BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas tahapan proses implementasi sistem berdasarkan hasil analisa kebutuhan dan perancangan perangkat lunak yang telah dibuat. Pembahasan terdiri dari penjelasan spesifikasi sistem, batasan-batasan implementasi, implementasi algoritma, implementasi antarmuka, dan implementasi *database*.

4.1. Spesifikasi Sistem

Sistem penggalian (penemuan) pola dikembangkan pada lingkungan implementasi yang terdiri dari perangkat keras dan perangkat lunak

4.1.1. Implementasi Perangkat Keras

Spesifikasi perangkat keras yang digunakan dapat dilihat pada tabel 4.1

Tabel 4.1 Tabel pesifikasi perangkat keras

Nama Komponen	**Spesifikasi**
Processor	2.4 GHz Intel Core 2 Duo
Memori (RAM)	2 GB 1067 MHz DDR3
Harddisk	250 GM
Graphic Card	Nvidia GeForce 320M

4.1.2. Implementasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan dapat dilihat pada tabel 4.2

Tabel 4.2 Tabel pesifikasi perangkat lunak

Nama	**Spesifikasi**
Sistem Operasi	Microsoft Windows 7 Ultimate
Bahasa Pemrograman	PHP
Tools pemrograman	Aptana Studio 3
Database Management System	phpMyAdmin

4.2. Batasan - batasan Implementasi

Batasan – batasan dalam mengimplementasikan sistem penggalian pola pada *database* adalah sebagai berikut:

1. Sistem penggalian (penemuan) pola pada *database* ini dirancang dan dijalankan dengan menggunakan *web application* (aplikasi web).

2. Data yang digunakan untuk penelitian ini diperoleh dari data induk mahasiswa yang terdapat pada *database* Universitas Brawijaya. Pada penemuan pola penerimaan, batasan atribut adalah angkatan, seleksi, program studi, dan asal propinsi. Sedangkan pada penemuan pola kelulusan, batasan atribut adalah angkatan, seleksi, program studi, dan kelulusan.
3. Teknik data mining yang digunakan dalam menyelesaikan masalah dengan algoritma penemuan pola yang digunakan adalah Algoritma *FP-Growth*.
4. Output yang dihasilkan oleh sistem adalah pola dari penerimaan dan kelulusan mahasiswa Fakultas Ekonomi dan Bisnis Universitas Brawijaya.

4.3. Implementasi Program

Sistem penemuan pola ini terdiri dari beberapa proses yaitu proses *preprocessing* dan proses dari algoritma *FP-Growth*.

4.3.1. Implementasi *Preprocessing*

Proses *preprocessing* yang dilakukan terdiri dari dua tahap yaitu pembersihan data dan transformasi data. Masukan dari proses ini adalah file data induk mahasiswa dengan format .csv. Hasil dari proses *preprocessing* ini adalah id dari masing-masing atribut yang digunakan.

4.3.1.1 Pembersihan Data

Pada proses ini setiap baris pada tabel data induk mahasiswa akan dibersihkan apabila terdapat data-data yang kurang valid. Pembersihan ini berdasarkan isi *field* yang tidak sama dengan komposisi data yang sebenarnya. Data yang tidak sama akan dihapus atau diubah menjadi data yang sesuai.

```
/*------ potongan source code untuk menghapus baris yang tidak diperlukan
dan perbaikan data -------*/

...
$cleaning_mhs = mysql_query("DELETE FROM mhs WHERE KELAMIN!='L' AND
KELAMIN!='P'");

$update_prodi = mysql_query ("UPDATE t_kelulusan SET PRODI='Manajemen'
WHERE PRODI='Management'");
$update_seleksiSAP = mysql_query("UPDATE t_kelulusan SET SELEKSI='SAP'
WHERE SELEKSI='Alih program' OR SELEKSI='Alih Program - Kerjasama'");
$update_seleksiPSB = mysql_query("UPDATE t_kelulusan SET SELEKSI='PSB'
WHERE SELEKSI='Jalur Prestasi Akad' OR SELEKSI='Jalur prestasi - P.I' OR
SELEKSI='Jalur Prestasi Non Akad'");
$update_seleksiSPMK = mysql_query("UPDATE t_kelulusan SET SELEKSI='SPMK'
WHERE SELEKSI='Minat dan kemampuan' OR SELEKSI='SPMK - Prog.
Internasional'");
...
```

Source Code 4.1 Potongan source code proses pembersihan data

Pada *source code* 4.1 merupakan potongan *source code* untuk pembersihan data. Beberapa data yang tidak sesuai akan dihapus. Saat melakukan proses *import file*, *file* yang dimasukan ke *database* awalnya berupa *file* dengan format .csv. Struktur data pada file .csv tersebut tidak langsung sesuai dengan kebutuhan sistem. Terdapat beberapa baris data yang harus dihapus, baris yang dihapus adalah baris pertama dari *file* yang dimasukan. Selain itu, pembersihan data tidak hanya menghapus data yang tidak sesuai atau tidak diperlukan, tapi juga dilakukan penyesuaian data sesuatu dengan kebutuhan. Salah satu penyesuaian data yang dilakukan adalah merubah program studi "Management" menjadi "Manajemen". Hal ini dilakukan untuk penyamaan data, agar lebih mudah saat dilakukan proses transformasi.

4.3.1.2 Transformasi Data

Pada proses tranformasi data ini semua baris pada tabel t_penerimaan dan t_kelulusan akan diperiksa dan baris yang diperlukan untuk proses sistem isinya akan diubah menjadi kode-kode tertentu sesuai dengan kebutuhan sistem.

```
/*------ potongan source code untuk merubah atribut menjadi kode-kode ---
----*/

...
$transformasi_tp_angkatan = mysql_query("UPDATE t_penerimaan tp,
kat_angkatan ka SET tp.id_angkatan=ka.id_angkatan WHERE
tp.ANGKATAN=ka.nama_angkatan");
$transformasi_tp_seleksi = mysql_query("UPDATE t_penerimaan tp,
kat_seleksi ks SET tp.id_seleksi=ks.id_seleksi WHERE
tp.SELEKSI=ks.nama_seleksi");
...
$transformasi_tk_seleksi = mysql_query("UPDATE t_kelulusan tk,
kat_seleksi ks SET tk.id_seleksi=ks.id_seleksi WHERE
```

```
14  tk.SELEKSI=ks.nama_seleksi");
16  $transformasi_tk_prodi = mysql_query("UPDATE t_kelulusan tk, kat_prodi kp
17  SET tk.id_prodi=kp.id_prodi WHERE tk.PRODI=kp.nama_prodi");
    ...
```

Source Code 4.2 Potongan source code proses transformasi atribut pada *database*

Pada *source code* 4.2 merupakan potongan *source code* dalam proses transformasi data. Pengubahan isi dari setiap *field* ini berupa kode-kode yang sudah ditetapkan. Seperti untuk *field* 'angkatan' akan dirubah menjadi kode yang sudah ditetapkan untuk tiap angkatan, yaitu angakatan 2008 mejadi B1, hingga angakatan 2012 menjadi B5. Untuk *field* yang lain aka diubah sesuai dengan kodenya masing-masing.

4.3.2. Implementasi Algoritma *FP-Growth*

Sebelum masuk ke algoritma *FP-Growth* itu sendiri, terlebih dahulu dilakukan pembuatan *FP-Tree* untuk menghasilkan lintasan-lintasan yang akan menjadi dasar dalam pembuatan pola dengan algoritma *FP-Growth*. Dalam program digunakan beberapa fungsi.

Tabel 4.3 Tabel daftar fungsi dalam aplikasi

No.	Fungsi	Keterangan
1.	function cariNode()	Fungsi untuk mencari item yang akan dimasukan ke *tree* terhadap node-node yang sudah terdapat di *tree.*
2.	function cekNode()	Fungsi untuk memeriksa item dengan node yang ada pada *tree*.
3.	function Node()	Fungsi untuk memeriksa node dan posisi dari node yang sama denga item.
4.	function NodeBaru()	Fungsi untuk membuat node baru.
5.	function TransformasiParent()	Fungsi untuk mengubah *parent* mejadi id sesuai dengan id node dari *parent*
6.	function GetParent()	Fungsi untuk mendapatkan *parent* dari node sesuai dengan lintasan pada *tree*
7.	function GetChild()	Fungsi untuk mendapatkan *child* dari node sesuai dengan lintasan pada *tree*
8.	function TransformasiChild()	Fungsi untuk mengubah *child* mejadi id sesuai dengan id node dari *child*
9.	function CountPlus()	Fungsi untuk menambah jumlah (coutn) pada node
10.	function array_empty_remover()	Fungsi untuk membersihan child yang sama pada node

4.3.2.1. Pembuatan *FP-Tree*

Pada proses pembuatan *tree* ini, strutktur tiap node dalam *tree* menggunakan array. Tiap node akan mempunyai *parent, child, id, count, level*. *Parent* merupakan node yang menjadi orang tua (*parent*) dari node yang ditunjuk. *Child* merupaka anak dari node yang ditunjuk. *Id* merupakan kode untuk tiap node. *Count* merupakan jumlah dari seberapa sering node tersebut dilewati atau muncul. Sedangkan, *level* merupakan kedalaman dari node. Potongan *source code* untuk pebentukan *tree* ditampilkan pada *source code* 4.3.

```
/*------ potongan source code pembentukan tree -------*/
...
for ($Trans = 0; $Trans < $output_count_TID_p; $Trans++){
   for ($T = 0; $T < count($items[$Trans]) ; $T++){
    $itemNow = current($items[$Trans]); $pNode = $itemNow['item'];
    $itemNext = next($items[$Trans]); $PNode = $itemNext['item'];
    $parentItem = $items[$Trans][$T-1]['item'];
    if ($pNode == CariNode($Tree, $pNode, $T)) {
     if (Node($Trans, $T, $Tree, $pNode, $parentItem, $items,
       $count,$level,                          $itemID)==TRUE){
       $TerimaCekNode = cekNode($Trans, $T, $Tree, $pNode, $parentItem,
                                $items, $count, $level, $itemID);
       if ($TerimaCekNode['node'] == 'beda'){
         array_push($Tree, NodeBaru($pNode, $level, $itemID, $Trans, $T,
                    $Tree, $PNode, $parentItem, $items, $count));
         $itemID++;
       }elseif ($TerimaCekNode[0] != 'beda'){
         for ($cekNode=0; $cekNode < count($Tree); $cekNode++){
           if ($Tree[$cekNode]['id'] == $TerimaCekNode[0]['id'] &&
               $Tree[$cekNode]['node'] == $pNode) {
             $Tree[$cekNode]['count']+=1;
             array_push($Tree[$cekNode]['child'], GetChild($Trans, $PNode,
                        $itemID, $Tree, $parentItem, $T, $pNode));
           }
       }
     }
    }elseif (Node($Trans, $T, $Tree, $pNode, $parentItem, $items,
             $count, $level, $itemID) == FALSE){
      array_push($Tree, NodeBaru($pNode, $level, $itemID, $Trans, $T,
                 $Tree, $PNode, $parentItem, $items, $count));
      $itemID++;
    }
   }else{
    array_push($Tree, NodeBaru($pNode, $level, $itemID, $Trans, $T,
               $Tree, $PNode, $parentItem, $items, $count));
    $itemID++;
   }
   cekNode($Trans, $T, $Tree, $pNode, $parentItem, $items, $count,
           $level, $itemID);
  }
  $TransformasiChild = TransformasiChild($Tree, $Trans, $items, $T);
  for ($i=0; $i < count($Tree); $i++){
     for ($j=0; $j < count($TransformasiChild); $j++){
        if ($Tree[$i]['node'] == $TransformasiChild[$j]['node'] &&
           $Tree[$i]['id'] == $TransformasiChild[$j]['id'] &&
           $Tree[$i]['parent'] == $TransformasiChild[$j]['parent'])
        }
     }
   }
}
...
```

Source Code 4.3 Potongan source code proses pembuatan *tree*

Source code 4.3 merupakan potongan *source code* dalam pembuatan *tree.* Pemasukan item kedalam *tree* dilakukan pada semua transaksi dan semua item dalam transaksi. Setiap node mempunyai satu *parent* dan bisa mempunyai lebih dari satu *child*. Setiap node mempunyai level, level merupakan kedalam dari node, semakin dalam node semakin besar pula levelnya. *Item* yang akan dimasukan ke *tree* akan diperiksa keberadaannya dalam *tree* dengan fungsi CariNode, *source code* 4.4.

```
1  /*------ potongan source code pembentukan tree -------*/
2  ...
3  function CariNode($Tree, $pNode, $T){
4          $allNode = array();
5          for ($i=0; $i < count($Tree) ; $i++) {
6                  if ($pNode == $Tree[$i]['node']) {
7                          $allNode[]=$Tree[$i];
8                  }
9          }
10         for ($i=0; $i < count($allNode) ; $i++) {
11                 if ($T+1 == $allNode[$i]['level'] ) {
12                   return $allNode[$i]['node'];
13                 }
14         }
15 }
16 ...
```

Source Code 4.4 Fungsi CariNode

Source code 4.4 merupakan fungsi CariNode, pemeriksaan item menggunakan parameter nama node dan level dari node. Nama node dibandingkan dengan nama item dan level dibandingkan dengan iterasi pada keberapa item tersebut dimasukan.

```
1  /*------ potongan source code pembentukan tree -------*/
2  ...
3  function Node($Trans, $T, $Tree, $pNode, $parentItem, $items, $count,
4  $level,
5                  $itemID){
6     $cekNode = array();
7     $cekTemp = array();
8     $itemTemp = array();
9     unset($cekNode);
10    unset($cekTemp);
11    unset($itemTemp);
12    if ($T == 0) {
13          for ($a=0; $a < count($Tree); $a++) {
14           if ($pNode == $Tree[$a]['node'] && $T+1 == $Tree[$a]['level']) {
15            return TRUE;
16           }elseif ($pNode == $Tree[$a]['node'] && $T+1 !=
17 $Tree[$a]['level']){
18             return FALSE;
19                 }
20         }
21    }elseif ($T>0){
22     for ($i=0; $i < count($Tree); $i++) {
23       if ($pNode == $Tree[$i]['node'] && $T+1 == $Tree[$i]['level']) {
24           $cekNode[] = $Tree[$i];
25         }
```

```
26      }
27      for ($cN=0; $cN < count($cekNode); $cN++) {
28       $nodeFirstTemp = $cekNode[$cN]['node'];
29          $cekTemp[$cN][0] = $nodeFirstTemp;
30          for ($tree=0; $tree < count($Tree); $tree++) {
31        if ($cekNode[$cN]['parent'] == $Tree[$tree]['id']) {
32             $nodeTemp = $Tree[$tree];
33            $nodeTempNode = $nodeTemp['node'];
34            array_push($cekTemp[$cN], $nodeTempNode);
35           for ($k=2; $k < $cekNode[$cN]['level']; $k++) {
36            for ($l=0; $l < count($Tree); $l++) {
37             if ($nodeTemp['parent'] == $Tree[$l]['id']) {
38              $nodeTemp = $Tree[$l];
39                  $nodeTempNode = $nodeTemp['node'];
40                  array_push($cekTemp[$cN], $nodeTempNode);
41                break;
42                }
43                }
44              }
45             }
46            }
47      }
48      for ($n=$T; $n >=0 ; $n--) {
49         $itemTemp[] = $items[$Trans][$n]['item'];
50      }
51      if (in_array($itemTemp, $cekTemp)) {
52          return TRUE;
53      }else {
54          return FALSE;
55      }
    }
}
...
```

Source Code 4.5 Fungsi Node

Source code 4.5 merupakan fungsi Node digunakan untuk memeriksa antara item dan node pada *tree* dengan perbandingan parent antara node dengan item. Fungsi ini menjadikan pemeriksaan node lebih spesifik. Nilai balik dari fungsi ini berupa nilai 'TRUE' atau 'FALSE'.

```
1   /*------ potongan source code pembentukan tree -------*/
2   ...
3   function cekNode($Trans, $T, $Tree, $pNode, $parentItem, $items, $count,
4                    $level, $itemID){
5     $gaSama = array(
6                          'node' => 'beda');
7     $arrayTemp = array();
8     $itemTemp = array();
9     $nodeFoundTemp = array();
10    $nodeFound = array();
11    $cekNode = array();
12    $cekParentItem = array();
13    $parentItem2 = $items[$Trans][$T-2]['item'];
14    $parentItem3 = $items[$Trans][$T-3]['item'];
15    unset($cekNode);
16    unset($nodeFound);
17    unset($cekParentNode2);
18    unset($cekParentNode3);
19    unset($cekParentNode4);
20    if ($T==0) {
21      for ($i=0; $i < count($Tree); $i++) {
22        if ($pNode == $Tree[$i]['node'] && $T+1 == $Tree[$i]['level']) {
23            $cekNode[] = $Tree[$i];
24        }
25      }
```

```
26      if ($cekNode[0]['node'] == $pNode) {
27        $nodeFound[] = $cekNode[0];
28          return $nodeFound;
29      }
30    }
31     else{
32        for ($i=0; $i < count($Tree); $i++) {
33         if ($pNode == $Tree[$i]['node'] && $T+1 == $Tree[$i]['level']) {
34             $cekNode[] = $Tree[$i];
35         }
36        }
37        for ($j=0; $j < count($cekNode); $j++) {
38           $nodeFound[] = $cekNode[$j];
39           for ($i=0; $i < count($Tree); $i++) {
40           if ($cekNode[$j]['parent'] == $Tree[$i]['id']) {
41                   $arrayTemp = $Tree[$i];
42                   array_push($nodeFound, $arrayTemp);
43                   for ($k=2; $k < $cekNode[$j]['level']; $k++) {
44                    for ($l=0; $l < count($Tree); $l++) {
45                   if ($arrayTemp['parent'] == $Tree[$l]['id']) {
46                       $arrayTemp = $Tree[$l];
47                  array_push($nodeFound, $arrayTemp);
48                    break;
49                       }
50                     }
51                   }
52               }
53           }
54        for ($m=0; $m < count($nodeFound); $m++) {
55           $nodeFoundTemp[] = $nodeFound[$m]['node'];
56        }
57        for ($n=$T; $n >=0 ; $n--) {
58           $itemTemp[] = $items[$Trans][$n]['item'];
59        }
60        if (count($cekNode) == 1 && $nodeFoundTemp != $itemTemp) {
61           return $gaSama;
62        }
63        if ($nodeFoundTemp == $itemTemp) {
64           return $nodeFound;
65        }
66        unset($nodeFound);
67        unset($nodeFoundTemp);
68        unset($itemTemp);
69      }
70    }
71  }
...
```

Source Code 4.6 Fungsi cekNode

Source code 4.6 merupakan fungsi cekNode yang berfungsi untuk pemeriksaan node dengan nilai balik berupa posisi dan node. Apabila ada node yang sama persis dengan item yang akan dimasukan, maka nilai kembalinya berupa node tersebut. Apanila tidak ada node yang sama persis dengan item yang akan dimasukan maka nilai kembalian ‘beda’.

```
/*------ potongan source code pembentukan tree -------*/
...
function NodeBaru($pNode, $level, $itemID, $Trans, $T, $Tree, $PNode,
                  $parentItem, $items, $count){
  $level = $T+1;
  $node = array(
                'node' => $pNode,
                'id' => $itemID,
                'parent' => GetParent($Trans, $T, $Tree, $pNode,
$parentItem,
                                      $items),
                'child' => array (GetChild($Trans, $PNode, $itemID, $Tree,
                                           $parentItem, $T, $pNode)),
                'count' => $count,
                'level' => $level,
                );
  return $node;
}
...
```

Source Code 4.7 Fungsi NodeBaru

Source code 4.7 merupakan fungsi NodeBaru untuk membuat struktur node dalam array. Node yang dibuat ini akan dimasukkan kedalam *tree* sebagai node baru.

```
/*------ potongan source code pembentukan tree -------*/
...
function GetParent($Trans, $T, $Tree, $pNode, $parentItem, $items){
  $allParent = array();
  $ParentOfParent = array ();
  $ParentOFParentParent = array();
    if ($Trans == 0) {
    if ($T == 0) {
      return $parentID = 0;
         }else{
      for ($i=0; $i <count($Tree) ; $i++) {
          if ($Tree[$i]['node'] == $parentItem) {
               return $parentID = $Tree[$i]['id'];
          }
         }
    }
  }else{
        if ($T == 0) {
       return $parentID = 0;
    }else{
         for ($i=0; $i <count($Tree) ; $i++) {
          if ($Tree[$i]['node'] == $parentItem) {
               $allParent[] = $Tree[$i];
          }
          if ($items[$Trans][$T-2]['item'] == null){
               $ParentOfParent[] = $Tree[0];
          }elseif ($Tree[$i]['node'] == $items[$Trans][$T-2]['item']) {
               $ParentOfParent[] = $Tree[$i];
          }
          if ($items[$Trans][$T-3]['item'] == null){
               $ParentOFParentParent[] = '-'; // root
            }elseif ($Tree[$i]['node'] == $items[$Trans][$T-3]['item']) {
               $ParentOFParentParent[] = $Tree[$i];                    }
       }
     for ($i=0; $i < count($allParent) ; $i++) {
      for ($j=0; $j < count($ParentOfParent); $j++) {
         for ($k=0; $k < count($ParentOFParentParent); $k++) {
```

```
38            if ($ParentOfParent[$j]['id'] == $allParent[$i]['parent'] &&
39              $ParentOFParentParent[$k]['id'] ==
40    $ParentOfParent[$j]['parent'] )
41              {
42                          return $parentID = $allParent[$i]['id'];
43              }
44            }
45        }
46      }
47    }
      }
      ...
```

Source Code 4.8 Fungsi GetParent

Source code 4.8 merupakan fungsi GetParent berfungsi untuk mencari *parent* dari node. Pencarian *parent* dilakukan dengan memeriksa *tree* dan mencocokan *parent* dari item dengan node *tree* yang sudah terbentuk.

```
1   /*------ potongan source code pembentukan tree -------*/
2   ...
3   function TranformasiParent ($Tree, $parentItem, $T){
4     for ($i=0; $i < count($Tree); $i++) {
5     if ($Tree[$i]['node'] == $parentItem && $Tree[$i]['level'] == $T) {
6       return $Tree[$i]['id'];
7     }
8     }
9   }
10  ...
```

Source Code 4.9 Fungsi TransformasiParent

Source code 4.9 merupakan fungsi TransformasiParent berfungsi untuk merubah *parent* dari node yang masih berbentuk nama node menjadi bentuk *id*. Pengubahan ini dilakukan dengan memeriksa *tree* dan mencari node yang memiliki nama dan level yang sama dengan *parent*.

```
1   /*------ potongan source code pembentukan tree -------*/
2   ...
3   function GetChild($Trans, $PNode, $itemID, $Tree, $parentItem, $T,
4   $pNode){
5     $allChild = array();
6     $allNode = array();
7     if ($Trans == 0) {
8      if ($PNode == null) {
9       return $childID = null;
10      }else{
11         return $childID = $itemID+1;
12      }
13    }
14    else{
15      if ($PNode == null) {
16          return $childID = null;
17      }else{
18          for ($i=0; $i < count($Tree); $i++) {
19            if ($Tree[$i]['node'] == $pNode) {
20                 $allNode[] = $Tree[$i];
21            }
22          }
23       for ($i=0; $i < count($allNode['child']); $i++) {
24          $allChild[] = $allNode['child'][$i];
25        }
```

```
      for ($i=0; $i < count($Tree); $i++) {
          for ($j=0; $j < count($allChild); $j++) {
              if ($Tree[$i]['id'] == $allChild[$j]) {
             $ChildFound = $Tree[$i]['node'];
              }
          }
        }
      if ($ChildFound != $PNode) {
          return $PNode;
      }
    }
  }
}
...
```

Source Code 4.10 Fungsi GetChild

Source code 4.10 merupakan fungsi GetChild berfungsi untuk mencari *child* dari node. Bentuk dari *child* yang didapat berupa nama node .

```
/*------ potongan source code pembentukan tree -------*/
...
function TransformasiChild($Tree, $Trans, $items, $T){
  $nodeTemp = array();
  $nodeTemp2 = array();
  $nodeChildTemp = array();
  $childTemp = array();
  $TransformasiChild = array();
  $idParent = array();
  $parentTemp = array();
  $pNodeTemp = array();
  for ($T = 0; $T < count($items[$Trans]) ; $T++){
       $itemNow = current($items[$Trans]); $pNode = $itemNow['item'];
       $itemNext = next($items[$Trans]); $PNode = $itemNext['item'];
       $parentItem = $items[$Trans][$T-1]['item'];
     unset($nodeTemp);
       unset($nodeChildTemp);
       unset($nodeTemp2);
       unset($childTemp);
       unset($idParent);
       unset($parentTemp);
       unset($pNodeTemp);
     if ($Trans == 0) {
     }elseif ($Trans > 0){
         for ($i=0; $i < count($Tree); $i++) {
           if ($Tree[$i]['node'] == $pNode) {
               $nodeTemp[] = $Tree[$i];
           }
           if ($Tree[$i]['node'] == $PNode) {
               $nodeChildTemp[] = $Tree[$i];
           }
         }
      for ($i=0; $i < count($nodeTemp) ; $i++) {
         $idParent[]=$nodeTemp[$i]['parent'];
        }
        for ($i=0; $i < count($Tree); $i++) {
         for ($j=0; $j < count($idParent); $j++) {
           if ($Tree[$i]['id'] == $idParent[$j]) {
               $parentTemp[] = $Tree[$i];
           }
         }
      }
      for ($i=0; $i < count($nodeTemp); $i++) {
         for ($j=0; $j < count($parentTemp); $j++) {
           if ($parentItem == null) {
                 $parentItem = 'root';
                 if ($nodeTemp[$i]['parent'] ==
$parentTemp[$j]['id'] &&
                       $parentItem == $parentTemp[$j]['node']) {
```

```
                                $nodeTemp2[] = $nodeTemp[$i];
                            }
                        }
                elseif ($parentItem != null) {
                        if ($nodeTemp[$i]['parent'] == $parentTemp[$j]['id'] &&
                            $parentItem == $parentTemp[$j]['node']) {
                            $nodeTemp2[] = $nodeTemp[$i];
                        }
                    }
                }
            }
        for ($i=0; $i < count($nodeTemp2); $i++) {
            for ($j=0; $j < count($nodeChildTemp); $j++) {
                if ($nodeTemp2[$i]['id'] < $nodeChildTemp[$j]['id'] &&
                    $nodeTemp2[$i]['id'] == $nodeChildTemp[$j]['parent']) {
                $childTemp[] = $nodeChildTemp[$j];
                }
            }
        }
        for ($i=0; $i < count($nodeTemp2); $i++) {
            for ($j=0; $j < count($childTemp); $j++) {
              if ($nodeTemp2[$i]['id'] == $childTemp[$j]['parent']) {
                $pNodeTemp[] = $nodeTemp2[$i];
              }
            }
        }
        for ($i=0; $i < count($Tree); $i++) {
            if ($Tree[$i]['id'] == $pNodeTemp[0]['id']) {
              for ($j=0; $j < count($Tree[$i]['child']); $j++) {
                if ($Tree[$i]['child'][$j] == $childTemp[0]['node']) {
                  $Tree[$i]['child'][$j] = $childTemp[0]['id'];
                $TransformasiChild[] = $Tree[$i];
                }
              }
            }
          }
      }
    }
    return $TransformasiChild;
}
...
```

Source Code 4.11 Fungsi TransformasiChild

Source code 4.11 merupakan fungsi TransformasiChild berfungsi untuk merubah *child* dari node yang masih berbentuk nama node menjadi *id* dari node *child* tersebut.

```
/*------ potongan source code pembentukan tree -------*/
...
function CountPlus($Tree, $pNode, $Trans, $T, $parentItem, $items, $count,
                   $level, $itemID){
  if (Node($Trans, $T, $Tree, $pNode, $parentItem, $items, $count, $level,
      $itemID) == TRUE) {
    $cekNode = array();
    for ($i=0; $i < count($Tree); $i++) {
      if ($pNode == $Tree[$i]['node']) {
        $cekNode[] = $Tree[$i];
      }
    }
    for ($n=0; $n < count($Tree); $n++) {
       for ($o=0; $o < count($cekNode); $o++) {
         if ($pNode == $cekNode[$o]['node'] && $cekNode[$o] == $Tree[$n])
{
                return $Tree[$n]['count'] += 1;
            }
      }
```

```
    }
  }
}
...
```

Source Code 4.12 Fungsi CountPlus

Source code 4.12 merupakan fungsi CountPlus berfungsi untuk penambahan *count* pada node. Setiap node yang dilewati oleh node yang sama, maka *count* akan ditambah 1.

```
/*------ potongan source code pembentukan tree -------*/
...
function array_empty_remover($array, $remove_null_number = true){
        $new_array = array();
        $null_exceptions = array();
        foreach ($array as $key => $value) {
                $value = trim($value);
                if ($remove_null_number) {
                        $null_exceptions[] ='0';
                }
                if (!in_array($value, $null_exceptions) && $value != "") {
                        $new_array[] = $value;
                }
        }
        return $new_array;
}
...
```

Source Code 4.13 Fungsi array_empty_remover

Source code 4.13 merupakan fungsi array_empty_remover berfungsi untuk menghapus *child* dari setiap node masih terdapat duplikasi *child*.

4.3.2.2. Algoritma *FP-Growth*

Pada tahap pengimplementasian algoritma *FP-Growth* ini terdapat beberapa tahap yaitu, pembangkitan *conditional pattern base,* pembangkitan *conditional FP-Tree,* dan pencarian *frequent itemset.*

```
/*------ potongan source code pembankitan conditional pattern base -------
*/
...
$nodeLink = array();
for ($polaItem=0; $polaItem < count($frequentList); $polaItem++) {
  for ($tree=0; $tree < count($Tree); $tree++) {
    if ($frequentList[$polaItem]['item'] == $Tree[$tree]['node']) {
        $nodeLinkItem[] = $Tree[$tree];
    }
  }
  array_push($nodeLink, $nodeLinkItem);
  unset($nodeLinkItem);
}
...
```

Source Code 4.14 Potongan *source code* pembangkitan *conditional pattern base*

Source code 4.14 merupakan potongan program dalam pembangkitan *conditional pattern base*. Lintasan dan pola akhiran didapat dari *FP-Tree* yang telah dibuat.

```
/*------ potongan source code pembankitan conditional FP-Tree -------*/
...
for ($tree=0; $tree < count($Tree); $tree++) {
 if ($nodeLink[$itemPolanya][$NodeItemPola] == $Tree[$tree]) {
   $polaNode = $Tree[$tree];
   $polaItem[$NodeItemPola][] = $polaNode;
   for ($tree2=0; $tree2 < count($Tree); $tree2++) {
    if ($polaNode['parent'] == $Tree[$tree2]['id'] && $polaNode['parent']
        != 0) {
       $polaNode = $Tree[$tree2];
       $polaNode['count'] = 0;
       array_push($polaItem[$NodeItemPola], $polaNode);
       for ($levelcount=2; $levelcount <
            $nodeLink[$itemPolanya][$NodeItemPola]['level'];
            $levelcount++) {
         for ($tree3=0; $tree3 < count($Tree); $tree3++) {
          if ($polaNode['parent'] == $Tree[$tree3]['id']) {
             $polaNode = $Tree[$tree3];
             $polaNode['count'] = 0;
             array_push($polaItem[$NodeItemPola], $polaNode);
             break;
          }
         }
       }
     }
   }
 }
}
...
```

Source Code 4.15 Potongan *source code* pembangkitan *conditional FP-Tree*

Source code 4.15 merupakan potongan program dalam pembangkitan *conditional FP-Tree*. Item-item yang dipakai adalah item yang memiliki nilai kemunculan lebih besar atau sama dengan dari *minimum support*. Pemilihan item-item ini didapat dari item yang masuk pada *frequent list*.

```
/*------ potongan source code pencarian frequent itemset-------*/
...
for ($i=1; $i <= $itemset4; $i++) {
 $nodeItemGabung1 = array($pola[$PolaCari][$ItemCari][$i]['node']);
 $nodeItemGabung2 = array($pola[$PolaCari][$ItemCari][2]['node']);
 $nodeItemGabung3 = array($pola[$PolaCari][$ItemCari][3]['node']);
 $gabungItem1 = array_merge($nodeItemCari, $nodeItemGabung1);
 $gabungItem2 = array_merge($gabungItem1, $nodeItemGabung2);
 $gabungItem3 = array_merge($gabungItem2, $nodeItemGabung3);
 $IS = $gabungItem3;
 $itemset = array(
                    'itemset' => $IS,
                    'frekuensi' => $frekuensi,
                    );
...
```

Source Code 4.16 Potongan *source code* pencarian *frequent itemset*

Source code 4.16 merupakan potongan progran dalam pencarian *frequent itemset*. Potongan program tersebut untuk 4-*itemset*. Kombinasi dari 4 item yang berada pada lintasa digabungkan menjadi sebuah *itemset*.

4.3.3. Implementasi Analisis Pola

Pada proses analisis pola dilakukan perhitungan *confidence* dan *lift ratio*. Pola yang dipakai adalah pola yang memiliki nilai *confidence* lebih besar atau sama dengan dari nilai *minimum confidence*. Selain *confidence* dan *lift ratio*, parameter pemakaian pola dilihat dari *itemset*, *itemset* yang dipakai yaitu yang memiliki minimal 2 *itemset*. Tidak semua pola yang memenuhi *confidence, lift ratio*, dan memiliki *2-itermset* dipakai, untuk penemuan pola penerimaan, pola yang dipakai adalah pola yang memiliki *consequent* (item akhir/kesimpulan) atribut propinsi. Sedangkan untuk penemuan pola kelulusan, pola yang dipakai adalah pola yang memiliki *antecedent* (item pertama) atau *consequent* dengan atribut kriteria kelulusan.

```
/*------ potongan source code analisis pola kelulusan -------*/
...
$kemunculanItemset = $itemset['frekuensi'];
$cekItemIS = explode(',', $itemset['IS_kelulusan']);
for ($i=0; $i < count($cekItemIS); $i++) {
  if ($i == 0) {
    if (substr($cekItemIS[$i], 0, 1) == "K") {
        $cariKemunculanISPembagi = mysql_query("SELECT frekuensi
                                    FROM itemset_kelulusan
                                    WHERE
IS_kelulusan='$cekItemIS[$i]'");
       list($ISPembagi) = mysql_fetch_row($cariKemunculanISPembagi);
        $itemCaksen = explode(',', $itemset['IS_kelulusan'], 2);
        $cariKemunculanCAksendibagi = mysql_query("SELECT frekuensi
                                    FROM itemset_kelulusan
                                    WHERE
IS_kelulusan='$itemCaksen[1]'");
        list($CAksenDibagi) =
mysql_fetch_row($cariKemunculanCAksendibagi);
    }
   }elseif ($i == count($cekItemIS)-1) {
     if (substr($cekItemIS[$i], 0, 1) == "K") {
        $hapusItemC = ",$cekItemIS[$i]";
        $C = str_replace($hapusItemC, '', $itemset['IS_kelulusan']);
        $cariKemunculanISPembagi = mysql_query("SELECT frekuensi
                                    FROM itemset_kelulusan
                                    WHERE IS_kelulusan='$C'");
        list($ISPembagi) = mysql_fetch_row($cariKemunculanISPembagi);
        $ItemCaksen = "$cekItemIS[$i]";
        $cariKemunculanCAksendibagi = mysql_query("SELECT frekuensi
                                    FROM itemset_kelulusan
                                    WHERE IS_kelulusan='$ItemCaksen'");
        list($CAksenDibagi) =
mysql_fetch_row($cariKemunculanCAksendibagi);
     }
     }else {
        $ISPembagi = 0;
        $CAksenDibagi = 0;
```

```
    }
}
if ($ISPembagi!=0 && $CAksenDibagi!=0) {
$rumusConfidence = $kemunculanItemset/$ISPembagi;
$rumusConfidenceAksen = $CAksenDibagi/$TID_k;
}
$RumusLiftRatio = $rumusConfidence/$rumusConfidenceAksen;
...
```

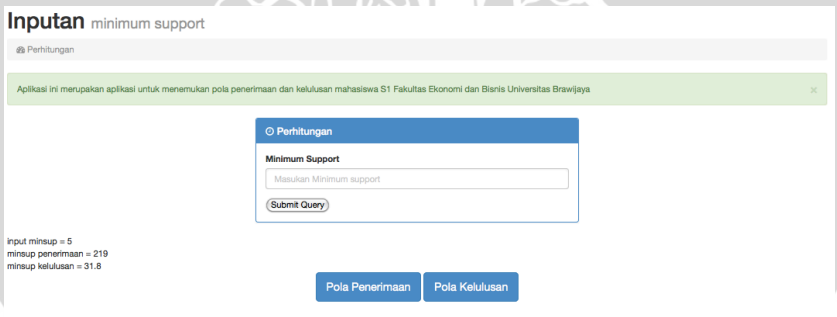
Source Code 4.17 Potongan *source code* analisis pola kelulusan

```
/*------ potongan source code analisis pola kelulusan -------*/
...
$kemunculanISPembagi = str_replace($dicari, '',
$itemset['IS_penerimaan']);
$cariKemunculanISPembagi = mysql_query("SELECT frekuensi
                            FROM itemset_penerimaan
                            WHERE
IS_penerimaan='$kemunculanISPembagi'");
list($ISPembagi) = mysql_fetch_row($cariKemunculanISPembagi);
$rumusConfidence = $kemunculanItemset/$ISPembagi;
$itemCaksen = explode(',', $itemset['IS_penerimaan'], 2);
$cariKemunculanCAksendibagi = mysql_query("SELECT frekuensi
                            FROM itemset_penerimaan
                            WHERE IS_penerimaan='$itemCaksen[1]'");
list($CAksenDibagi) = mysql_fetch_row($cariKemunculanCAksendibagi);
$rumusConfidenceAksen = $CAksenDibagi/$TID_p;
$RumusLiftRatio = $rumusConfidence/$rumusConfidenceAksen;
...
```

Source Code 4.18 Potongan *source code* analisis penerimaan

4.4. Implementasi Antarmuka

Antarmuka suatu aplikasi merupakan gerbang penghubung antara pengguna dengan sistem. Aplikasi ini mempunyai beberapa antar muka, mulai dari pemasukan *minimum support* hingga antarmuka hasil analisis pola.



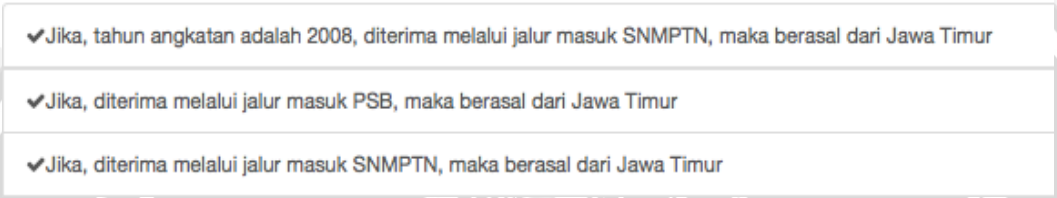
Gambar 4.1 Antarmuka pemasukan *minimum support*

Gambar 4.1 menunjukkan proses pemasukkan *minimum support* yang dilakukan oleh pengguna. Saat menentukan *minimum support* akan ditampilkan pula nilai *minimum support* dari masing-masing pola.

Itemset	Frekuensi	Support (%)	Confidence (%)	Lift Ratio
K2,B1	273	42.9245283019	80.0586510264	1.13908953138
S4,K2	35	5.50314465409	71.4285714286	1.33221617093
B2,PS4,K3	57	8.96226415094	76	1.63850847458

Gambar 4.2 Antarmuka hasil analisa pola

Gambar 4.2 menunjukkan hasil analisis pola dari proses algoritma *FP-Growth*. Untuk penemuan pola penerimaan maupun kelulusan, tampilan dari halaman sama persis. Hanya perbedaanya terletak pada isi dari tabel hasil analisis itu sendiri.



Gambar 4.3 Antarmuka analisa pola sesudah normalisasi

Gambar 4.3 menunjukkan hasil pola dari perhitungan sudah dikembalikan mejadi format yang lebih mudah dimengerti oleh pengguna. Tampilan untuk pola penerimaan maupun kelulusan sama, yang membedakan hanya isi dari pola itu sendiri.