DAFTAR PUSTAKA

Andi Amri. (2020). Dampak Covid-19 Terhadap UMKM di Indonesia. *Jurnal BRAND*, *2*(1), 124. https://ejournals.umma.ac.id/index.php/brand/article/view/605

Ema Yahniar, N. I. A. (2021). Sistem Pendukung Keputusan Penentuan Penerimaan Bahan Pangan Bersubsidi Untuk Keluarga Miskin Dengan Metode Electre (Studi Kasus: Kantor Camat Selesai). *Jurnal Sistem Informasi Kaputama*, *5*(1), 27–28. https://doi.org/https://doi.org/10.1234/jsik.v5i1.427

Faisal, S. D. H. P. (2015). Sistem Penunjang Keputusan Pemilihan Sekolah Menengah Kejuruan Teknik Komputer dan Jaringan yang terfavorit dengan menggunakan Multi-Criteria Decision Making. *Jurnal Teknologi Informasi Dan Ilmu Komputer (JTIIK)*, *2*(1), 11–19. https://doi.org/http://dx.doi.org/10.25126/jtiik.201521123

Hendryan Winata, M. R. S. Y. M. (2019). Sistem Pendukung Keputusan Menentukan Nasabah yang Layak Menerima Kredit Angsuran Pembiayaan Mekar di Kecamatan Medan Belawan pada PT. PNM dengan Metode AHP. *Jurnal SAINTIKOM*, *18*(2), 92–103. https://doi.org/https://doi.org/10.53513/jis.v18i2.147

Kadek Ary Putri Sanjiwani, M. N. (2020). Pengaturan Pembiayaan Usaha Mikro dan Kecil oleh Pemerintah sebagai Upaya Strategis dalam Mendukung Perekonomian Nasional. *Jurnal Kertha Semaya*, *8*(9), 1396–1406. https://doi.org/blob:https://ojs.unud.ac.id/e022b562-5a74-44a7-907c-b6dfbaabc09b

Kusumadewi, Sri., Hartati, Sri., Harjoko, Agus., da Wardoyo, Rentyanto. 2006. *Fuzzy Multi Attribute Decision Making (Fuzzy MADM*). Graha Ilmu. Yogyakarta.

Liani, H. H. A. dan P. C. Y. (2017). Dampak Pinjaman Dan Bantuan Modal Kerja Terhadap Kinerja Usaha (Studi Kasus UMKM Binaan KADIN Jawa Tengah). *Jurnal USM*, *10*, 16. https://doi.org/http://dx.doi.org/10.26623/jreb.v10i3.789

Parlina, Iin. *Analisis Sistem Pendukung Keputusan Pemilihan Anggota Paskibraka Menggunakan Metode Electre*.JITE (Journal Of Informatics And Telecommunication Engineering), Vol.2, No.1 (2018).

Ramadhan, Mukhlis., Nofriansyah, Dicky., Rizky, Firahmi. *Sistem Pendukung* Keputusan *Pemberian Bantuan Program Keluarga Harapan dengan Metode Elimination Et Choix Traduisant la Realite (ELECTRE)* Studi Kasus Kecamatan Borbor. Sains dan Komputer (SAINTIKOM) Vol.18, No.1. Februari 2019, pp. 17-29, P-ISSN: 1978-6603: 2615-3475.

Rahmadina, R. M. (2020). Pengaruh Program PNM Mekaar Terhadap Pendapatan Masyarakat Miskin Kecamatan Campalagian. *J-ALIF*, *5*(1), 74–86. https://doi.org/http://dx.doi.org/10.35329/jalif.v5i1.1788

Roulina Simarmata, Rahmat W. Sembiring, Rafiqa Dewi, Anjar Wanto.*Penentuan Masyarakat Penerima Bantuan Perbaikan Rumah di Kecamatan Siantar Barat Menggunakan Metode ELECTRE*. JoSYC, Vol.1 No.2, Februari 2020.

LAMPIRAN 1

LISTING PROGRAM

// Electre Algorithm

```
<?php
/* NOTE
 * R = matrix r = normalized matrix
 * w = weight
 * V or v = weighted normalized matrix
 * C/c = concordance
 * D/d = discordance
 * F/f = dominant matrix concordance
 * D/d = dominant matrix discordance */

class Electre_algorithm {
    protected $object;
    protected $rows;
    protected $columns;
    protected $columns_arr;
    public function __construct($object = NULL, $columns_arr = NULL, $ro
ws = NULL, $columns = NULL) {
        $this->object = $object;
        $this->rows = $rows;
        $this->columns = $columns;
        $this->columns_arr = $columns_arr;}
```

```
// STEP 1 * Normalization matrix //
   public function normalization() {
    // $example = 5 / sqrt(5*5 + 4*4 + 2*2 + 3*3 + 4*4);
    // $normalization = round($example, 2);
    // $columns = [[4,4,5,4], [5,4,4,3], [4,3,2,3], [2,3,4,5], [3,5,3,2]
];
       $columns = $this->columns_arr;
       $squared_arr = [];
       $squared_arr_sum = [];

   for ($row = 0; $row <= $this->rows; $row++) {
    for ($column = 0; $column < count($columns[$row]); $column++) {
       $squared = $columns[$row][$column] * $columns[$row][$column];
       $squared_arr[$row][] = $squared;       }

       $squared_arr_sum[$row] = array_sum($squared_arr[$row]); }
    foreach ($columns as $key_row => $row) {
     foreach ($row as $key_col => $col) {
        $calculate = $col / sqrt($squared_arr_sum[$key_row]);
```



```
// normalized "r" matrix, example(r11, r21, r31..n)
       $key_r = 'r_'.($key_col + 1).'_'.($key_row + 1);
       $normalization_arr[$key_r] = $this>_number_format($calculate);
}}
       $normalization_arr;
/* create new row and column after you get normalization result
 * based number of rows and numbers of columns
```

```
*/
        $matrix = array_chunk($normalization_arr, $this->rows);
return [$matrix, $normalization_arr];}

// STEP 2 weighting on the normalized matrix//
    public function weighting($matrix, $normalization) {
// change key for object and including weight
    foreach ($this->object as $key => $value) {
        $weight_key = 'w_'.($value['bobot']);
// $normalization_groups[$weight_key] = $normalization[$key];
        $normalization_groups[] = [$weight_key => $matrix[$key]];
        $normalization_groups;
// multiplication rows with weight
foreach ($normalization_groups as $key_normalization => $normalization_g
roup) {
     foreach ($normalization_group as $key_group => $group) {
        $weight = (int) explode('_', $key_group)[1];
     foreach ($group as $key => $value) {
// normalized "v" weighting matrix, example(v11, v21, v31..n)
        $key_v = 'v_'.($key + 1).'_'.($key_normalization + 1);
        $weight_normalized_arr[$key_v] = $this-
>_number_format($value * $weight); }}}
        $weight_normalized_arr;
        $weight_normalized = array_chunk($weight_normalized_arr, $this-
>rows);
        return [$weight_normalized, $weight_normalized_arr];}

// STEP 3 Determine concordance and discordance sets //
```

```
     * Determine C11, C12, C13, etc as many as the number of matrix

     * and then compare with Y21, Y22, Y23 etc as many as the number of
matrix

     * description: C1(K)1(L), C1(K)2(L), Y1(L)1(J), Y2(L)2(J) etc

     *

     * ex: if c11 >= y2,1 = value c11(3,201) >= y2,1 (2,561) = true (gre
ather than) * */

    public function concordance_and_discordance($matrix_v, $weight_norma
lized_arr) {

        $concordance = $this-
>concordance($matrix_v, $weight_normalized_arr);

        $discordance = $this-
>discordance($matrix_v, $weight_normalized_arr);

        return [$concordance, $discordance];      }


// STEP 4
Calculate matrix concordance and matrix discordance based on weight //

    public function calculate_concordance_and_discordance($matrix_v, $co
ncordance, $discordance) {

        list($matrix_c, $concordance_arr) = $this-
>calcutating_concordance($matrix_v, $concordance);

        list($matrix_d, $discordance_arr) = $this-
>calcutating_discordance($matrix_v, $discordance);

return [$matrix_c, $concordance_arr, $matrix_d, $discordance_arr]; }


// STEP 5 determine dominant matrix concordance and discordance //

    public function dominant_matrix($matrix_c, $matrix_d) {

        foreach ($matrix_c as $key => $value) {

            $sum_matrix_c[] = array_sum($value);}

        foreach ($matrix_d as $key => $value) {
```

```
            $sum_matrix_d[] = array_sum($value);}


        $threshold_c = array_sum($sum_matrix_c) / (($this-
>rows * ($this->rows - 1)));

        $threshold_c = $this->_number_format($threshold_c);

        $matrix_f = $this->get_dominant_matrix($matrix_c, $threshold_c);


        $threshold_d = array_sum($sum_matrix_d) / (($this-
>rows * ($this->rows - 1)));

        $threshold_d = $this->_number_format($threshold_d);

        $matrix_g = $this->get_dominant_matrix($matrix_d, $threshold_d);

        return [$matrix_f, $matrix_g]; }


// STEP 6 determinant aggregate dominance matrix //

    public function dominant_aggregation($matrix_f, $matrix_g) {

        $matrix_e_arr = $this->set_array_keys('e');


   foreach ($matrix_e_arr as $key => $value) {

        $index = explode('_', $key);

        $arr_matrix_e[$key] = $matrix_f[$index[1]][$index[2]] * $matrix_
g[$index[1]][$index[2]];  }

        $matrix_e = $this-
>get_concordance_discordance_matrix($arr_matrix_e);

        return [$matrix_e, $arr_matrix_e];          }


// STEP 7 elimination alternative less favourable //

    public function eliminations($matrix_e, $arr_matrix_e) {
```

```
/* later will be used
        for ($row=0; $row <= $this->rows; $row++) {
            for ($column=0; $column < $this->rows; $column++) {
                if ($row == $column) {
                    $table[$row + 1][$column + 1] = '-';
                } else {
                    $table[$row + 1][$column + 1] = NULL;   } } }

        $arr_list = $table;
        unset($arr_list[count($table)]);

    foreach ($arr_matrix_e as $key_object => $value_object) {
        $index = explode('_', $key_object);
        $arr_list[$index[1]][$index[2]] = $value_object;   }
*/
    foreach ($matrix_e as $key => $value) {
        $arr_count[$key] = count(array_keys($matrix_e[$key], 1));}
        $arr_eliminate_count = $arr_count;
 /* later will be use
        $arr_count = array_diff($arr_count, [0]);
        $i = 0;
        foreach ($arr_count as $key => $value) {
            if ($value > $i) {
                $i = $value;
                $arr_eliminate = [$key => $arr_count[$key]];      }}
 */
```

```
return $arr_eliminate_count;}


    public function rating_result($alternative, $arr_eliminate_count) {
    $i = 1;

    foreach ($alternative as $key => $value) {

        $ratings[$key] = $arr_eliminate_count[$i++];     }

        arsort($ratings);

        $number = 1;

   foreach ($ratings as $key => $value) {

           $ratings[$key] = $number++; }

return $ratings;    }


// INCLUDING OF STEP 3 //

protected function concordance($matrix_v, $weight_normalized_arr) {

       $concordance_arr = $this->set_array_keys('c');

       $i = 1;

  foreach ($concordance_arr as $key_concordance => $concordance_value) {

       $split_key = explode('_', $key_concordance);

       $index_concordance = [];

   for ($column = 0; $column < $this->columns; $column++) {

       $col = $column + 1;

          list($pattern_c, $pattern_y) = $this-
>set_pattern_weighting_key('v', $split_key, $col);


   if ($weight_normalized_arr[$pattern_c] >= $weight_normalized_arr[$pat
tern_y]) {

     array_push($index_concordance, $col);
```

```
        $concordance[$key_concordance] = $index_concordance; }  }
   $i++;       }

return $concordance; }


// INCLUDING OF STEP 3*/

protected function discordance($matrix_v, $weight_normalized_arr) {

        $discordance_arr = $this->set_array_keys('d');

        $i = 1;

   foreach ($discordance_arr as $key_discordance => $discordance_value)
{  $split_key = explode('_', $key_discordance);

        $index_discordance = [];

   for ($column = 0; $column < $this->columns; $column++) {

        $col = $column + 1;

          list($pattern_c, $pattern_y) = $this-
>set_pattern_weighting_key('v', $split_key, $col);


   if ($weight_normalized_arr[$pattern_c] < $weight_normalized_arr[$patt
ern_y]) {

   array_push($index_discordance, $col);

        $discordance[$key_discordance] = $index_discordance; } }

             $i++;}

return $discordance; }


// INCLUDING OF STEP 4 - CONCORDANCE//

   protected function calcutating_concordance($matrix_v, $concordance) {

        $concordance_arr = $this->set_array_keys('c');

        $weight_columns = $this->create_weight_column($matrix_v);

  foreach ($concordance as $key_concordance => $value_concordance) {
```

```
     foreach ($value_concordance as $key => $value) {

       $new_concordance_arr[$key_concordance][] = $weight_columns[$value
];   } }


     foreach ($new_concordance_arr as $key => $value) {

       $concordance_arr[$key] = array_sum($new_concordance_arr[$key]);
      }


       $matrix_c = $this-
>get_concordance_discordance_matrix($concordance_arr);

return [$matrix_c, $concordance_arr]; }


// INCLUDING OF STEP 4 - DISCORDANCE//

    protected function calcutating_discordance($matrix_v, $discordance)
{

        $discordance_arr = $this->set_array_keys('d');

        $replace_matrix_v = $this->replace_key_matrix_v($matrix_v);


foreach ($discordance as $key_discordance => $value_discordance) {

  foreach ($value_discordance as $key => $value) {

        $new_discordance_arr[$key_discordance][$value] = $replace_matrix
_v[$value];  }}


foreach ($new_discordance_arr as $key_discordance => $value_discordance)
 {

        $index = explode('_', $key_discordance);


foreach ($value_discordance as $key_col => $col_value) {
```

```
        $differences[$key_discordance][$key_col] = abs($col_value[$inde
x[1] - 1] - $col_value[$index[2] - 1]);


foreach ($replace_matrix_v as $key_v => $value_v) {

        $all_differences[$key_discordance][] = abs($value_v[$index[1] -
 1] - $value_v[$index[2] - 1]);  }}}


  $max_differences = $this->set_differences_matrix($differences);

  $all_max_differences = $this-
>set_differences_matrix($all_differences);


  foreach ($max_differences as $key_difference => $difference) {

  foreach ($all_max_differences as $key_all_difference => $all_differenc
e) {

    if ($key_difference == $key_all_difference) {

       $discordance_arr[$key_difference] = $this-
>_number_format($difference / $all_difference);     }}}


       $matrix_d = $this-
>get_concordance_discordance_matrix($discordance_arr);

return [$matrix_d, $discordance_arr];

      }

// Setting array

     * ex: array('c_1_1' => array(), 'c_1_2' => array());

     * note: c or d similar to concordance and discordance

//

    private function set_array_keys($pattern) {

        for ($row = 0; $row < $this->rows; $row++) {
```

```
            for ($column = 0; $column < $this->rows; $column++) {
               if ($row != $column) {
                   $key = $pattern.'_'.($row + 1).'_'.($column + 1);
                   $args[$key] = NULL; }}}
return $args; }

    private function _number_format($number) {
        return number_format($number, 3, '.', ''); }
    private function set_pattern_weighting_key($pattern, $object, $colum
n) {
        $pattern_c = $pattern.'_'.$object[1].'_'.$column;
        $pattern_y = $pattern.'_'.$object[2].'_'.$column;
        return [$pattern_c, $pattern_y];
        }

// create column based on weight or criteria
     * ex: column C1 = weight 5, column C2 = weight 4
//
    private function create_weight_column() {
        foreach ($this->object as $key => $value) {
            for ($column=0; $column < $this->columns ; $column++) {
                if ($key == $column) {
                   $weight_columns[($column + 1)] = $value['bobot'];
                } } }
return $weight_columns; }
```

```
// replace array index starting form 0,1,2,3 to 1,2,3, //
    private function replace_key_matrix_v($matrix_v) {
        foreach ($matrix_v as $key_v => $matric_v) {
            for ($i = 0; $i < count($matric_v); $i++) {
                $matrix[$key_v+1][] = $matric_v[$i]; }}
return $matrix;}

// set differences matrix //
    private function set_differences_matrix($differences) {
        foreach ($differences as $key => $difference) {
          $matrix[$key] = max($difference);}
return $matrix; }

// get concordance matrix and discordance matrix //
    private function get_concordance_discordance_matrix($object) {
        $arr_list = $this->create_matrix_table();
        foreach ($object as $key_object => $value_object) {
            $index = explode('_', $key_object);
            $arr_list[$index[1]][$index[2]] = $value_object;   }
return $arr_list;    }

    private function create_matrix_table() {
        for ($row=0; $row < $this->rows; $row++) {
            for ($column=0; $column < $this->rows; $column++) {
                if ($row == $column) {
                    $table[$row + 1][$column + 1] = '-';
```

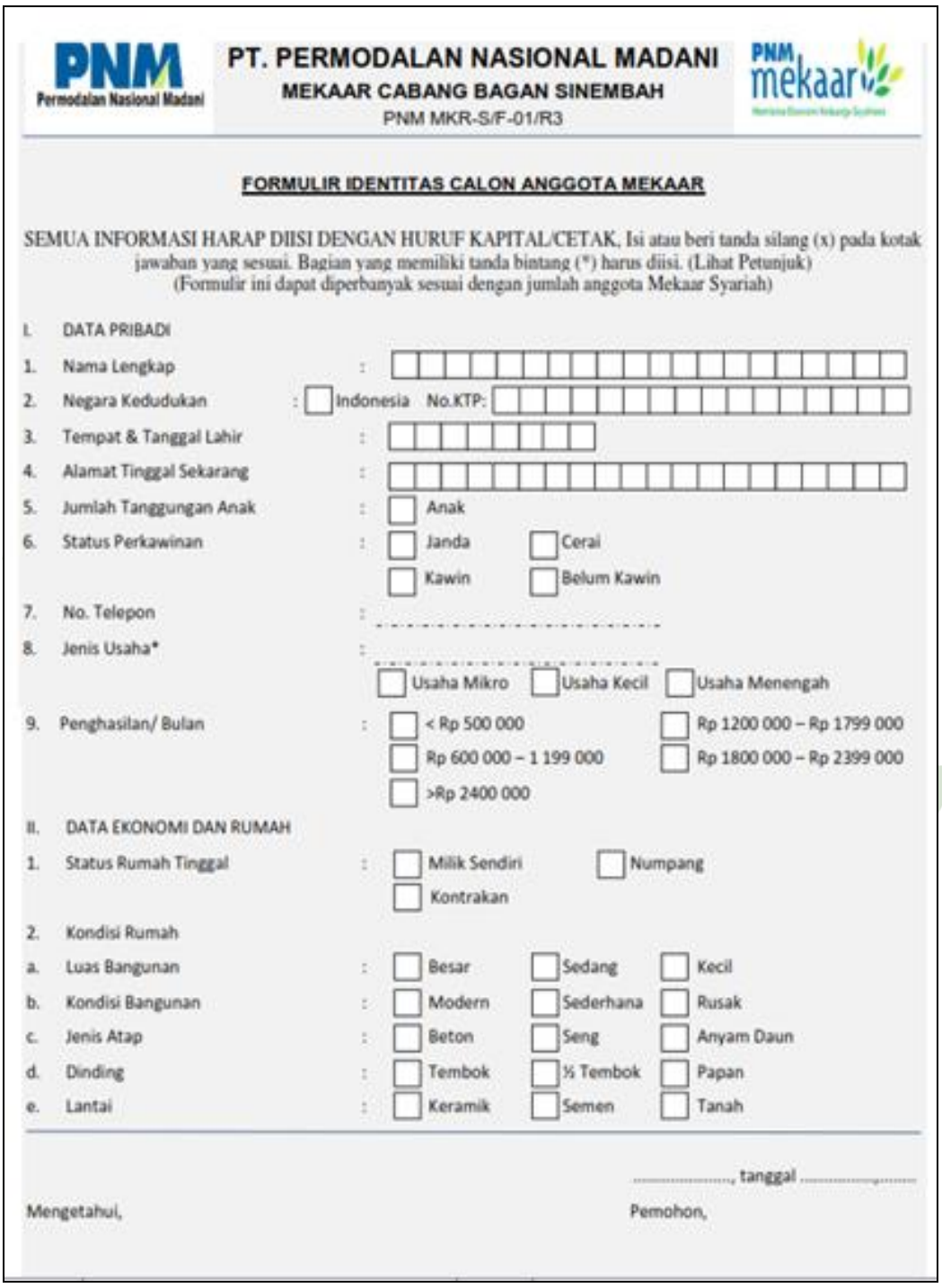
```
                } else {
                    $table[$row + 1][$column + 1] = NULL;}}}
        return $table; }


// INCLUDING STEP 5 //
    private function get_dominant_matrix($matrix, $threshold) {
        $matrix_table = $this->create_matrix_table();


        foreach ($matrix as $key_matrix => $matrix_list) {
            foreach ($matrix_list as $key => $matric) {
                if ($matric == '-') {
                    $matrix_dominant[$key_matrix][$key] = '-';
                } else {
                    if ($matric >= $threshold) {
                        $matrix_dominant[$key_matrix][$key] = 1;
                    } else {
                        $matrix_dominant[$key_matrix][$key] = 0;
                    } }} }
        return $matrix_dominant;}}
```

LAMPIRAN 2

ANGKET



PT. PERMODALAN NASIONAL MADANI
MEKAAR CABANG BAGAN SINEMBAH
PNM MKR-S/F-01/R3

FORMULIR IDENTITAS CALON ANGGOTA MEKAAR

SEMUA INFORMASI HARAP DIISI DENGAN HURUF KAPITAL/CETAK, Isi atau beri tanda silang (x) pada kotak jawaban yang sesuai. Bagian yang memiliki tanda bintang (*) harus diisi. (Lihat Petunjuk)
(Formulir ini dapat diperbanyak sesuai dengan jumlah anggota Mekaar Syariah)

I. DATA PRIBADI

1. Nama Lengkap :
2. Negara Kedudukan : ☐ Indonesia No.KTP:
3. Tempat & Tanggal Lahir :
4. Alamat Tinggal Sekarang :
5. Jumlah Tanggungan Anak : ☐ Anak
6. Status Perkawinan : ☐ Janda ☐ Cerai ☐ Kawin ☐ Belum Kawin
7. No. Telepon : -
8. Jenis Usaha* : -
   ☐ Usaha Mikro ☐ Usaha Kecil ☐ Usaha Menengah
9. Penghasilan/ Bulan : ☐ < Rp 500 000 ☐ Rp 1200 000 – Rp 1799 000 ☐ Rp 600 000 – 1 199 000 ☐ Rp 1800 000 – Rp 2399 000 ☐ >Rp 2400 000

II. DATA EKONOMI DAN RUMAH

1. Status Rumah Tinggal : ☐ Milik Sendiri ☐ Numpang ☐ Kontrakan
2. Kondisi Rumah
   a. Luas Bangunan : ☐ Besar ☐ Sedang ☐ Kecil
   b. Kondisi Bangunan : ☐ Modern ☐ Sederhana ☐ Rusak
   c. Jenis Atap : ☐ Beton ☐ Seng ☐ Anyam Daun
   d. Dinding : ☐ Tembok ☐ ½ Tembok ☐ Papan
   e. Lantai : ☐ Keramik ☐ Semen ☐ Tanah

....., tanggal .....

Mengetahui, Pemohon,

LAMPIRAN 3

HASIL RANGKING METODE ELECTRE

Hasil Perangkingan Seleksi Penerima Bantuan UMKM Program Mekaar

No	Alternatif	Nilai akhir	Keterangan
1	A7 (Maymunah)	4	Layak
2	A10 (Rosinah br Nasution)	4	Layak
3	A3 (Sarifah Aini)	3	Layak
4	A5 (Juriah Br Ritonga)	3	Layak
5	A9 (Nurmauli dia)	2	Tidak layak
6	A4 (Anita Koto)	1	Tidak layak
7	A1 (Ernita)	1	Tidak layak
8	A2 (Millisah)	1	Tidak layak
9	A6 (Siti Moyana Sinaga)	1	Tidak layak
10	A8 (Meri Agustina)	0	Tidak layak

LAMPIRAN 4

DAFTAR RIWAYAT HIDUP

I. DATA PRIBADI

Nama Lengkap	: Fitri Aulia
Tempat, Tanggal Lahir	: Tanjung Balai,11 Desember 1998
Alamat Rumah	: Jl.Imam Bonjol Ujung, Kec.Bagan Sinembah, Kab.Rokan Hilir, Prov.Riau
Email	: Fitriaulia1998@gmail.com
No-HP	: 082288449557
Agama	: Islam
Status/ Pekerjaan	: Single/ Mahasiswi
Tinggi/ berat badan	: 165 cm/ 57 kg
Kemampuan	: Komputerisasi (Microsoft Office)
Motto	: Jadilah diri sendiri dan lakukan yang terbaik semampu kita, belajar, berdoa dan berusaha.

II. DATA PENDIDIKAN

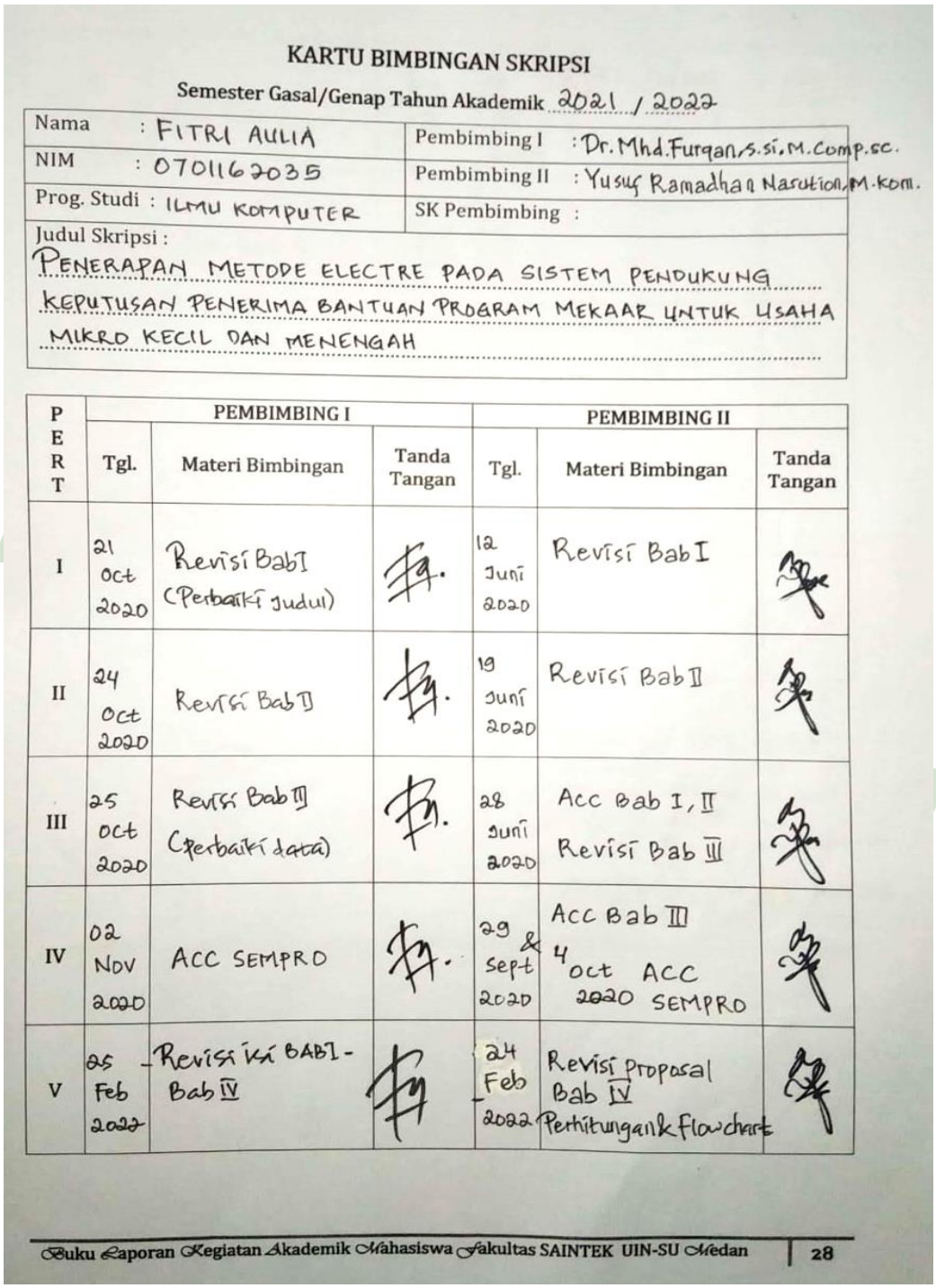
SD	: SD Negeri 003 Bagan Sinembah (2004-2010)
SMP	: SMP Negeri 1 Bagan Sinembah (2010-2013)
SMA	: SMA Negeri 1 Bagan Sinembah (2013-2016)
Perguruan Tinggi	: Universitas Islam Negeri Sumatera Utara Medan (2016-2022)

Medan, Maret 2022

Fitri Aulia

LAMPIRAN 5

KARTU BIMBINGAN SKRIPSI



KARTU BIMBINGAN SKRIPSI

Semester Gasal/Genap Tahun Akademik 2021 / 2022

Nama : FITRI AULIA	Pembimbing I : Dr. Mhd. Furqan, S.Si, M.Comp.Sc.
NIM : 0701162035	Pembimbing II : Yusuf Ramadhan Nasution, M.Kom.
Prog. Studi : ILMU KOMPUTER	SK Pembimbing :
Judul Skripsi : PENERAPAN METODE ELECTRE PADA SISTEM PENDUKUNG KEPUTUSAN PENERIMA BANTUAN PROGRAM MEKAAR UNTUK USAHA MIKRO KECIL DAN MENENGAH	

PERT	PEMBIMBING I			PEMBIMBING II		
	Tgl.	Materi Bimbingan	Tanda Tangan	Tgl.	Materi Bimbingan	Tanda Tangan
I	21 Oct 2020	Revisi Bab I (Perbaiki Judul)		12 Juni 2020	Revisi Bab I	
II	24 Oct 2020	Revisi Bab II		19 Juni 2020	Revisi Bab II	
III	25 Oct 2020	Revisi Bab II (perbaiki data)		28 Juni 2020	Acc Bab I, II Revisi Bab III	
IV	02 Nov 2020	ACC SEMPRO		29 & 4 Sept Oct 2020	Acc Bab III ACC SEMPRO	
V	25 Feb 2022	Revisi isi BAB I - Bab IV		24 Feb 2022	Revisi Proposal Bab IV Perhitungan & Flowchart	

VI	7 maret 2022	Acc Bab 1		14 maret 2022	Acc. Sidang	
VII	8 maret 2022	Acc Bab 2				
VIII	9 maret 2022	Acc Bab 3				
IX	10 maret 2022	Acc Bab 4 Acc Bab 5				
X	14 maret 2022	Acc Sidang				

Medan, 16 Maret 2022
An. Dekan
Ketua Jurusan/Program Studi
ILMU KOMPUTER

IIKA ZUFRIA, S.Kom., M.Kom.
NIP. 198506042015031006

Catatan: *Pada saat bimbingan, kartu ini harus diisi dan ditandatangani oleh pembimbing*