.1.1 Analisis Arena Permainan

Pada sistem yang diberi nama **Sudoku Break** ini, diharapkan pemain dapat menyelesaikan permainan Sudoku yang beberapa elemen awalnya diketahui. Permainan sudoku ini hanya dapat dimainkan oleh satu pemain (*single player*). Papan permainan ini berupa *grid* berukuran 9x9 yang terdiri dari 81 sel, sembilan baris, sembilan kolom dan sembilan *minigrid* berukuran 3x3.



Gambar 3.1 Ilustrasi papan permainan Sudoku

3.1.2 Analisis Algoritma

Seperti yang sudah dipaparkan dalam bab-bab sebelumnya, algoritma yang akan digunakan dalam membangun aplikasi ini adalah algoritma *Backtrack*. Penerapan algoritma *Backtrack* dalam menentukan elemen awal permainan Sudoku adalah sebagai berikut:

1. Algoritma dimulai dari sel kosong di baris pertama (x=0) dan kolom pertama (y=0) pada matriks Sudoku berukuran 9x9.
2. Pilih nilai `num` secara acak, dengan `num` ∈ {1, 2, 3, 4, 5, 6, 7, 8, 9}.
3. Jika nilai `num` yang dipilih memenuhi aturan sudoku (*valid*), maka isilah sel kosong tersebut dengan nilai `num`, dan lanjutkan pemeriksaan ke sel kosong berikutnya.
4. Jika nilai `num` yang dipilih tidak memenuhi aturan sudoku, uji kembali dengan nilai `num` acak lainnya yang belum terpilih.
5. Jika seluruh nilai `num` telah teruji, dan tidak ada nilai `num` yang memenuhi aturan sudoku, maka lakukan *backtrack* ke sel sebelumnya.
6. Sel ini akan diuji kembali dengan nilai `num` baru berdasarkan aturan sudoku.
7. Lakukan cara yang sama dengan poin nomor 3.
8. Proses di atas akan dilakukan terus menerus secara rekursif hingga seluruh sel kosong pada matriks berisi nilai-nilai `num` yang *valid*.
9. Pilih posisi sel secara acak, kemudian hapus nilai `num` pada sel tersebut.
10. Lakukan poin nomor 9 secara terus menerus, hingga diperoleh matriks Sudoku berukuran 9x9 dengan beberapa nilai `num` ditampilkan. Banyaknya nilai `num` yang ditampilkan tergantung dari *level* permainan yang dipilih pemain, seperti yang dapat dilihat pada tabel 3.1. Pada tabel tersebut, terlihat bahwa, semakin sedikit jumlah elemen awal yang ditampilkan, maka tingkat kerumitan soal (level) permainan Sudoku akan semakin sulit. Nilai-nilai `num` yang ditampilkan ini merupakan elemen-elemen awal permainan Sudoku yang

berfungsi sebagai petunjuk awal bagi pemain untuk dapat menyelesaikan permainan Sudoku.

Tabel 3.1 Tabel banyaknya elemen awal yang ditampilkan pada setiap Level

No.	Level	Jumlah Elemen Awal
1	Easy	35-40
2	Medium	29-34
3	Hard	23-28

Berikut ini merupakan *pseudocode* pembangkitan elemen awal permainan Sudoku, yaitu:

1. Deklarasi konstanta umum

```
DEKLARASI (umum)
  grid : array[0..NBaris-1] of array[0..NKolom-1] of
  integer
  selGrid : array[0..Nbaris*Nkolom-1] of integer
  NBaris : integer = 9
  NKolom : integer = 9
```

Bagian ini merupakan deklarasi variabel umum yang akan digunakan, yaitu:

a. `grid` merupakan *array* dua dimensi bertipe integer berupa matriks berukuran NBaris x NKolom = 9x9. Indeks *array* pada `grid` dimulai dari 0.
b. `selGrid` merupakan *array* berukuran NBaris*NKolom = 9*9 = 81 buah elemen yang bertipe integer. Indeks *array* pada `selGrid` dimulai dari 0.
c. `NBaris` menyatakan jumlah atau ukuran baris pada matriks Sudoku.
d. `NKolom` menyatakan jumlah atau ukuran kolom pada matriks Sudoku.

2. Implementasi fungsi pembatas

Fungsi `SesuaiAturanSudoku` merupakan fungsi untuk memeriksa apakah nilai `num` yang diberikan sesuai dengan fungsi pembatas atau tidak. Fungsi pembatas ini berdasarkan batasan-batasan pada aturan permainan Sudoku, yaitu:

1. Tidak boleh ada angka yang berulang pada baris.
2. Tidak boleh ada angka yang berulang pada kolom.
3. Tidak boleh ada angka yang berulang pada *minigrid*.

```
FUNCTION sesuaiAturanSudoku(input baris, kolom, num : integer) → boolean
{fungsi untuk menentukan apakah nilai num yang dimasukkan pada sel memenuhi aturan Sudoku atau tidak. bernilai false jika tidak sesuai aturan, true jika sesuai aturan}
DEKLARASI (lokal)
    x, y, i, j : integer
    minigrid, startBarisMini, startKolomMini : integer
```

Bagian ini merupakan deklarasi lokal yang akan digunakan pada fungsi `sesuaiAturanSudoku`, yaitu:

a. `minigrid` merupakan bagian dari *grid* Sudoku dengan ukuran yang lebih kecil yaitu 3x3.
b. `startBarisMini` merupakan posisi awal baris *minigrid* di dalam *grid*.
c. `startKolomMini` merupakan posisi awal kolom *minigrid* di dalam *grid*.

```
ALGORITMA
1.  {memeriksa baris}
2.  for y ← 0 to (Nkolom-1) do
3.     if grid[baris][y] = num then
4.        return false
5.     endif
```

```
6.  endfor
```

Pada baris 2-6, fungsi akan memeriksa apakah terdapat nilai `num` yang sama pada baris yang sama atau tidak. Jika ya, maka nilai `num` tersebut tidak memenuhi aturan Sudoku dan fungsi akan mengembalikan nilai `false`. Jika tidak, maka nilai `num` tersebut memenuhi aturan Sudoku dan fungsi akan mengembalikan nilai `true`.

```
7.  {memeriksa kolom}
8.  for x ← 0 to (Nbaris-1) do
9.    if grid[x][kolom] = num then
10.     return false
11.   endif
12. endfor
```

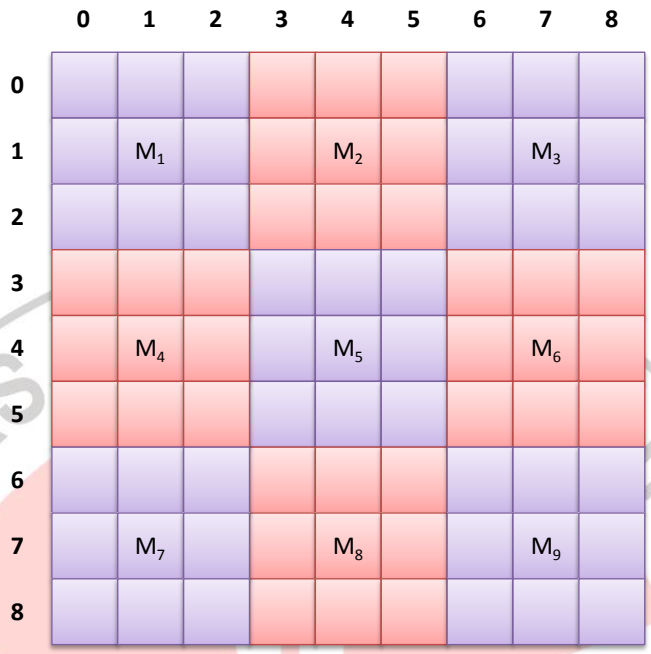
Pada baris 8-12, fungsi akan memeriksa apakah terdapat nilai `num` yang sama pada kolom yang sama atau tidak. Jika ya, maka nilai `num` tersebut tidak memenuhi aturan Sudoku dan fungsi akan mengembalikan nilai `false`. Jika tidak, maka nilai `num` tersebut memenuhi aturan Sudoku dan fungsi akan mengembalikan nilai `true`.

```
13. {cek posisi angka pada minigrid}
14. if (kolom < 3) then
15.   if (baris < 3) then
16.     minigrid ← 1
17.     startBarisMini ← 0
18.     startKolomMini ← 0
19.   endif
20.   else
21.   if (baris >= 3 & baris < 6) then
22.     minigrid ← 2
23.     startBarisMini ← 3
```

```
24.       startKolomMini ← 0
25.    endif
26.    else
27.    if (baris >= 6) then
28.       minigrid ← 3
29.       startBarisMini ← 6
30.       startKolomMini ← 0
31.    endif
32. endif
33. else
34. if (kolom >= 3 & kolom < 6) then
35.    if (baris < 3) then
36.       minigrid ← 4
37.       startBarisMini ← 0
38.       startKolomMini ← 3
39.    endif
40.    else
41.    if (baris >= 3 & baris < 6) then
42.       minigrid ← 5
43.       startBarisMini ← 3
44.       startKolomMini ← 3
45.    endif
46.    else
47.    if (baris >= 6) then
48.       minigrid ← 6
49.       startBarisMini ← 6
50.       startKolomMini ← 3
51.    endif
52. endif
53. else
54. if (kolom >= 6) then
55.    if (baris < 3) then
```

```
56.       minigrid ← 7
57.       startBarisMini ← 0
58.       startKolomMini ← 6
59.    endif
60.    else
61.    if (baris >= 3 & baris < 6) then
62.       minigrid ← 8
63.       startBarisMini ← 3
64.       startKolomMini ← 6
65.    endif
66.    else
67.    if (baris >= 6) then
68.       minigrid ← 9
69.       startBarisMini ← 6
70.       startKolomMini ← 6
71.    endif
72. endif
```

Pada baris 14-72, fungsi akan memeriksa posisi nilai `num` pada *minigrid* Sudoku. Sebagai contoh, jika nilai `num` berada pada sel di kolom ke-3 dan baris ke-5, maka dari *pseudocode* tersebut diperoleh informasi bahwa nilai `num` berada pada *minigrid* ke-5. Dari baris 42-44, diperoleh informasi bahwa *minigrid* ke-5 dimulai dari baris ke-3 dan kolom ke-3. Informasi ini nantinya akan digunakan untuk memeriksa apakah nilai `num` pada sel tersebut memenuhi aturan Sudoku pada *minigrid* atau tidak. Untuk lebih jelasnya dapat dilihat pada Gambar 3.2.



Gambar 3.2 Posisi *Minigrid* pada Grid Sudoku

```
73. {memeriksa minigrid}
74. for i ← startBarisMini to startBarisMini+2 do
75.    for j ← startKolomMini to startKolomMini+2 do
76.       if grid[i][j] = num then
77.          return false
78.       endif
79.    endfor
80. endfor
```

Pada baris 74-80, fungsi akan memeriksa apakah terdapat nilai `num` yang sama pada *minigrid* yang sama. Jika ya, maka nilai `num` tersebut tidak memenuhi aturan Sudoku dan fungsi akan mengembalikan nilai `false`. Jika tidak, maka nilai `num` tersebut memenuhi aturan Sudoku dan fungsi akan mengembalikan nilai `true`.

```
81. return true
```

Jika semua batasan telah diperiksa dan ternyata nilai num tersebut memenuhi semua aturan Sudoku, maka fungsi akan mengembalikan nilai true.

3. Implementasi fungsi *generator* Sudoku

Fungsi generateSudoku merupakan fungsi untuk membangkitkan elemen-elemen awal permainan Sudoku.

```
FUNCTION generateSudoku()
{fungsi untuk membangkitkan elemen awal yang diketahui pada
permainan Sudoku dengan menggunakan algoritma Backtrack}
DEKLARASI
    values : array [0..8] of integer
    index, i : integer
    num : integer
    z, zindex : integer
    maxbacktrack, backtrack, backtracked : integer
    NElemenAwal : integer
    level : String
    fungsi sesuaiAturanSudoku(input baris, kolom, num :
    integer, output boolean)
```

Bagian ini merupakan deklarasi lokal yang akan digunakan pada fungsi generateSudoku, yaitu:

a. values merupakan variabel penampung berupa *array* berukuran 9.
b. num merupakan nilai elemen yang mungkin pada sel Sudoku.
c. NElemenAwal merupakan banyaknya elemen awal yang akan ditampilkan pada soal permainan Sudoku.
d. level merupakan tingkat kerumitan soal yang dipilih oleh pemain.
e. fungsi sesuaiAturanSudoku merupakan fungsi yang telah didefinisikan sebelumnya, untuk memeriksa apakah nilai num yang diberikan memenuhi aturan Sudoku atau tidak.

```
ALGORITMA
1.  index ← 0
2.
3.  while (index < Nbaris*NKolom) do
4.     num ← 0
5.     for i  ← 0 to 8 do
6.        values[i] ← i+1
7.        i++
8.     endfor
```

Pada baris 1-8, inisialisasi `index` dengan nilai 0. Selama `index` < 81, inisialisasi `num` dengan nilai 0. Pada baris 5-8, dilakukan pengisian nilai pada *array* `values`, sehingga diperoleh `values` = [1, 2, 3, 4, 5, 6,7, 8, 9].

```
9.     while (num = 0 & values.length > 0) do
10.       zindex ← random(0, values.length)
11.       z ← values[zindex]
12.       values[zindex] = " "
```

Pada baris 9-12, fungsi akan memilih angka secara acak antara 0 sampai dengan banyaknya angka pada `values`. Angka acak yang diperoleh akan dimasukkan ke dalam variabel penampung `zindex` dan nilai dari elemen ke-`zindex` dari `values` dimasukkan ke dalam variabel penampung `z`. Untuk menghindari terpilihnya nilai yang sama jika ternyata nilai z tidak memenuhi aturan Sudoku, maka nilai pada elemen ke-`zindex` pada `values` harus dihapus.

```
13.       {cek apakah z pada selGrid[index] memenuhi aturan
          Sudoku atau tidak}
14.       if (sesuaiAturanSudoku(z, selGrid[index]) = true)
          then
15.          num ← z
```

```
16.       else
17.          num ← 0
18.       endif
19.    endwhile
```

Pada baris 14-19, fungsi akan memeriksa apakah nilai `z` memenuhi aturan Sudoku atau tidak dengan menggunakan fungsi yang sebelumnya telah didefinisikan, yaitu fungsi `sesuaiAturanSudoku(baris, kolom, num)`. Jika fungsi `sesuaiAturanSudoku` bernilai `true` maka `num = z` sehingga proses pada fungsi akan dilanjutkan ke baris 35-36. Tetapi jika fungsi `sesuaiAturanSudoku` bernilai `false`, maka `num = 0`. Selama `num = 0` dan banyaknya angka pada `values` lebih dari 0, maka lakukan kembali algoritma pada baris 10-19.

```
20.    if num = 0 then
21.       if index > 5 then
22.          maxbacktrack ← 5
23.       else
24.          maxbacktrack ← index
25.       endif
26.
27.       backtrack ← random (1, maxbacktrack)
28.       backtracked ← 0
29.       while (backtracked < backtrack) do
30.          selGrid[index - backtracked] ← 0
31.          backtracked++
32.       endwhile
33.       index ← index - backtrack
```

Jika ternyata banyaknya angka pada `values = 0`, atau dengan kata lain tidak ada nilai `z` yang memenuhi aturan Sudoku, maka akan dilakukan *backtrack* seperti yang dilakukan pada baris 20-37. Jika `index > 5` maka variabel

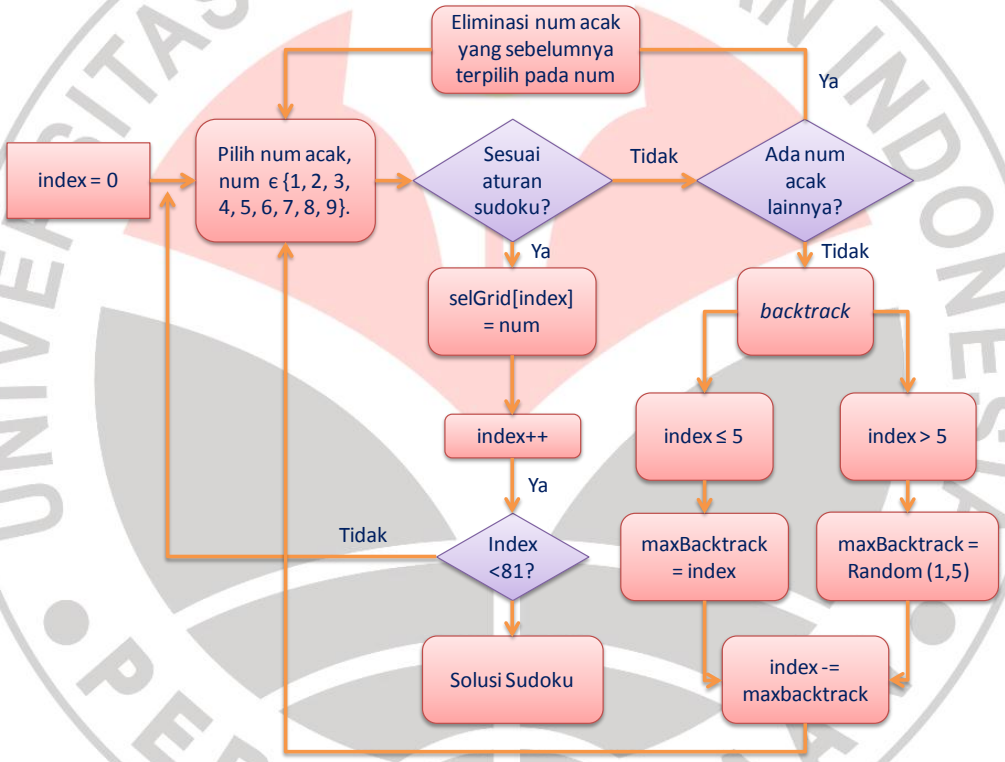
`maxbacktrack` = 5, sedangkan jika `index` ≤ 5 maka `maxbacktrack` = `index`. Kemudian, pilih secara acak antara 1 sampai dengan `maxbacktrack` untuk memperoleh banyaknya elemen pada `selGrid` yang akan di*backtrack*. Nilai-nilai pada `selGrid` ke-`index` yang di*backtrack* kemudian dikembalikan nilainya menjadi nol.

```
34.    else
35.       selGrid[index] ← num
36.       index++
37.    endif
38. endwhile
```

Proses fungsi pada baris 35-36 dilakukan apabila `num` = `z`, sehingga diperoleh nilai pada elemen `selGrid` ke-`index` adalah `num`. Kemudian proses dimulai lagi dari algoritma baris ke-3 dengan `index` = `index+1`. Proses ini akan terus berulang selama `index` < 81.

```
39. {menampilkan elemen-elemen awal Sudoku}
40.    read(level)
41.    case level
42.       'Easy': NElemenAwal = Acak(35, 40)
43.       'Medium': NElemenAwal = Acak(29, 34)
44.       'Hard': NElemenAwal = Acak(23, 28)
45.    endcase
46.
47.    while(NElemenAwal > 0) do
48.       var baris = Random(0,Nbaris-1)
49.       var kolom = Random(0,Nkolom-1)
50.       write(grid[baris][kolom])
51.       NElemenAwal--
52.    endwhile
53. endwhile
```

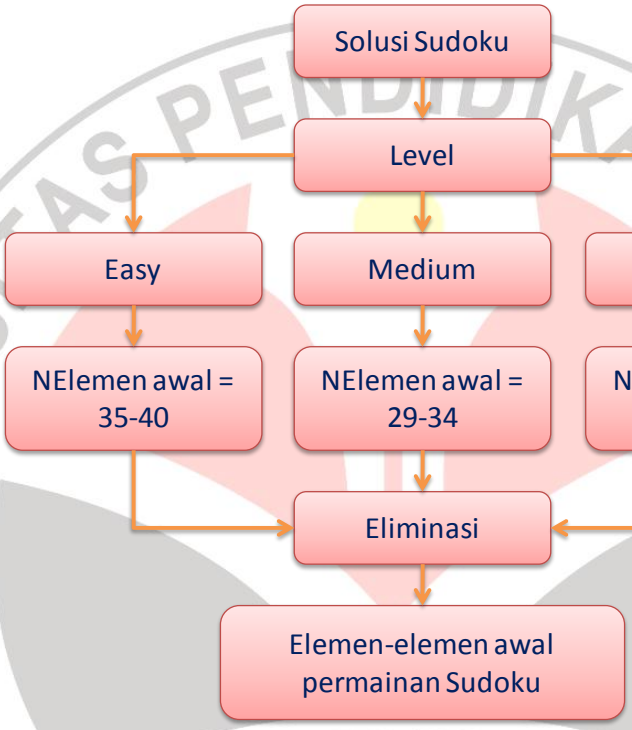
Pada baris 40-44, fungsi akan menerima input `level` yang dipilih pemain. `level` yang dipilih akan mempengaruhi banyaknya elemen awal yang ditampilkan, ketentuannya dapat dilihat pada tabel sebelumnya, yaitu tabel 3.1. Pada baris 47-52, fungsi akan mengeliminasi elemen-elemen pada solusi Sudoku hingga diperoleh sejumlah elemen awal yang harus ditampilkan sesuai dengan level yang dipilih, dengan posisi baris dan kolom bernilai acak antara 0 sampai dengan 8.



Gambar 3.3 *Flowchart* Solusi Sudoku

Proses yang terjadi pada aplikasi *SudokuBreak* ini dapat disederhanakan ke dalam bentuk *flowchart* seperti pada gambar 3.3 dan gambar 3.4. Gambar 3.3 menunjukkan proses diperolehnya solusi permainan Sudoku dengan menggunakan algoritma *Backtrack*. Solusi permainan Sudoku yang diperoleh, kemudian diproses kembali (lih. gambar 3.4) sehingga diperoleh elemen-elemen

awal yang merupakan petunjuk awal dalam permainan Sudoku. Banyaknya elemen awal yang ditampilkan tergantung dari level yang dipilih. Semakin sulit level yang dipilih, maka banyaknya elemen awal yang ditampilkan akan semakin sedikit, begitupun sebaliknya.



Gambar 3.4 *Flowchart* Elemen Awal Sudoku

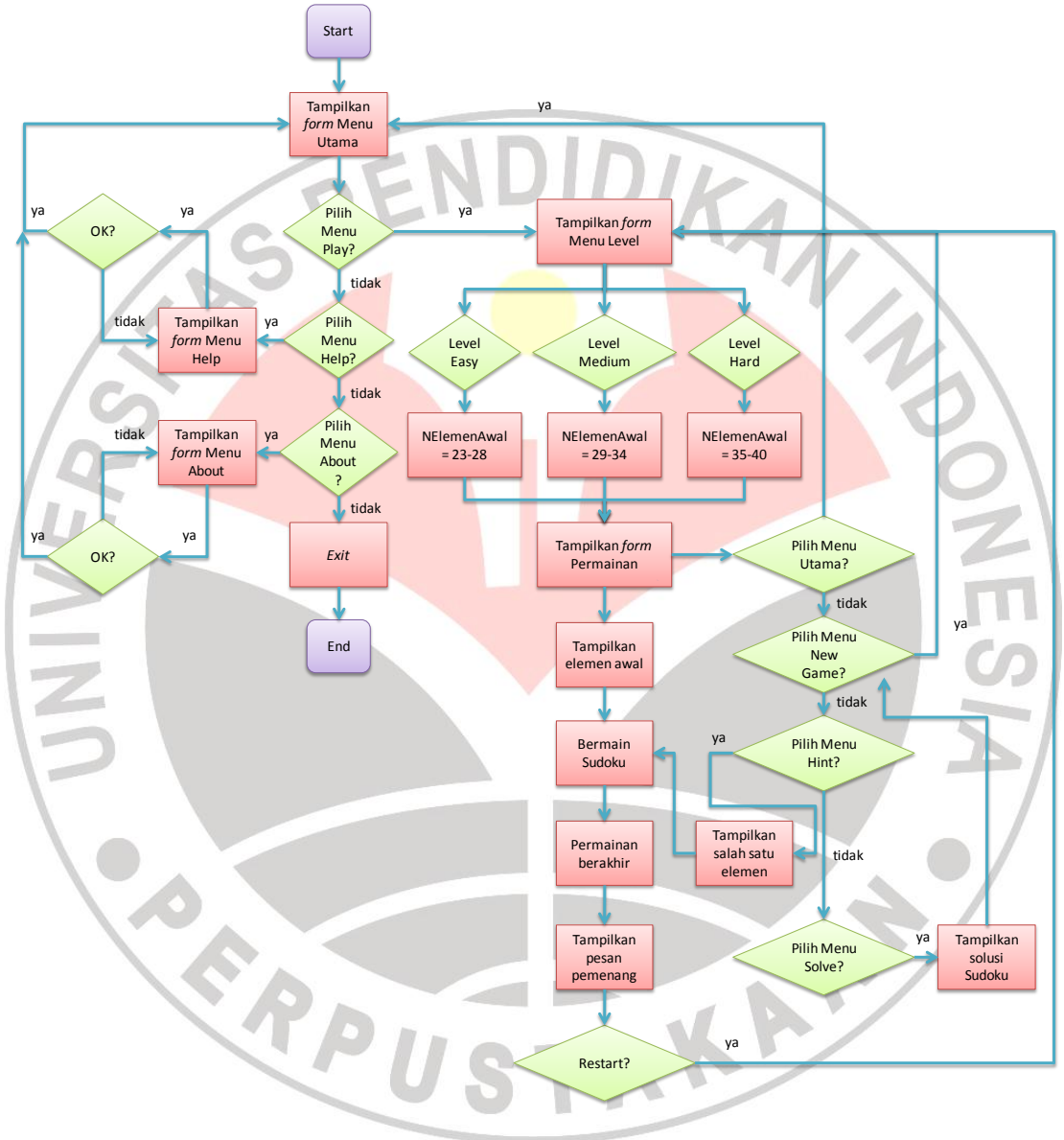
3.2 Perancangan Sistem

Perancangan sistem yang akan dibahas meliputi perancangan aplikasi permainan dan perancangan antarmuka:

3.2.1 Perancangan Aplikasi Permainan

Perancangan aplikasi permainan merupakan perancangan interaksi yang dapat dilakukan pemain terhadap aplikasi yang dibangun. Aplikasi ini akan dimulai dengan menampilkan *form* Menu Utama, yang berisi menu pilihan yaitu Menu

Permainan, Menu Help dan Menu About. Pada setiap pilihan menu yang dipilih akan ditampilkan *form-form* yang ingin ditampilkan pemain.



Gambar 3.5 *Flowchart* Aplikasi

Jika Menu Permainan dipilih, maka akan ditampilkan *form* Menu Level. Menu Level ini terdiri dari beberapa level, yaitu level *easy*, *medium*, dan *hard*. Menu Level yang dipilih pemain ini, akan mempengaruhi banyaknya elemen awal yang

akan ditampilkan pada permainan Sudoku. Semakin banyak elemen awal yang ditampilkan, maka tingkat kerumitan soal permainan Sudoku akan semakin mudah.

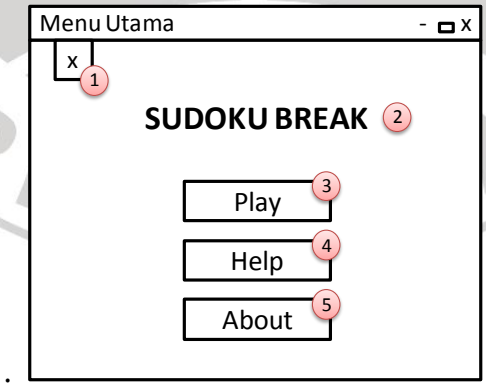
Setelah pemain memilih Menu Level yang diinginkan, maka akan ditampilkan *form* menu permainan. Aplikasi ini akan berhenti jika pemain menekan tombol *exit*. Gambar 3.5 menunjukkan proses yang terjadi pada aplikasi yang dirancang.

3.2.2 Perancangan Antarmuka

Perancangan antarmuka merupakan salah satu tahapan yang penting, karena pemain akan melakukan interaksi dengan aplikasi. Antarmuka yang dibangun haruslah sederhana dan menarik, sehingga mudah dipahami oleh pemain. Antarmuka pada aplikasi permainan Sudoku Break ini terdiri dari beberapa *form* yaitu *form* menu utama, *form* menu permainan, *form* menu help, *form* menu about, dan *form* menu level.

1. *Form* Menu Utama

   Form menu utama merupakan *form* yang ditampilkan pertama kali pada saat aplikasi dijalankan. *Form* ini terdiri dari beberapa menu yang dapat dipilih oleh pemain, yaitu menu play, help dan about. *Form* ini dapat dilihat pada gambar 3.6



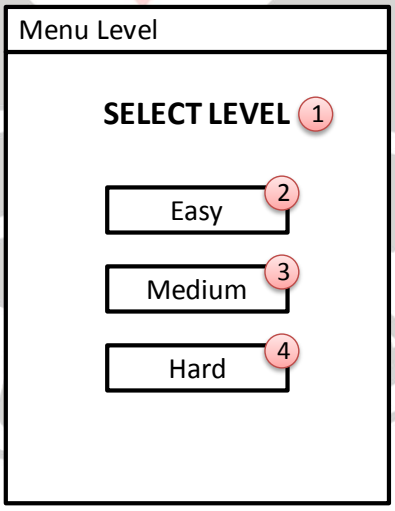
Gambar 3.6 *Form* Menu Utama

Keterangan (Gambar 3.6):

1. Tombol ini merupakan tombol menu yang berfungsi untuk keluar dari aplikasi.
2. SUDOKU BREAK merupakan judul dari aplikasi.
3. Tombol Play merupakan tombol menu yang berfungsi untuk menampilkan *form* Menu Permainan.
4. Tombol Help merupakan tombol menu yang berfungsi untuk menampilkan *form* Menu Help yang berisi tentang panduan cara bermain Sudoku.
5. Tombol About merupakan tombol menu yang berfungsi untuk menampilkan *form* Menu About yang berisi tentang *author* aplikasi.

2. *Form* Menu Level

Form Menu Level merupakan *form* yang berisi beberapa pilihan Level yang diberikan kepada pemain. Apapun tombol level yang dipilih, nantinya akan sama-sama menampilkan *form* menu Permainan.



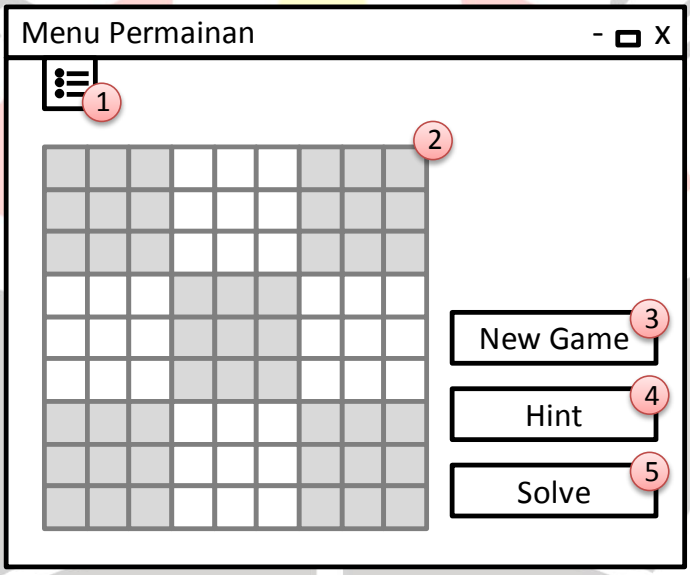
Gambar 3.7 *Form* Menu Level

Keterangan (Gambar 3.7):

1. SELECT LEVEL merupakan judul dari *form* menu Level.
2. Tombol Easy merupakan tombol level yang paling mudah.
3. Tombol Medium merupakan tombol level sedang.
4. Tombol Hard merupakan tombol level yang paling sulit.

3. *Form* Menu Permainan

   Form menu permainan merupakan *form* yang digunakan dalam bermain permainan Sudoku. *Form* ini dapat dilihat pada gambar 3.8.



Gambar 3.8 *Form* Menu Permainan

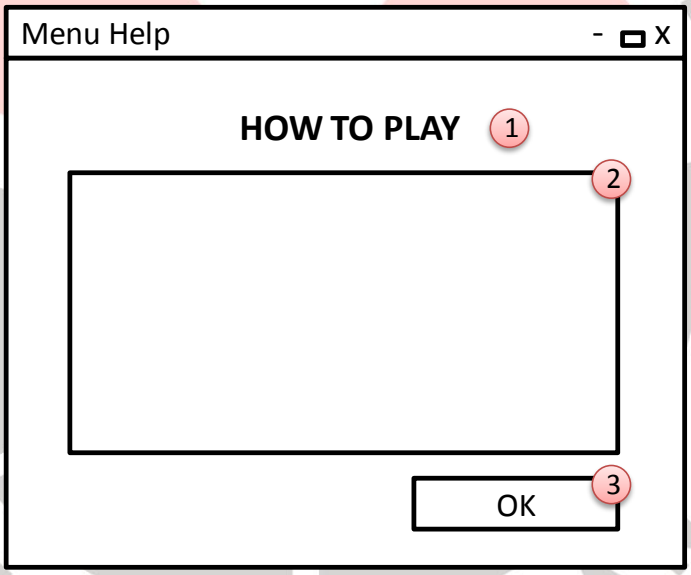
Keterangan (Gambar 3.8):

1. Tombol ini merupakan tombol yang berfungsi untuk menampilkan *form* menu utama.
2. Papan permainan Sudoku berukuran 9x9.
3. Tombol New Game merupakan tombol yang berfungsi untuk menampilkan soal permainan yang berbeda.

4. Tombol Hint merupakan tombol yang berfungsi untuk memberikan satu angka bantuan kepada pemain untuk dapat menyelesaikan permainan Sudoku.
5. Tombol Solve merupakan tombol yang berfungsi untuk menampilkan seluruh solusi permainan yang *valid* kepada pemain.

4. *Form* Menu Help

   Form Menu Help merupakan *form* yang berisikan informasi mengenai aturan dan cara bermain Sudoku. *Form* ini dapat dilihat pada gambar 3.9.



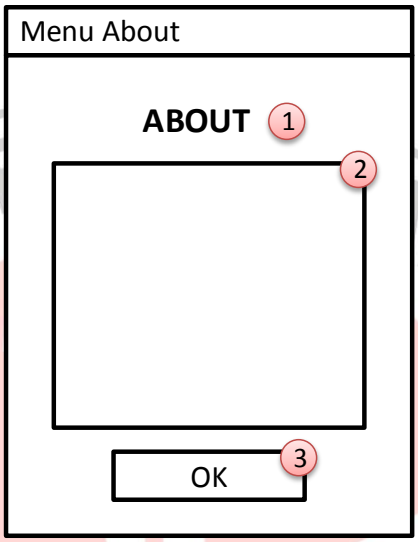
Gambar 3.9 *Form* Menu Help

Keterangan (Gambar 3.9):

1. HOW TO PLAY merupakan judul dari *form* menu Help.
2. *Textbox* ini berisi informasi mengenai cara bermain SUDOKU BREAK.
3. Tombol OK berfungsi untuk menampilkan *form* menu utama.

5. *Form* Menu About

Form Menu About merupakan *form* yang berisikan informasi mengenai *author* aplikasi. *Form* ini dapat dilihat pada gambar 3.10.



Gambar 3.10 *Form* Menu About

Keterangan (Gambar 3.10):

1. ABOUT merupakan judul dari *form* menu About.
2. *Textbox* ini berisi informasi mengenai author aplikasi SUDOKU BREAK.
3. Tombol OK berfungsi untuk menampilkan *form* menu utama.