

**SKRIPSI**

**SISTEM PENILAIAN SIDANG SKRIPSI 2 DENGAN  
ANGULARJS**



**BILLY YANUAR**

**NPM: 2012730017**

**PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS  
UNIVERSITAS KATOLIK PARAHYANGAN**

**«tahun»**



**UNDERGRADUATE THESIS**

**«JUDUL BAHASA INGGRIS»**



**BILLY YANUAR**

**NPM: 2012730017**

**DEPARTMENT OF INFORMATICS  
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES  
PARAHYANGAN CATHOLIC UNIVERSITY  
«tahun»**



**LEMBAR PENGESAHAN**

**SISTEM PENILAIAN SIDANG SKRIPSI 2 DENGAN  
ANGULARJS**

**BILLY YANUAR**

**NPM: 2012730017**

**Bandung, «tanggal» «bulan» «tahun»**

**Menyetujui,**

**Pembimbing Utama**

**Pembimbing Pendamping**

**«pembimbing utama/1»  
Ketua Tim Penguji**

**«pembimbing pendamping/2»  
Anggota Tim Penguji**

**«penguji 1»**

**«penguji 2»**

**Mengetahui,**

**Ketua Program Studi**

**Thomas Anung Basuki, Ph.D.**



## PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

### **SISTEM PENILAIAN SIDANG SKRIPSI 2 DENGAN ANGULARJS**

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,  
Tanggal «tanggal» «bulan» «tahun»

Meterai

Billy Yanuar  
NPM: 2012730017





## ABSTRAK

«Tuliskan abstrak anda di sini, dalam bahasa Indonesia» Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetur.

**Kata-kata kunci:** «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Indonesia»



## ABSTRACT

«Tuliskan abstrak anda di sini, dalam bahasa Inggris» Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetur.

**Keywords:** «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Inggris»



*«kepada siapa anda mempersembahkan skripsi ini...?»*



## KATA PENGANTAR

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Bandung, «bulan» «tahun»

Penulis





## DAFTAR ISI

<b>KATA PENGANTAR</b>	<b>xv</b>
<b>DAFTAR ISI</b>	<b>xvii</b>
<b>DAFTAR GAMBAR</b>	<b>xix</b>
<b>DAFTAR TABEL</b>	<b>xx</b>
<b>1 PENDAHULUAN</b>	<b>1</b>
1.1 Latar Belakang . . . . .	1
1.2 Rumusan Masalah . . . . .	2
1.3 Tujuan . . . . .	2
1.4 Batasan Masalah . . . . .	2
1.5 Metode Penelitian . . . . .	2
1.6 Sistematika Penulisan . . . . .	3
<b>2 DASAR TEORI</b>	<b>5</b>
2.1 CodeIgniter . . . . .	5
2.1.1 Flowchart Aplikasi CodeIgniter . . . . .	5
2.1.2 Model-View-Controller . . . . .	6
2.1.3 Controller . . . . .	6
2.1.4 Views . . . . .	7
2.1.5 Models . . . . .	8
2.1.6 Helper . . . . .	9
2.1.7 Basis data . . . . .	10
2.1.8 Konfigurasi Basis Data . . . . .	10
2.2 AngularJS . . . . .	12
2.2.1 Gambaran Konseptual . . . . .	13
2.2.2 Directives . . . . .	14
2.2.3 Data Binding . . . . .	14
2.2.4 Model-View-Controller(MVC) . . . . .	15
2.3 Twitter Bootstrap . . . . .	17
2.3.1 Grid System . . . . .	17
2.3.2 Form Class . . . . .	18
<b>3 ANALISIS</b>	<b>21</b>
3.1 Analisis Sistem Kini . . . . .	21
3.1.1 Form Rekapitulasi Penilaian . . . . .	21
3.1.2 Form Berita Acara Sidang Skripsi . . . . .	21
3.2 Analisis Sistem Usulan . . . . .	22
3.2.1 Analisis Back End . . . . .	22
3.2.2 Analisis Front End . . . . .	24
3.2.3 Analisis Basis Data . . . . .	30

3.3	Use Case . . . . .	32
<b>4</b>	<b>PERANCANGAN</b>	<b>33</b>
4.1	Perancangan Kelas . . . . .	33
4.2	Routes . . . . .	33
4.3	Controllers . . . . .	33
4.4	Models . . . . .	34
4.5	Perancangan Basis Data . . . . .	34
4.6	Perancangan Tampilan . . . . .	35
<b>5</b>	<b>IMPLEMENTASI DAN PENGUJIAN</b>	<b>37</b>
5.1	Implementasi . . . . .	37
5.1.1	Lingkungan Implementasi dan Pengujian . . . . .	37
5.1.2	Hasil Implementasi . . . . .	37
5.2	Hasil Pengujian . . . . .	40
5.2.1	Pengujian Eksperimental . . . . .	40
5.2.2	Pengujian Fungsional . . . . .	41
<b>6</b>	<b>KESIMPULAN DAN SARAN</b>	<b>43</b>
6.1	Kesimpulan . . . . .	43
6.2	Saran . . . . .	43
	<b>DAFTAR REFERENSI</b>	<b>45</b>
	<b>A FORM PENILAIAN SKRIPSI</b>	<b>47</b>
	<b>B THE SOURCE CODE</b>	<b>49</b>

## DAFTAR GAMBAR

2.1	Flowchart CodeIgniter . . . . .	5
2.2	Contoh Kode Controller . . . . .	6
2.3	Contoh Method ber-Parameter . . . . .	7
2.4	Penggantian Variable pada Route . . . . .	7
2.5	Contoh File View . . . . .	8
2.6	Contoh Pemanggilan File View pada Controller . . . . .	8
2.7	Contoh Query Builder insert . . . . .	9
2.8	Contoh Query Builder Update . . . . .	9
2.9	Contoh Pemanggilan File Model pada Controller . . . . .	9
2.10	Kode yang ditambahkan untuk menjalankan helper . . . . .	10
2.11	Kode yang ditambahkan untuk autoload basis data . . . . .	10
2.12	Konfigurasi Basis Data . . . . .	11
2.13	Data Binding Classical Templates System . . . . .	15
2.14	Data Binding pada Angular . . . . .	15
2.15	Grid Option pada Bootstrap . . . . .	18
2.16	Contoh Pembagian Grid Columns . . . . .	18
2.17	Contoh Penggunaan Kelas Form . . . . .	19
2.18	Contoh Hasil Penggunaan Kelas Form . . . . .	19
3.1	Use case diagram . . . . .	32
4.1	Gambar diagram kelas <i>file controllers</i> . . . . .	34
4.2	Perkiraan Tampilan . . . . .	36
5.1	Formulir berita acara sidang skripsi 2 terisi . . . . .	38
5.2	Formulir rekapitulasi ketua tim penguji terisi . . . . .	38
5.3	Formulir rekapitulasi anggota tim penguji terisi . . . . .	39
5.4	Formulir rekapitulasi pembimbing terisi . . . . .	39
5.5	Ketika tombol selesai di klik . . . . .	40
5.6	Sebagian hasil pada database . . . . .	40
A.1	Form Penilaian Skripsi saat sidang . . . . .	47
A.2	Form Rekapitulasi Penilaian Skripsi saat sidang . . . . .	48

## DAFTAR TABEL

# BAB 1

## PENDAHULUAN

### 1.1 Latar Belakang

Program Studi Teknik Informatika di Universitas Katolik Parahyangan memiliki beberapa syarat kelulusan antara lain minimal sks yang lulus 144 yang terdiri dari matakuliah wajib dan pilihan, indeks prestasi minimum adalah 2.00 dengan maksimum 14 semester. Salah satu matakuliah wajib yang harus ditempuh dan lulus adalah skripsi. Skripsi di Program Studi Teknik Informatika di Universitas Katolik Parahyangan dibagi menjadi 2 matakuliah yaitu skripsi 1 dan skripsi 2.

Sistem penilaian sidang skripsi 2 pada Program Studi Teknik Informatika di Universitas Katolik Parahyangan masih bersifat manual dimana penilai mengisi data-data mahasiswa memberikan nilai untuk mahasiswa pada saat sidang dan juga melakukan penghitungan bobot nilai total.

Sifat manual ini mengakibatkan kelalaian manusia dalam melakukan penilaian pun beberapa kali tidak dapat dihindarkan. Kelalaian manusia yang biasa terjadi contohnya adalah kesalahan perhitungan nilai akhir oleh penilai, kesalahan penulisan nama dan NPM mahasiswa yang bersangkutan, kesalahan penulisan semester atau tahun ajaran saat penilaian skripsi<sup>1</sup>. Selain itu, penyimpanan nilai skripsi pun tergolong sulit karena tidak langsung dibarengi dengan nilai dan npm mahasiswa yang mengerjakan. Untuk mengatasi hal-hal tersebut, diperlukan suatu sistem yang dapat menanggulangi masalah pengisian, kalkulasi perhitungan, dan juga penyimpanan skripsi.

Menurut penjelasan di atas, maka otomatisasi sistem dalam penilaian skripsi penulis mengusulkan oleh Universitas guna mengurangi kesalahan - kesalahan kecil yang dapat berakibat fatal pada nilai mahasiswa yang bersangkutan. Berdasarkan hal tersebut dibuatlah penelitian otomatisasi sistem penilaian skripsi dengan cara membuat sebuah aplikasi berbasis web yaitu Sistem Informasi Penilaian Skripsi.

Pada penelitian ini, akan dibuat sebuah sistem penilaian yang menanggulangi masalah-masalah tersebut dengan cara membuat beberapa masukan dijadikan otomatis dan juga melakukan eksekusi perhitungan nilai akhir sesuai bobot secara otomatis. Hal ini dianggap akan memudahkan penilai dalam proses penilaian skripsi, karena penilai tidak perlu lagi repot menghitung dan juga mengisi hal-hal yang sudah terisi secara otomatis.

Dalam penelitian ini saya memakai framework AngularJS yang dimiliki oleh perusahaan *Google*. AngularJS merupakan salah satu framework yang paling sering digunakan untuk membuat sebuah aplikasi berbasis web dengan konsep *Single Page Application (SPA)*. *Single Page Application* merupakan aplikasi berbasis web yang memungkinkan sebuah halaman HTML memiliki konten - konten

---

<sup>1</sup>berdasarkan diskusi dengan dosen pembimbing

yang dapat digunakan di halaman tersebut tanpa perlu berganti ke halaman lain.

AngularJS juga bisa diintegrasikan dengan aplikasi yang menggunakan framework lain, sehingga sangat berguna dalam pengerjaan aplikasi berbasis web yang sangat luas cakupannya.

## 1.2 Rumusan Masalah

Berikut adalah susunan permasalahan yang akan dibahas pada penelitian ini:

1. Bagaimana sistem penilaian skripsi yang ada pada Program Studi Teknik Informatika di Universitas Katolik Parahyangan?
2. Bagaimana proses penyimpanan nilai skripsi?
3. Bagaimana AngularJS bekerja pada eksekusi perhitungan nilai akhir?

## 1.3 Tujuan

Berdasarkan rumusan masalah yang telah dibuat, maka tujuan penelitian ini dijelaskan ke dalam poin-poin sebagai berikut:

1. Mempelajari sistem penilaian skripsi pada Program Studi Teknik Informatika di Universitas Katolik Parahyangan
2. Merancang dan mengimplementasi proses penyimpanan nilai skripsi
3. Menentukan dan mengimplementasi AngularJS untuk mengeksekusi perhitungan nilai akhir

## 1.4 Batasan Masalah

Penelitian ini memiliki batasan-batasan seperti berikut:

1. Penelitian ini hanya dilakukan untuk form penilaian matakuliah skripsi 2

## 1.5 Metode Penelitian

Dalam penelitian ini, akan dilakukan langkah-langkah berikut:

1. Melakukan studi terhadap CodeIgniter, Twitter Bootstrap, dan AngularJS sebagai framework yang akan dipakai.
2. Melakukan perancangan untuk implementasi integrasi sistem tersebut.
3. Melakukan implementasi dari rancangan yang sudah dilakukan.
4. Melakukan pengujian pada saat sidang skripsi2 sehingga penilai dapat menguji hasil implementasi tersebut.
5. Menganalisa dan menarik kesimpulan atas hasil penelitian yang telah dilaksanakan.

---

## 1.6 Sistematika Penulisan

Berikut adalah sistematika penulisan dari dokumen ini:

- Bab 1 membahas latar belakang, rumusan masalah, tujuan penulisan, batasan-batasan, serta metode yang digunakan pada penelitian ini.
- Bab 2 membahas teori-teori yang digunakan dalam penelitian ini, yaitu AngularJS, Code Igniter, dan Twitter Bootstrap.
- Bab 3 menganalisis sistem kini, beserta perubahan-perubahan yang harus dilakukan.
- Bab 4 membahas perancangan yang dilakukan sebelum mengimplementasikan integrasi yang dimaksud, mencakup protokol, basisdata, beserta antarmukanya.
- Bab 5 membahas implementasi serta pengujian dari integrasi yang telah dilakukan.
- Bab 6 membahas kesimpulan dari keseluruhan penelitian ini, serta saran-saran yang dapat diberikan untuk penelitian berikutnya.





## BAB 2

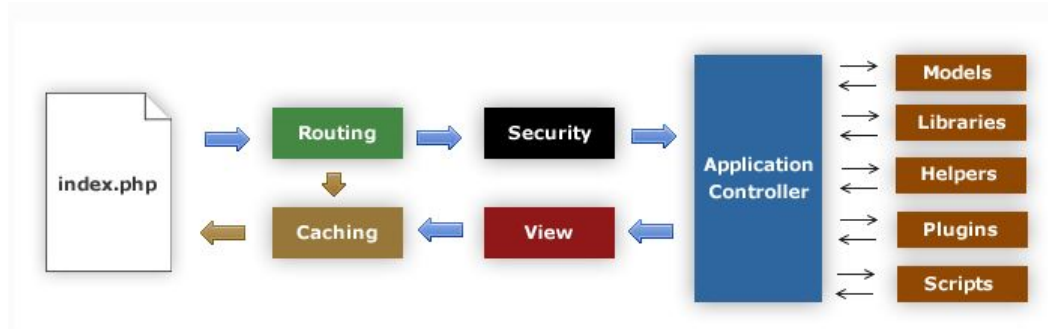
### DASAR TEORI

#### 2.1 CodeIgniter

CodeIgniter[1] merupakan sebuah peralatan bagi orang-orang yang ingin membuat sebuah *web* dengan menggunakan bahasa PHP. CodeIgniter sendiri dibuat dengan tujuan memungkinkan pengembangan proyek-proyek lebih cepat daripada menuliskan kode dari awal. Tujuan tersebut diwujudkan dengan tersedianya *library* yang berisi *task* yang biasa dibutuhkan dalam pengembangan program dibarengi dengan antarmuka yang sederhana serta struktur logika untuk mengakses *library* tersebut. Dengan begitu dapat disimpulkan bahwa CodeIgniter membuat pemrogram fokus pada kreativitas pembuatan program dengan meminimalkan jumlah kode yang dituliskan.

##### 2.1.1 Flowchart Aplikasi CodeIgniter

Pada gambar 2.1 menunjukkan *flowchart* aliran data pada CodeIgniter:



Gambar 2.1: Flowchart CodeIgniter

Keterangan:

1. Index.php berfungsi sebagai pengontrol utama, yang menginisialisasikan sumber-sumber yang diperlukan untuk menjalankan CodeIgniter.
2. Router akan memeriksa permintaan HTTP untuk menentukan apa yang harus dilakukan selanjutnya
3. Jika terdapat *cache*, maka cache tersebut akan dikirim langsung ke browser dengan menjalankan sistem eksekusi normal.

4. HTTP *request* dan data yang diserahkan oleh *user* akan disaring oleh sistem keamanan terlebih dahulu oleh bagian keamanan(*security*) dari CodeIgniter yang dijalankan sebelum *controller* dari aplikasi diisi.
5. *Application Controller* akan mengambil isi dari *model*, *libraries*, *helpers*, *plugins*, *scripts*, dan sumber lain yang diperlukan untuk menjalankan perintah-perintah spesifik.
6. Kemudian *View* akan diterjemahkan dari *Application Controller* dan dikirim ke *web browser* untuk kemudian ditampilkan. Jika pada view final terdapat *file cache*, maka view tersebut akan terlebih dahulu dilakukan *cached* sehingga permintaan berikutnya dapat dilayani.

### 2.1.2 Model-View-Controller

CodeIgniter menggunakan dasar pola pengembangan *Model-View-Controller*(MVC). Pola pengembangan MVC ini merupakan suatu pendekatan yang memisahkan antara pengerjaan logika dan tampilan dari aplikasi.

MVC sendiri terdiri dari 3 bagian, yaitu:

1. *Model* merepresentasikan struktur data. Secara khusus, *model* merupakan kelas yang membantu menangani kueri-kueri sql seperti *insert*, *update*, dan *delete* pada basis data.
2. *View* merepresentasikan informasi yang ditunjukkan kepada pengguna. Sebuah *view* biasanya berbentuk *web page*, tetapi dalam CodeIgniter *view* bisa berbentuk *header*, *footer*, dan berbagai jenis *page* lainnya.
3. *Controller* berfungsi sebagai perantara antara *Model*, *View*, dan sumber daya lain yang diperlukan untuk memproses HTTP *request* dan menghasilkan halaman web.

### 2.1.3 Controller

*Controller* merupakan sebuah kelas simple dengan penerapan seperti URL. Seperti kelas pada umumnya, ketika nama kelas dari *controller* dan nama kelas dari *file controller* tersebut cocok, maka kelas dapat dijalankan dengan baik. Nama kelas suatu *controller* dikatakan sah jika diawali dengan huruf besar. Untuk lebih jelasnya, perhatikan gambar 2.2.

```
<?php
class Blog extends CI_Controller {

    public function index()
    {
        echo 'Hello World!';
    }

}
```

Gambar 2.2: Contoh Kode Controller

Nama *file* pada gambar 2.2 haruslah "Blog.php" dengan B besar dan disimpan pada *application/controllers* sehingga url dapat berjalan dengan baik.

## Method

*Method* merupakan nama fungsi dari suatu kelas. Nama *method* pada gambar 2.2 adalah *index()*. *Method* bernama "index" akan selalu dijalankan jika tidak ada arahan ke metode pada URL. Cara lain untuk menjalankan *method* pada gambar 2.2 adalah "example.com/index.php/blog/index/" dimana bagian terakhir adalah nama method yang ingin dijalankan.

```
<?php
class Products extends CI_Controller {

    public function shoes($sandals, $id)
    {
        echo $sandals;
        echo $id;
    }
}
```

Gambar 2.3: Contoh Method ber-Parameter

Jika *method* yang dituju memiliki parameter, diperlukan tambahan pada URL pemanggilannya. Sebagai contoh, pemanggilan *method* pada gambar 2.3 dilakukan dengan URL "example.com/index.php/product" dimana "sandals" dan "123" merupakan isi dari *parameter 1* dan 2 dari *method* "shoes".

## Mendefinisikan Controller Default

CodeIgniter dapat menjalankan *default controller* sehingga tidak diperlukannya penulisan URL yang lengkap untuk pemanggilan, melainkan *controller* dapat dipanggil secara otomatis dengan URL "example.com" saja. Namun, untuk dapat menjalankan fungsi ini, diperlukan sedikit pengaturan pada file "application/config/routes.php" yaitu perubahan variabel pada gambar 2.4.

```
$route['default_controller'] = 'blog';
```

Gambar 2.4: Penggantian Variable pada Route

Pada gambar 2.4, "blog" merupakan nama *file controller* yang telah dibuat pada direktori "application/controllers/". Setelah pengaturan tersebut, maka pengguna bisa menjalankan aplikasi tanpa URL yang terspesifikasi menjalankan *controller*.

### 2.1.4 Views

Sebuah *views* merupakan bagian yang mengatur tampilan aplikasi yang akan ditunjukkan kepada pengguna. *Views* meliputi *footer*, *header*, *sidebar*, dll. Pada CodeIgniter, *Views* tidak dapat dijalankan secara langsung dari URL, tapi *views* harus dijalankan melalui file *controller* yang ada. Hal ini dilakukan guna memudahkan *programmer* dan mewujudkan *framework MVC* pada CodeIgniter.

## Pembuatan Views

Pembuatan *file view* pada dasarnya sama seperti pembuatan *file* berbasis PHP biasa. Gambar 2.5 merupakan salah satu contoh *file view* sederhana.

```
<html>
<head>
    <title>My Blog</title>
</head>
<body>
    <h1>Welcome to my Blog!</h1>
</body>
</html>
```

Gambar 2.5: Contoh File View

Setelah selesai membuat *file view* yang diinginkan, maka penyimpanan *file* tersebut harus diletakkan di direktori "application/views/".

## Menjalankan View

Menjalankan *view* pada CodeIgniter dilakukan di *file controller*. Gambar 2.6 menunjukkan kode yang harus ditulis di dalam *method controller*.

```
<?php
class Blog extends CI_Controller {

    public function index()
    {
        $this->load->view('blogview');
    }

}
```

Gambar 2.6: Contoh Pemanggilan File View pada Controller

### 2.1.5 Models

*Model* merupakan *file* berbasis PHP yang didesain sebagai penghubung aplikasi dengan basis data. *Model* berfungsi menjalankan kueri-kueri sql seperti *insert*, *update*, *delete*, *select*, dll. Pada CodeIgniter terdapat fungsi *Query builder* yang memudahkan *programmer* dalam membuat kueri. Gambar 2.7 dan gambar 2.8 merupakan contoh penggunaan *Query builder* untuk kueri sql *insert* dan *update*.

```
public function insert_entry()
{
    $this->title    = $_POST['title']; // please read the below note
    $this->content  = $_POST['content'];
    $this->date     = time();

    $this->db->insert('entries', $this);
}
```

Gambar 2.7: Contoh Query Builder insert

```
public function update_entry()
{
    $this->title    = $_POST['title'];
    $this->content  = $_POST['content'];
    $this->date     = time();

    $this->db->update('entries', $this, array('id' => $_POST['id']));
}
```

Gambar 2.8: Contoh Query Builder Update

## Menjalankan Model

Sama seperti menjalankan *file view*, *model* pun tidak bisa dijalankan secara langsung menggunakan URL. Untuk menjalankan *model* perlu dilakukan pemanggilan pada *controller*.

```
class Blog_controller extends CI_Controller {

    public function blog()
    {
        $this->load->model('blog');

        $data['query'] = $this->blog->get_last_ten_entries();

        $this->load->view('blog', $data);
    }
}
```

Gambar 2.9: Contoh Pemanggilan File Model pada Controller

Gambar 2.9 menunjukkan bahwa *file controller* melakukan pemanggilan *model* yang diikuti dengan inisialisasi *array data* dari basis data yang dimasukkan ke pemanggilan *view*.

### 2.1.6 Helper

*Helper* merupakan kelas yang membantu *programmer* dalam menjalankan *task*. CodeIgniter memiliki banyak kelas *helper*, seperti *URL Helper* yang membantu dalam membuat *link*, *Form Helper*

yang membantu dalam pembuatan elemen-elemen di dalam form, *Text Helper* yang membantu dalam menjalankan berbagai *text formatting routines*, *Cookies Helper* yang membantu dalam mengatur dan membaca *cookies* yang ada, dll. *Helper* pada CodeIgniter umumnya ada pada direktori "application/helpers directory" atau "system/helpers".

## Menjalankan Helper

Cara menjalankan *helper* pada CodeIgniter cukup dengan menambahkan kode pada gambar 2.10 di dalam *kdoeHelper* atau *view*.

```
$this->load->helper('name');
```

Gambar 2.10: Kode yang ditambahkan untuk menjalankan helper

Penulisan "name" pada gambar 2.10 diisi dengan *part helper* yang diinginkan. Contoh jika pada aplikasi perlu *URL Helper* maka "name" diganti dengan "url". Helper juga dapat dijalankan secara otomatis dengan cara mengisi variable 'helper' pada *file autoload* yang berada di direktori "application/config/autoload.php".

### 2.1.7 Basis data

#### Menyambungkan ke Basis Data

Perlu diingat bahwa kelas *model* tidak menjalankan basis data secara otomatis. Untuk membuat aplikasi terkoneksi dengan basis data, diperlukan beberapa tambahan kode pada *file model* atau *file controller*. CodeIgniter memiliki fitur *automatically connecting* yang membuat seluruh aplikasi tersambung dengan basis data pada setiap *page load*. untuk mengaktifkan fitur ini cukup mengetikkan "database" pada variabel autoload['libraries'] di "application/config/autoload.php" seperti gambar 2.11.

```
$autoload['libraries'] = array('database');
```

Gambar 2.11: Kode yang ditambahkan untuk autoload basis data

Selain *autoload*, CodeIgniter juga mendukung koneksi ke basis data dengan cara manual, dengan cara menambahkan "\$this->load->database();" pada *method* atau kelas basis data ingin dijalankan.

### 2.1.8 Konfigurasi Basis Data

Konfigurasi basis data pada CodeIgniter disimpan dengan cara *multi-dimensional array*.

```
$db['default'] = array(
    'dsn' => '',
    'hostname' => 'localhost',
    'username' => 'root',
    'password' => '',
    'database' => 'database_name',
    'dbdriver' => 'mysqli',
    'dbprefix' => '',
    'pconnect' => TRUE,
    'db_debug' => TRUE,
    'cache_on' => FALSE,
    'cachedir' => '',
    'char_set' => 'utf8',
    'dbcollat' => 'utf8_general_ci',
    'swap_pre' => '',
    'encrypt' => FALSE,
    'compress' => FALSE,
    'stricton' => FALSE,
    'failover' => array()
);
```

Gambar 2.12: Konfigurasi Basis Data

Keterangan gambar [2.12](#):

Nama Konfigurasi	Deskripsi
dsn	membuat koneksi string( <i>an all-in-one configuration sequence</i> )
hostname	nama host dari server basis data yang dipakai.(umumnya bernama "localhost")
username	username yang dipakai untuk menyambungkan basis data
password	password yang cocok dengan username yang dipakai untuk menyambungkan basis data
database	nama basis data yang ingin di sambungkan
dbdriver	tipe basis data (mysql, postgre, odbc, dll). Perlu ditulis dengan huruf kecil secara spesifik.
dbprefix	dbprefix tidak harus terisi, berguna untuk menambahkan awalan nama tabel pada saat dijalankan Query Builder.
pconnect	berisi TRUE atau FALSE untuk perlunya koneksi yang tetap
db_debug	berisi TRUE atau FALSE untuk perlunya menampilkan error dari basis data
cache_on	berisi TRUE atau FALSE untuk diperbolehkannya database query caching
cachedir	server path yang mutlak untuk direktori database query cache
char_set	set karakter yang digunakan untuk komunikasi dengan basis data
dbcollat	pemeriksaan karakter yang digunakan dalam berkomunikasi dengan basis data(hanya dipakai di driver 'mysql' dan 'mysql').
swap_pre	sebuah tabel default yang harus bertukar dengan dbprefix.
schema	skema basis data yang nilai defaultnya adalah 'public'. Digunakan untuk driver PostgreSQL and ODBC.
encrypt	berisi TRUE atau FALSE perlu tidaknya memakai koneksi yang ter-enkripsi.
compress	perlu tidaknya memakai client compression (hanya untuk MYSQL)
stricton	berisi TRUE atau FALSE untuk perlu tidaknya memakai koneksi "Strict Mode"
port	nomor port dari basis data. Untuk menggunakannya diperlukan penambahan di config array database.

## 2.2 AngularJS

AngularJS[2] merupakan sebuah *framework* terstruktur yang digunakan untuk aplikasi web yang bersifat dinamis. Hal tersebut memungkinkan *programmer* untuk mempergunakan HTML sebagai template bahasa pemrograman dan memperluas sintaks HTML agar dapat mengekspresikan



komponen aplikasi dengan jelas dan ringkas. Sifat AngularJS yang mengikat data dan mempunyai ketergantungan injeksi akan menghilangkan banyak kode yang seharusnya dituliskan oleh *programmer*, dan semua itu terjadi pada *browser* sehingga dapat disimpulkan bahwa AngularJS merupakan pasangan yang sangat ideal bagi penggunaan teknologi server. Dalam pembuatannya, ketidakcocokkan halaman statik dan dinamik biasanya diselesaikan dengan pendekatan sebagai berikut:

1. *Library*: merupakan sebuah koleksi dari berbagai macam fungsi yang berguna dalam pembuatan aplikasi *web*, contoh: JQuery.
2. *Frameworks*: merupakan suatu implementasi dari sebuah aplikasi *web* yang menempatkan kode yang dituliskan secara detail. *Framework* akan berperan melakukan pemanggilan ke kode yang dituliskan *programmer* ketika aplikasi membutuhkan sesuatu yang spesifik, contoh: durandal, ember, dll.

Dalam pembentukannya, AngularJS memiliki pendekatan yang berbeda. AngularJS berupaya untuk meminimalkan ketidakcocokan antara dokumen utama dari HTML dengan apa yang dibutuhkan oleh aplikasi untuk membuat konstruksi HTML baru. AngularJS mengajarkan *browser* sintaks baru yang disebut *directives*. Contoh contoh *directives* adalah:

1. Keterikatan data di dalam `{{}}`;
2. Dukungan untuk *Form* dan *Form Validation*
3. Pengelompokkan HTML menjadi komponen - komponen yang dapat dipakai kembali.

### 2.2.1 Gambaran Konseptual

Berikut ini adalah beberapa bagian-bagian terpenting dalam AngularJS.

Konsep	Deskripsi
Template	HTML dengan tambahan markup
Directives	Pengembangan HTML dengan atribut dan elemen yang dibuat khusus
Model	Data yang ditunjukkan kepada pengguna pada tampilan dan bagaimana pengguna berinteraksi
Scope	Konteks dimana model disimpan, sehingga controller, directives dan expression dapat mengaksesnya
Expression	Mengakses variabel dan fungsi dari scope
Compiler	Menguraikan template, directives, dan expression
Filter	Mengatur nilai dari sebuah expression untuk di tunjukkan kepada pengguna
View	Apa yang akan dilihat oleh pengguna (DOM)
Data Binding	Menyelaraskan data yang ada pada <i>model</i> dan view
Controller	Mengatur logika dibalik tampilan
Dependency Injection	Membuat dan menyambungkan objek dan fungsi
Injector	Tempat penyimpanan dependency Injection
Module	Tempat penyimpanan untuk bagian-bagian yang berbeda dalam sebuah aplikasi, yang mencakup: controllers, services, filters, directives yang mengkonfigurasi injector
Services	Logika bisnis independen dari views yang bisa dipakai kembali

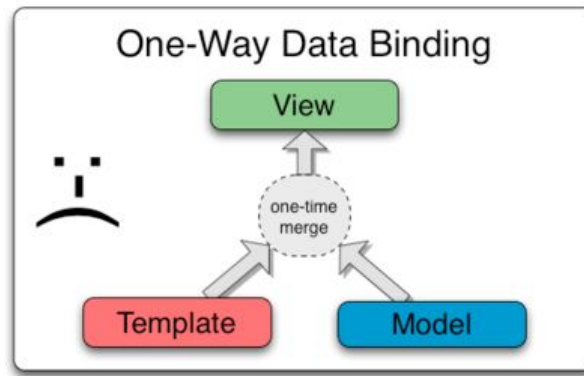
### 2.2.2 Directives

*Directives* merupakan penanda pada *DOM elements* (seperti atribut, nama elemen, *comment*, dan kelas CSS) yang memberitahukan kepada *AngularJS HTML compiler* untuk melampirkan perilaku yang diinginkan kepada *DOM element* (contohnya memakai *event listener*), atau bahkan mengubah *DOM element* yang dituju beserta dengan peranannya.

AngularJS menyediakan sekumpulan *directives built-in* seperti ng-Model, ng-Bind, dan ng-Class.

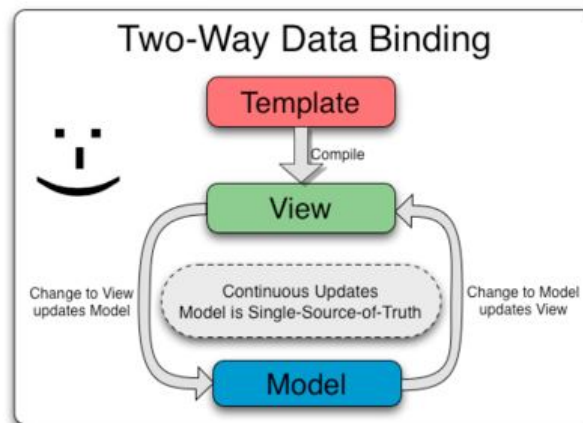
### 2.2.3 Data Binding

*Data Binding* pada AngularJS merupakan penyelarasan data antara *model* dan komponen - komponen *view*. Ketika *model* berubah, maka *view* pun akan berubah, begitu juga dengan sebaliknya.



Gambar 2.13: Data Binding Classical Templates System

Pada gambar 2.13 menjelaskan bahwa kebanyakan *data binding* adalah proses satu arah. Hal itu dilakukan dengan menyatukan *template* dan *model* menjadi *view*. Setelah penyatuan, pergantian pada *model* tidak secara otomatis mengganti *view* yang sudah ditampilkan.



Gambar 2.14: Data Binding pada Angular

Pada gambar 2.14 menjelaskan perbedaan yang diberikan oleh pelaksanaan *data binding* pada AngularJS. Pertama, *template* akan di *compile* pada browser. Hasil dari *compile* tersebut adalah *live view*. Pada tahap ini perubahan yang terjadi di *view* akan disampaikan kepada *model*, dan perubahan yang terjadi pada *model* akan mengubah *view*.

Karena *view* merupakan proyeksi dari *model*, menyebabkan *controller* benar-benar terpisahkan dari *view* tanpa disadari. Hal ini mempermudah pengujian *controller*, karena terisolasi tanpa adanya *view* dan DOM( *browser dependency*).

#### 2.2.4 Model-View-Controller(MVC)

AngularJS[3] juga merupakan salah satu *framework* yang menggunakan *Model-View-Controller* sebagai patokan desain aplikasi. Walaupun AngularJS mempunyai banyak fleksibilitas dalam membangun aplikasi, tetapi akan ada beberapa hal yang selalu dijumpai dalam mendesain sebuah aplikasi, diantaranya:

- Sebuah *model* selalu menampung data yang merepresentasikan keadaan aplikasi.

- *Views* yang menyajikan data tersebut.
- *Controller* yang akan selalu mengatur hubungan antara *model* dan *views*.

Model dibuat dengan menggunakan atribut berupa objek atau konten-konten primitif yang dapat menyimpan data. Berikut adalah salah satu contoh praktis dalam pembuatan *model*:

```
1 | var someText = 'You have started your journey.'
```

Setelah itu untuk menampilkannya maka perlu dibuat *view* dari data *model* "someText" diatas dengan cara:

```
1 | <p> {{someText}} </p>
```

*Syntax view* 2 kurung kurawal diatas disebut sebagai interlopasi(penyusupan), karena hal tersebut memasukkan konten baru ke dalam *template* yang sudah ada.

Sementara kelas *controllers* berguna untuk memberitahu AngularJS tentang objek atau konten primitif mana dari model yang akan dipakai dengan cara menetapkannya ke objek '\$scope', objek '\$scope' tersebut kemudian akan diberikan kepada *controller* seperti contoh berikut:

```
1 | function TextController($scope){
2 |     $scope.someText = someText;
3 | }
```

Berikut ini adalah contoh penggabungan fungsi *model*, *view*, dan *controller*:

```
1 | <html>
2 | <body ng-controller = "TextController">
3 |     <p>{{someText}}</p>
4 |
5 |     <script>
6 |         src="https://ajax.googleapis.com/ajax/libs/angularjs/1.0.4/angular.min.js">
7 |     </script>
8 |
9 |     <script>
10 |         function TextController($scope){
11 |             $scope.someText = 'You have started your journey.';
12 |         }
13 |     </script>
14 | </body>
15 | </html>
```

Hasil dari kode diatas adalah tulisan "You have started your journey". Walaupun cara ini dapat dilakukan dengan mudah pada aplikasi sederhana seperti contoh diatas, tetapi untuk kebanyakan aplikasi sebaiknya dibuat objek model untuk menyimpan *data* yang ada. Untuk itu, daripada membuat model seperti:

```
1 | function TextController($scope){
2 |     $scope.someText = someText;
3 | }
```

Lebih baik menggunakan kode:

```
1 | var message= {};
2 | message.someText = 'You have started your journey';
3 | function TextController($scope){
4 |     $scope.message = message;
5 | }
```

Yang kemudian akan dipanggil di *template* dengan kode:

```
1 | <p>{{message.someText}}</p>
```

Perubahan yang dilakukan diatas berfungsi untuk mencegah perilaku tidak terduga yang dapat terjadi dari *prototypal inheritance* dalam objek \$scope. Walaupun untuk sementara hal ini dapat berjalan dengan baik, tetapi cara yang benar dalam mendefinisikan sebuah *controller* adalah dengan menggunakan sebuah kelas yang dinamakan *module* yang menyediakan *namespace* untuk bagian lain dari aplikasi berhubungan. Perubahan tersebut akan mengubah kode-kode diatas menjadi:

```
1 <html ng-app='myApp'>
2 <body ng-controller='TextController'>
3   <p>{{someText.message}}</p>
4   <script>
5     src="https://ajax.googleapis.com/ajax/libs/angularjs/1.0.4/angular.min.js">
6   </script>
7
8   <script>
9     var myAppModule = angular.module('myApp',[]);
10
11     myAppModule.controller('TextController',
12       function($scope){
13         var someText = {};
14         someText.message = 'You have started your journey';
15         $scope.someText = someText;
16       });
17   </script>
18 </body>
19 </html>
```

Pada versi di atas, aplikasi memberi tahu elemen ng-app tentang nama dari modul yang dipakai di baris ke 9. Setelah itu pada baris ke 11 sampai 16 dilakukan pemanggilan objek Angular untuk membuat sebuah modul bernama myApp dan memberikan fungsi dari *controller* untuk memanggil fungsi controller dari modul.

## 2.3 Twitter Bootstrap

*Twitter Bootstrap*<sup>[4]</sup> atau yang lebih dikenal dengan *Bootstrap* adalah *framework* HTML, CSS, dan JS terpopuler dalam hal pengembangan tampilan yang responsif *mobile* pertama dalam hal aplikasi berbasis web.

### 2.3.1 Grid System

*Bootstrap* merupakan responsif *mobile* pertama yang mempunyai sistem skala (*grid system*). Sistem skala tersebut membagi layar perangkat menjadi 12 kolom yang berukuran sama, dimana besar ukuran masing-masing kolom mengikuti besar layar perangkat. Ketika layar semakin besar, maka ukuran masing-masing kolom pun akan semakin besar, begitu juga sebaliknya. Cara sistem skala *Bootstrap* bekerja adalah:

1. *Rows* harus ditempatkan diantara *.container(fixed-width)* atau *.container-fluid (full-width)* untuk mendapatkan keselarasan ukuran
2. *Rows* dipergunakan untuk membuat grup kolom secara *horizontal*.
3. Konten tampilan harus berada diantara kelas *columns* atau peranakan dari kelas *columns*.
4. Kelas-kelas yang telah ditetapkan seperti ".row" dan ".col-xs-4" dapat digunakan dengan segera untuk membentuk *layout*.
5. Kelas *columns* membuat *gutters* (jarak antara kolom konten) menggunakan kelas *padding*.
6. *Grid columns* dibuat dengan menyesuaikan ke-12 kolom yang sudah disediakan. Contohnya jika ingin membuat 3 kolom sama rata, maka diperlukan 3 buah kelas ".col-xs-4".
7. Jika ada lebih dari 12 kolom dalam 1 baris, maka kolom yang lebih tersebut akan dipindahkan ke baris baru sebagai satu kesatuan.

8. Kelas *grid* mempunyai fungsi untuk menyesuaikan ukuran sesuai dengan patokan ukuran yang sudah diberikan oleh *bootstrap* atau lebih besar dari angka patokan yang ada. Oleh karena itu ketika sebuah kelas ".col-md-\*" tidak memiliki kelas yang lebih besar darinya seperti kelas ".col-lg-\*", maka kelas md akan mengambil alih pada saat aplikasi dijalankan di ukuran perangkat yang lebih besar.

	Extra small devices Phones (<768px)	Small devices Tablets (≥768px)	Medium devices Desktops (≥992px)	Large devices Desktops (≥1200px)
<b>Grid behavior</b>	Horizontal at all times	Collapsed to start, horizontal above breakpoints		
<b>Container width</b>	None (auto)	750px	970px	1170px
<b>Class prefix</b>	.col-xs-	.col-sm-	.col-md-	.col-lg-
<b># of columns</b>	12			
<b>Column width</b>	Auto	~62px	~81px	~97px
<b>Gutter width</b>	30px (15px on each side of a column)			
<b>Nestable</b>	Yes			
<b>Offsets</b>	Yes			
<b>Column ordering</b>	Yes			

Gambar 2.15: Grid Option pada Bootstrap

.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1
.col-md-8								.col-md-4			
.col-md-4				.col-md-4				.col-md-4			
.col-md-6						.col-md-6					

Gambar 2.16: Contoh Pembagian Grid Columns

### 2.3.2 Form Class

Masing-masing form akan memiliki bentuk otomatis yang diatur secara global. Dengan memakai kelas ".form-control", pengaturan ukuran dari kelas <input>, <textarea>, dan <select> akan otomatis memiliki variabel *width* 100% secara *default*. Untuk mendapatkan jarak *spacing* yang maksimal, *Bootstrap* memiliki kelas ".form-group" yang membungkus kelas *form* menjadi grup-grup.

```
<form>
  <div class="form-group">
    <label for="exampleInputEmail1">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1" placeholder="Email">
  </div>
  <div class="form-group">
    <label for="exampleInputPassword1">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword1" placeholder="Password">
  </div>
  <div class="form-group">
    <label for="exampleInputFile">File input</label>
    <input type="file" id="exampleInputFile">
    <p class="help-block">Example block-level help text here.</p>
  </div>
  <div class="checkbox">
    <label>
      <input type="checkbox"> Check me out
    </label>
  </div>
  <button type="submit" class="btn btn-default">Submit</button>
</form>
```

Gambar 2.17: Contoh Penggunaan Kelas Form

## EXAMPLE

## Email address

## Password

## File input

 No file chosen

Example block-level help text here.

☐ Check me out

Gambar 2.18: Contoh Hasil Penggunaan Kelas Form





## BAB 3

### ANALISIS

#### 3.1 Analisis Sistem Kini

Analisis sistem kini akan menjelaskan bagaimana sistem penilaian sidang skripsi 2 yang telah ada dan dipakai pada saat ini di Universitas Katolik Parahyangan jurusan Teknik Informatika. Berikut ini adalah penjelasan penggunaan kertas *form* tersebut:

##### 3.1.1 Form Rekapitulasi Penilaian

*Form* rekapitulasi dibagi menjadi 3 bagian (Gambar A.2), yaitu:

1. Lembar Rekapitulasi Penilaian Pembimbing
2. Lembar Rekapitulasi Penilaian Ketua Tim Penguji
3. Lembar Rekapitulasi Penilaian Anggota Tim Penguji

Ketiga lembaran tersebut akan dipotong dan diberikan kepada pembimbing, ketua tim penguji, dan anggota tim penguji sesuai dengan keperluannya. Setelah dibagikan, penilai wajib mengisi kolom nilai yang ingin diberikan kepada mahasiswa yang bersangkutan sesuai dengan kolom komponen penilaian yang ada. Setelah penilai memberikan nilai, maka penilai harus melakukan perkalian antara kolom nilai dengan kolom bobot yang akan menghasilkan kolom nilai akhir mahasiswa. Terakhir penilai akan menjumlahkan seluruh kolom nilai akhir yang akan menghasilkan total nilai akhir dari mahasiswa tersebut.

Setelah semua kolom terisi, maka lembaran rekapitulasi tersebut akan dikumpulkan ke ketua tim penguji. Kemudian data yang telah tersedia akan disalin oleh ketua tim penguji kepada *form* berita acara sidang skripsi.

##### 3.1.2 Form Berita Acara Sidang Skripsi

*Form* berita acara sidang skripsi merupakan *form* yang mencakup pengisian waktu sidang bersangkutan, data diri mahasiswa, nama dosen penguji dan pembimbing, nilai akhir dari masing-masing penilai, dan nilai akhir yang diterima mahasiswa. Seperti yang telah dibahas pada subbab sebelumnya (3.1.1) *form* berita acara sidang skripsi akan diisi oleh ketua tim penguji setelah seluruh form rekapitulasi dari masing-masing penilai di kumpulkan kembali kepada ketua tim penguji.

Setelah ketua tim penguji melakukan pengisian pada masing-masing kolom nilai dari masing-masing penilai yang bersangkutan, maka ketua tim penguji akan melakukan perkalian nilai tersebut

dengan bobot masing-masing penilai yang akan menghasilkan nilai akhir mahasiswa dari masing-masing penilai. Kemudian akan dihasilkan 90% nilai akhir mahasiswa untuk diberitahukan kepada mahasiswa dan diberikan kepada koordinator skripsi untuk melengkapi 10% dari nilai mahasiswa berdasarkan nilai kedisiplinan. Hasil dari seluruh proses tersebut adalah nilai akhir sidang skripsi 2 mahasiswa bersangkutan.

## 3.2 Analisis Sistem Usulan

Analisis sistem usulan dibagi menjadi beberapa tahap, yaitu analisis *back end*, analisis *front end*, dan analisis basis data. Berikut ini penjelasannya:

### 3.2.1 Analisis Back End

Analisis tahap *back end* merupakan analisis pada lapisan data akses dan kode-kode yang bekerja secara tidak terlihat pada suatu aplikasi. Pada sistem informasi penilaian sidang skripsi 2, analisis tahap ini membahas tentang pembuatan kode *model*, *view*, *controller* dari *codeigniter*. Berikut ini adalah penjelasan lengkapnya:

#### Model

Pada bagian ini akan dijelaskan tentang penggunaan *model* pada *codeigniter*. *Model* mempunyai fungsi untuk membuat sambungan dari aplikasi ke basis data. Pada *codeigniter* pemanggilan *model* dilakukan pada file *controller* dengan menggunakan fungsi khusus *codeigniter* yaitu:

```
1 | $data = $this->skripsi_model->getAllMahasiswa();
```

Kode di atas merupakan fungsi dari *codeigniter* yang melakukan pemanggilan terhadap *file model* yang akan dipakai. Pada kasus sistem usulan, nama *file model* yang digunakan adalah "skripsi\_model". *Model* sendiri berisi kode-kode sebagai berikut:

```
1 | <?php
2 | defined('BASEPATH') OR exit('No direct script access allowed');
3 |
4 | class Skripsi_model extends CI_Model {
5 |
6 |     public function insertDataMahasiswa($tableName, $data){
7 |         $res = $this->db->insert($tableName, $data);
8 |     }
9 | }
```

Berikut adalah *method* yang dimiliki oleh kelas *model*:

- public function insertDataMahasiswa(\$tablename, \$data)  
 Berfungsi untuk melakukan fungsi *insert* pada basis data.  
 Parameter:
  - tablename merepresentasikan nama tabel basis data.
  - data merepresentasikan data dari controller yang sudah diubah dan ingin dimasukkan kedalam basis data.

## Controller

Pada bagian ini akan dijelaskan tentang kode dan kegunaannya pada kelas *controller*. *Controller* merupakan kelas yang mengatur hubungan antara kelas *model* dan *view* pada *codeigniter*. Dengan memanfaatkan fungsi-fungsi dari *codeigniter*, maka kelas *controller* dapat dipersingkat dan dipermudah dalam pembuatannya. Berikut ini adalah kode pada kelas *C\_Skripsi*:

```

1  <?php
2  defined('BASEPATH') OR exit('No direct script access allowed');
3
4  class C_skripsi extends CI_Controller {
5
6  /**
7   * Index Page for this controller.
8   *
9   * Maps to the following URL
10   *      http://example.com/index.php/welcome
11   * - or -
12   *      http://example.com/index.php/welcome/index
13   * - or -
14   * Since this controller is set as the default controller in
15   * config/routes.php, it's displayed at http://example.com/
16   *
17   * So any other public methods not prefixed with an underscore will
18   * map to /index.php/welcome/<method_name>
19   * @see https://codeigniter.com/user_guide/general/urls.html
20   */
21   public function index()
22   {
23       $this->load->view('skripsi');
24   }
25   //Check database
26   public function view_cekMahasiswa() {
27       $data = $this->skripsi_model->getAllMahasiswa();
28       $this->load->view('cek_mahasiswa', array('data' => $data));
29   }
30
31   public function tambahDataMahasiswa() {
32       $semester = $_POST['semester'];
33       $tahun = $_POST['tahun'];
34       $npm = $_POST['npm'];
35       $nama = $_POST['nama'];
36       $judul = $_POST['judul'];
37       $namaPembimbing = $_POST['namaPembimbing'];
38       $namaPembimbingPendamping = $_POST['namaPembimbingPendamping'];
39       $namaKetuaTimPenguji = $_POST['namaKetuaTimPenguji'];
40       $namaAnggotaTimPenguji = $_POST['namaAnggotaTimPenguji'];
41       $bobotKetuaTimPenguji = $_POST['bobotKetuaTimPenguji'];
42       $bobotAnggotaTimPenguji = $_POST['bobotAnggotaTimPenguji'];
43       $bobotPembimbing = $_POST['bobotPembimbing'];
44       $nilaiKoordinatorSkripsi = $_POST['nilaiKoordinatorSkripsi'];
45       $bobotKoordinatorSkripsi = $_POST['bobotKoordinatorSkripsi'];
46       $bobotTataTulisLaporanAnggota = $_POST['bobotTataTulisLaporanAnggota'];
47       $bobotKelengkapanMateriAnggota = $_POST['bobotKelengkapanMateriAnggota'];
48       $bobotPenguasaanMateriAnggota = $_POST['bobotPenguasaanMateriAnggota'];
49       $bobotPresentasiAnggota = $_POST['bobotPresentasiAnggota'];
50       $bobotPencapaianTujuanAnggota = $_POST['bobotPencapaianTujuanAnggota'];
51       $bobotTataTulisLaporanKetua = $_POST['bobotTataTulisLaporanKetua'];
52       $bobotKelengkapanMateriKetua = $_POST['bobotKelengkapanMateriKetua'];
53       $bobotPenguasaanMateriKetua = $_POST['bobotPenguasaanMateriKetua'];
54       $bobotPresentasiKetua = $_POST['bobotPresentasiKetua'];
55       $bobotPencapaianTujuanKetua = $_POST['bobotPencapaianTujuanKetua'];
56       $bobotTataTulisLaporanPembimbing = $_POST['bobotTataTulisLaporanPembimbing'];
57       $bobotKelengkapanMateriPembimbing = $_POST['bobotKelengkapanMateriPembimbing'];
58       $bobotPenguasaanMateriPembimbing = $_POST['bobotPenguasaanMateriPembimbing'];
59       $prosesBimbinganPembimbing = $_POST['prosesBimbinganPembimbing'];
60       $nilaiAkhirMahasiswa = $_POST['nilaiAkhirMahasiswa'];
61       $data_insert = array(
62           'semester' => $semester,
63           'tahun' => $tahun,
64           'npm' => $npm,
65           'nama' => $nama,
66           'judul' => $judul,
67           'namaPembimbing' => $namaPembimbing,
68           'namaPembimbingPendamping' => $namaPembimbingPendamping,
69           'namaKetuaTimPenguji' => $namaKetuaTimPenguji,
70           'namaAnggotaTimPenguji' => $namaAnggotaTimPenguji,
71           'bobotKetuaTimPenguji' => $bobotKetuaTimPenguji,
72           'bobotAnggotaTimPenguji' => $bobotAnggotaTimPenguji,

```

```

73         'bobotPembimbing' => $bobotPembimbing,
74         'nilaiKoordinatorSkripsi' => $nilaiKoordinatorSkripsi,
75         'bobotKoordinatorSkripsi' => $bobotKoordinatorSkripsi,
76         'bobotTataTulisLaporanAnggota' => $bobotTataTulisLaporanAnggota,
77         'bobotKelengkapanMateriAnggota' => $bobotKelengkapanMateriAnggota,
78         'bobotPenguasaanMateriAnggota' => $bobotPenguasaanMateriAnggota,
79         'bobotPresentasiAnggota' => $bobotPresentasiAnggota,
80         'bobotPencapaianTujuanAnggota' => $bobotPencapaianTujuanAnggota,
81         'bobotTataTulisLaporanKetua' => $bobotTataTulisLaporanKetua,
82         'bobotKelengkapanMateriKetua' => $bobotKelengkapanMateriKetua,
83         'bobotPenguasaanMateriKetua' => $bobotPenguasaanMateriKetua,
84         'bobotPresentasiKetua' => $bobotPresentasiKetua,
85         'bobotPencapaianTujuanKetua' => $bobotPencapaianTujuanKetua,
86         'bobotTataTulisLaporanPembimbing' => $bobotTataTulisLaporanPembimbing,
87         'bobotKelengkapanMateriPembimbing' => $bobotKelengkapanMateriPembimbing,
88         'bobotPenguasaanMateriPembimbing' => $bobotPenguasaanMateriPembimbing,
89         'prosesBimbinganPembimbing' => $prosesBimbinganPembimbing,
90         'nilaiAkhirmahasiswa' => $nilaiAkhirmahasiswa,
91     );
92     $res = $this->skripsi_model->insertDataMahasiswa('beritaacarasidangskripsi',$data_insert);
93     redirect(base_url(), 'refresh');
94 }
95
96
97 }

```

Berikut adalah *method-method* yang dimiliki oleh kelas *controller*:

- view\_cekMahasiswa

Berfungsi untuk memilih *file view* dan *model* yang akan dipakai pada sistem informasi.

- tambahDataMahasiswa

Berfungsi untuk mengambil data yang telah terisi dari *view* sistem dan mengubahnya menjadi *compatible* sehingga dapat diproses kedalam *method* insertDataMahasiswa pada kelas *model* yang kemudian akan diproses ke dalam bahasa sql.

### 3.2.2 Analisis Front End

Pada subbab ini akan dijelaskan bagaimana pembuatan dan fungsi otomatisasi dari AngularJS di sistem usulan. Berikut ini adalah contoh proses otomatisasi pada sistem usulan:

```

1 <body ng-app="penilaian">
2   <form ng-controller="DefaultValue">
3
4   </form>
5
6   <script>
7     angular.module('penilaian', [])
8     .controller('DefaultValue', ['$scope', function ($scope) {
9
10    }]);
11  </script>
12 </body>

```

Contoh diatas adalah inisialisasi dari AngularJS dengan menggunakan fungsi ng-app dan ng-controller yang mengatur keseluruhan fungsi otomatisasi pada sistem usulan. Pada baris pertama dilakukan inisialisasi ng-app yang berfungsi menginisialisasi nama app yang digunakan pada sistem. Setelah ng-app diinisialisasi, baru sistem usulan dapat menggunakan fungsi-fungsi AngularJS seperti menginisialisasi ng-controller pada baris ke-2 dengan nama "Default Value".

Agar *controller* dapat berfungsi, perlu dilakukan pemanggilan terhadap ng-controller dengan memanfaatkan fungsi "angular.module". Baris ke-7 bekerja dengan *parameter* ng-app("penilaian"). Selanjutnya diikuti dengan sebuah *array*(

) kosong yang merupakan tempat yang menunjukkan *list modules* diperlukan oleh ng-app('penilaian') Setelah melakukan inisialisasi modul, maka kita dapat memanggil fungsi-fungsi daripada AngularJS untuk dijalankan, seperti *controller("DefaultValue")* ke dalam aplikasi AngularJS.

```

1 <tr>
2   <td><label for="nTTLaporanK">Tata Tulis Laporan</label></td>
3   <td><input type="number" id="nTTLaporanK" max="100" ng-model="nilai_TTLaporanK" class="form-nilai"/></td>
4   <!-- 20 -->
5   <td><input type="number" name="bobotTataTulisLaporanKetua" ng-model="TTLaporanK.value" ng-init="
      TTLaporanK.value = 15" min="0" max="100" class="form-nilai" readonly="readonly" /></td>
6   <td><input type="number" disabled="disabled" value="{nilai_TTLaporanK * TTLaporanK.value / 100}" ng-
      model="total_TTLaporanK" class="form-nilai"/></td>
7 </tr>

```

Contoh diatas diambil dari kode *file view* untuk mengatur otomatisasi pada kolom tata tulis laporan milik ketua tim penguji. Pada baris ke-3 dan baris ke-5 adalah contoh kode diatas merupakan contoh inisialisasi fungsi ng-model dari AngularJS, sementara baris ke-6 dilakukan perhitungan otomatis dari ng-model baris ke-3 dikalikan dengan ng-model baris ke-5 dan hasilnya ditampung di nilai *value* yang kemudian akan muncul ke layar *user* secara otomatis.

Berikut ini adalah nama-nama dari *model*, *view*, dan *controller* AngularJS yang dipakai pada sistem penilaian sidang skripsi 2:

- *Controller*: "DevaultValue" = Controller yang dipakai di seluruh sistem
- *Model*:
  - "tahun" = untuk melakukan otomatisasi tahun+1
  - "n\_npm" = untuk melakukan pengisian otomatis npm mahasiswa pada semua lembaran penilaian
  - "nilai\_ketua" = untuk menyimpan hasil perolehan total nilai dari lembar rekapitulasi ketua tim penguji
  - "ketua.value" = untuk menyimpan bobot ketua tim penguji terhadap nilai akhir mahasiswa
  - "total\_ketua" = untuk menyimpan perolehan nilai akhir dari ketua tim penguji
  - "nilai\_anggota" = untuk menyimpan hasil perolehan total nilai dari lembar rekapitulasi anggota tim penguji
  - "anggota.value" = untuk menyimpan bobot anggota tim penguji terhadap nilai akhir mahasiswa
  - "total\_anggota" = untuk menyimpan perolehan nilai akhir dari anggota tim penguji
  - "nilai\_pembimbing" = untuk menyimpan hasil perolehan total nilai dari lembar rekapitulasi pembimbing
  - "pembimbing.value" = untuk menyimpan bobot pembimbing terhadap nilai akhir mahasiswa
  - "total\_pembimbing" = untuk menyimpan perolehan nilai akhir dari pembimbing
  - "nilai\_koordinator" = untuk menyimpan hasil perolehan total nilai dari lembar rekapitulasi koordinator

- "koodinator.value" = untuk menyimpan bobot koordinator terhadap nilai akhir mahasiswa
- "total\_koordinator" = untuk menyimpan perolehan nilai akhir dari koordinator
- "nilai\_TTLaporanK" = untuk menyimpan nilai tata tulis laporan pada lembar rekapitulasi ketua tim penguji
- "TTLaporanK.value" = untuk menyimpan bobot nilai tata tulis laporan pada lembar rekapitulasi ketua tim penguji
- "total\_TTLaporanK" = untuk menyimpan nilai akhir tata tulis laporan pada lembar rekapitulasi ketua tim penguji
- "nilai\_KMateriK" = untuk menyimpan nilai kelengkapan materi pada lembar rekapitulasi ketua tim penguji
- "KMateriK.value" = untuk menyimpan bobot nilai kelengkapan materi pada lembar rekapitulasi ketua tim penguji
- "total\_KMateriK" = untuk menyimpan bobot nilai akhir kelengkapan materi pada lembar rekapitulasi ketua tim penguji
- "nilai\_PMateriK" = untuk menyimpan nilai penguasaan materi pada lembar rekapitulasi ketua tim penguji
- "PMateriK.value" = untuk menyimpan bobot nilai penguasaan materi pada lembar rekapitulasi ketua tim penguji
- "total\_PMateriK" = untuk menyimpan bobot nilai akhir penguasaan materi pada lembar rekapitulasi ketua tim penguji
- "nilai\_presentasiK" = untuk menyimpan nilai presentasi pada lembar rekapitulasi ketua tim penguji
- "presentasiK.value" = untuk menyimpan bobot nilai presentasi pada lembar rekapitulasi ketua tim penguji
- "total\_presentasiK" = untuk menyimpan bobot nilai akhir presentasi pada lembar rekapitulasi ketua tim penguji
- "nilai\_PMateriK" = untuk menyimpan nilai penguasaan materi pada lembar rekapitulasi ketua tim penguji
- "PMateriK.value" = untuk menyimpan bobot nilai penguasaan materi pada lembar rekapitulasi ketua tim penguji
- "total\_PMateriK" = untuk menyimpan bobot nilai akhir penguasaan materi pada lembar rekapitulasi ketua tim penguji
- "nTotalKetua" = untuk menyimpan perhitungan nilai keseluruhan ketua tim penguji yang akan dimasukkan ke lembar berita acara sidang skripsi
- "nilai\_TTLaporanA" = untuk menyimpan nilai tata tulis laporan pada lembar rekapitulasi anggota tim penguji
- "TTLaporanA.value" = untuk menyimpan bobot nilai tata tulis laporan pada lembar rekapitulasi anggota tim penguji

- "total\_TTLaporanA" = untuk menyimpan nilai akhir tata tulis laporan pada lembar rekapitulasi anggota tim penguji
- "nilai\_KMateriA" = untuk menyimpan nilai kelengkapan materi pada lembar rekapitulasi anggota tim penguji
- "KMateriA.value" = untuk menyimpan bobot nilai kelengkapan materi pada lembar rekapitulasi anggota tim penguji
- "total\_KMateriA" = untuk menyimpan bobot nilai akhir kelengkapan materi pada lembar rekapitulasi anggota tim penguji
- "nilai\_PMateriA" = untuk menyimpan nilai penguasaan materi pada lembar rekapitulasi anggota tim penguji
- "PMateriA.value" = untuk menyimpan bobot nilai penguasaan materi pada lembar rekapitulasi anggota tim penguji
- "total\_PMateriA" = untuk menyimpan bobot nilai akhir penguasaan materi pada lembar rekapitulasi anggota tim penguji
- "nilai\_presentasiA" = untuk menyimpan nilai presentasi pada lembar rekapitulasi anggota tim penguji
- "presentasiA.value" = untuk menyimpan bobot nilai presentasi pada lembar rekapitulasi anggota tim penguji
- "total\_presentasiA" = untuk menyimpan bobot nilai akhir presentasi pada lembar rekapitulasi anggota tim penguji
- "nilai\_PMateriA" = untuk menyimpan nilai penguasaan materi pada lembar rekapitulasi anggota tim penguji
- "PMateriA.value" = untuk menyimpan bobot nilai penguasaan materi pada lembar rekapitulasi anggota tim penguji
- "total\_PMateriA" = untuk menyimpan bobot nilai akhir penguasaan materi pada lembar rekapitulasi anggota tim penguji
- "nTotalAnggota" = untuk menyimpan perhitungan nilai keseluruhan anggota tim penguji yang akan dimasukkan ke lembar berita acara sidang skripsi
- "nilai\_TTLaporanP" = untuk menyimpan nilai tata tulis laporan pada lembar rekapitulasi pembimbing tim penguji
- "TTLaporanP.value" = untuk menyimpan bobot nilai tata tulis laporan pada lembar rekapitulasi pembimbing
- "total\_TTLaporanP" = untuk menyimpan nilai akhir tata tulis laporan pada lembar rekapitulasi pembimbing
- "nilai\_KMateriP" = untuk menyimpan nilai kelengkapan materi pada lembar rekapitulasi pembimbing
- "KMateriP.value" = untuk menyimpan bobot nilai kelengkapan materi pada lembar rekapitulasi pembimbing

- "total\_KMateriP" = untuk menyimpan bobot nilai akhir kelengkapan materi pada lembar rekapitulasi pembimbing
- "nilai\_PMateriP" = untuk menyimpan nilai penguasaan materi pada lembar rekapitulasi pembimbing
- "PMateriP.value" = untuk menyimpan bobot nilai penguasaan materi pada lembar rekapitulasi pembimbing
- "total\_PMateriP" = untuk menyimpan bobot nilai akhir penguasaan materi pada lembar rekapitulasi pembimbing
- "nilai\_PBimbinganP" = untuk menyimpan nilai proses bimbingan pada lembar rekapitulasi pembimbing
- "PBimbinganP.value" = untuk menyimpan bobot nilai proses bimbingan pada lembar rekapitulasi pembimbing
- "total\_PBimbinganP" = untuk menyimpan bobot nilai akhir proses bimbingan pada lembar rekapitulasi pembimbing
- "nTotalPembimbing" = untuk menyimpan perhitungan nilai keseluruhan pembimbing yang akan dimasukkan ke lembar berita acara sidang skripsi

• View:

- "{tahun+1}" = menampilkan hasil dari model "tahun" ditambahkan dengan 1
- "{ n\_npm }" = menampilkan npm mahasiswa
- "{nilai\_TTLaporanK \* TTLaporanK.value / 100 + nilai\_KMateriK \* KMateriK.value / 100 + nilai\_PMateriK \* PMateriK.value / 100 + nilai\_PresentasiK \* presentasiK.value / 100 + nilai\_PTujuanK \* PTujuanK.value / 100}" = nilai ketua tim penguji pada lembar berita acara sidang skripsi
- "{(nilai\_TTLaporanK \* TTLaporanK.value / 100 + nilai\_KMateriK \* KMateriK.value / 100 + nilai\_PMateriK \* PMateriK.value / 100 + nilai\_PresentasiK \* presentasiK.value / 100 + nilai\_PTujuanK \* PTujuanK.value / 100) \* ketua.value / 100}" = menampilkan perhitungan nilai ketua dikalikan dengan bobot ketua tim penguji pada lembar berita acara sidang skripsi
- "{nilai\_TTLaporanA \* TTLaporanA.value / 100 + nilai\_KMateriA \* KMateriA.value / 100 + nilai\_PMateriA \* PMateriA.value / 100 + nilai\_PresentasiA \* presentasiA.value / 100 + nilai\_PTujuanA \* PTujuanA.value / 100}" = nilai anggota tim penguji pada lembar berita acara sidang skripsi
- "{(nilai\_TTLaporanA \* TTLaporanA.value / 100 + nilai\_KMateriA \* KMateriA.value / 100 + nilai\_PMateriA \* PMateriA.value / 100 + nilai\_PresentasiA \* presentasiA.value / 100 + nilai\_PTujuanA \* PTujuanA.value / 100) \* anggota.value / 100}" = nilai anggota dikalikan dengan bobot anggota tim penguji pada lembar berita acara sidang skripsi
- "{nilai\_TTLaporanP \* TTLaporanP.value / 100 + nilai\_KMateriP \* KMateriP.value / 100 + nilai\_PMateriP \* PMateriP.value / 100 + nilai\_PBimbinganP \* PBimbinganP.value / 100}" = nilai pembimbing pada lembar berita acara sidang skripsi



- " $\{(nilai\_TTLaporanP * TTLaporanP.value / 100 + nilai\_KMateriP * KMateriP.value / 100 + nilai\_PMateriP * PMateriP.value / 100 + nilai\_PBimbinganP * PBimbinganP.value / 100) * pembimbing.value / 100\}$ " = nilai pembimbing dikalikan bobot pembimbing pada lembar berita acara sidang skripsi
- " $\{nilai\_koordinator * koordinator.value / 100\}$ " = nilai koordinator skripsi dikalikan dengan bobot koordinator skripsi pada lembar berita acara sidang skripsi
- " $\{ketua.value + anggota.value + pembimbing.value + koordinator.value\}$ " = menampilkan total bobot pada lembar berita acara sidang skripsi
- " $\{(nilai\_TTLaporanK * TTLaporanK.value / 100 + nilai\_KMateriK * KMateriK.value / 100 + nilai\_PMateriK * PMateriK.value / 100 + nilai\_PresentasiK * presentasiK.value / 100 + nilai\_PTujuanK * PTujuanK.value / 100) * ketua.value / 100 + (nilai\_TTLaporanA * TTLaporanA.value / 100 + nilai\_KMateriA * KMateriA.value / 100 + nilai\_PMateriA * PMateriA.value / 100 + nilai\_PresentasiA * presentasiA.value / 100 + nilai\_PTujuanA * PTujuanA.value / 100) * anggota.value / 100 + (nilai\_TTLaporanP * TTLaporanP.value / 100 + nilai\_KMateriP * KMateriP.value / 100 + nilai\_PMateriP * PMateriP.value / 100 + nilai\_PBimbinganP * PBimbinganP.value / 100) * pembimbing.value / 100 + nilai\_koordinator * koordinator.value / 100\}$ " = nilai akhir mahasiswa
- " $\{nilai\_TTLaporanK * TTLaporanK.value / 100\}$ " = nilai tata tulis laporan ketua tim penguji
- " $\{nilai\_KMateriK * KMateriK.value / 100\}$ " = nilai kelengkapan materi ketua tim penguji
- " $\{nilai\_PMateriK * PMateriK.value / 100\}$ " = nilai penguasaan materi ketua tim penguji
- " $\{nilai\_PresentasiK * presentasiK.value / 100\}$ " = nilai presentasi ketua tim penguji
- " $\{nilai\_PTujuanK * PTujuanK.value / 100\}$ " = nilai pencapaian tujuan ketua tim penguji
- " $\{nilai\_TTLaporanK * TTLaporanK.value / 100 + nilai\_KMateriK * KMateriK.value / 100 + nilai\_PMateriK * PMateriK.value / 100 + nilai\_PresentasiK * presentasiK.value / 100 + nilai\_PTujuanK * PTujuanK.value / 100\}$ " = total nilai ketua tim penguji
- " $\{nilai\_TTLaporanA * TTLaporanA.value / 100\}$ " = nilai tata tulis laporan anggota tim penguji
- " $\{nilai\_KMateriA * KMateriA.value / 100\}$ " = nilai kelengkapan materi anggota tim penguji
- " $\{nilai\_PMateriA * PMateriA.value / 100\}$ " = nilai penguasaan materi ketua tim penguji
- " $\{nilai\_PresentasiA * presentasiA.value / 100\}$ " = nilai presentasi anggota tim penguji
- " $\{nilai\_PTujuanA * PTujuanA.value / 100\}$ " = nilai pencapaian tujuan anggota tim penguji

- " $\{\{\text{nilai\_TTLaporanA} * \text{TTLaporanA.value} / 100 + \text{nilai\_KMateriA} * \text{KMateriA.value} / 100 + \text{nilai\_PMateriA} * \text{PMateriA.value} / 100 + \text{nilai\_PresentasiA} * \text{presentasiA.value} / 100 + \text{nilai\_PTujuanA} * \text{PTujuanA.value} / 100\}\}$ " = total nilai anggota tim penguji
- " $\{\{\text{nilai\_TTLaporanP} * \text{TTLaporanP.value} / 100\}\}$ " = nilai tata tulis laporan pembimbing
- " $\{\{\text{nilai\_KMateriP} * \text{KMateriP.value} / 100\}\}$ " = nilai kelengkapan materi pembimbing
- " $\{\{\text{nilai\_PMateriP} * \text{PMateriP.value} / 100\}\}$ " = nilai penguasaan materi pembimbing
- " $\{\{\text{nilai\_PBimbinganP} * \text{PBimbinganP.value} / 100\}\}$ " = nilai proses bimbingan pembimbing
- " $\{\{\text{nilai\_TTLaporanP} * \text{TTLaporanP.value} / 100 + \text{nilai\_KMateriP} * \text{KMateriP.value} / 100 + \text{nilai\_PMateriP} * \text{PMateriP.value} / 100 + \text{nilai\_PBimbinganP} * \text{PBimbinganP.value} / 100\}\}$ " = nilai akhir pembimbing

### 3.2.3 Analisis Basis Data

Sistem informasi penilaian sidang skripsi 2 menggunakan perangkat lunak mysql sebagai sarana penyimpanan dan pengolahan basis data. Seperti yang dijelaskan pada subbab ??, didalam *folder* "application" terdapat *folder* "models" (Gambar??) yang berfungsi menghubungkan basis data dengan sistem.

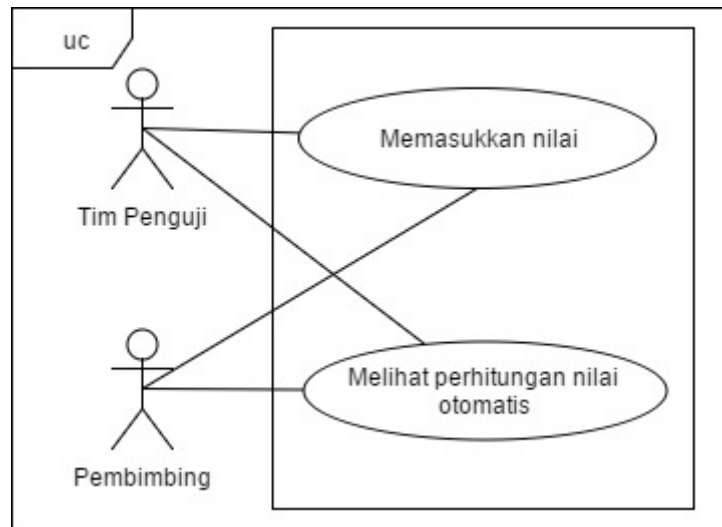
Berdasarkan analisa dari contoh form penilaian skripsi yang ada (gambar A.1 dan gambar A.2), dapat disimpulkan bahwa penilaian skripsi membutuhkan data-data sebagai berikut:

1. Semester
2. Tahun ajaran
3. NPM mahasiswa
4. Nama mahasiswa
5. Judul skripsi
6. Nama pembimbing utama/tunggal
7. Nama pembimbing pendamping(tidak harus)
8. Nama ketua tim penguji
9. Nama anggota tim penguji
10. Bobot ketua tim penguji
11. Bobot anggota tim penguji
12. Bobot pembimbing
13. Nilai koordinator skripsi
14. Bobot koordinator skripsi

15. Bobot tata tulis laporan ketua
16. Bobot kelengkapan materi ketua
17. Bobot penguasaan materi ketua
18. Bobot presentasi ketua
19. Bobot pencapaian tujuan ketua
20. Bobot tata tulis laporan anggota
21. Bobot kelengkapan materi anggota
22. Bobot penguasaan materi anggota
23. Bobot presentasi anggota
24. Bobot pencapaian tujuan anggota
25. Bobot tata tulis laporan pembimbing
26. Bobot kelengkapan materi pembimbing
27. Bobot penguasaan materi pembimbing
28. Bobot bimbingan pembimbing
29. Nilai akhir mahasiswa

Berdasarkan diskusi dengan dosen pembimbing, disimpulkan bahwa sistem penilaian sidang skripsi 2 ini hanya memerlukan penyimpanan untuk bobot masing-masing penilaian dan nilai akhir mahasiswa untuk tahap perhitungan. Hal ini dikarenakan nilai-nilai lainnya dapat dihasilkan dengan melakukan perhitungan pada nilai akhir mahasiswa dan bobot nilai yang diinginkan. Begitu pula dengan nilai dari masing-masing penguji.

### 3.3 Use Case



Gambar 3.1: Use case diagram

1. Skenario memasukkan nilai

Deskripsi: Kegiatan memasukkan nilai ke dalam kotak input yang ada.

Aktor: Pengguna

Prakondisi: -

Skenario:

- Pengguna memilih tempat/kolom yang sudah tersedia di tampilan
- Pengguna memasukkan nilai yang diinginkan pada tempat/kolom yang telah dipilih.

2. Skenario melakukan perhitungan otomatis

Deskripsi: Kegiatan melakukan perhitungan secara otomatis pada tampilan

Aktor: Sistem

Prakondisi: Tempat atau kolom nilai yang ingin dihitung sudah terisi

Skenario:

- Pengguna mengisi kolom nilai yang sudah disediakan
- Dengan ng-model, sistem mengambil nilai dari tempat/kolom yang sudah diisi dan melakukan perhitungan
- Sistem menampilkan hasil perhitungan ke dalam kolom yang disediakan untuk hasil perhitungan.

## BAB 4

### PERANCANGAN

Pada bab ini akan dijelaskan mengenai perancangan aplikasi yang dibangun meliputi perancangan kelas, *routes*, *controllers*, *models*, perancangan antarmuka.

#### 4.1 Perancangan Kelas

Seperti yang sudah di jelaskan pada bab sebelumnya, untuk memodelkan sistem penilaian sidang skripsi 2 dengan menggunakan *codeigniter* membutuhkan *routes*, *controllers*, *models*, dan *views*. Hal-hal berikut akan dijelaskan pada subbab selanjutnya.

#### 4.2 Routes

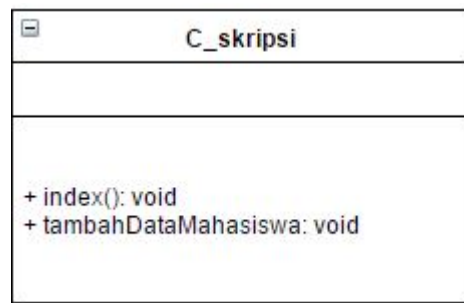
*Routes* merupakan bagian dari *codeigniter* untuk melakukan pemetaan terhadap lokasi *file controllers* dari aplikasi. Berikut adalah isi dari "config/routes":

```
1 | $route['default_controller'] = 'C_skripsi';  
2 | $route['404_override'] = "";  
3 | $route['translate_url_dahses'] = FALSE;
```

Baris pertama dari kode di atas adalah nama *file controller* yang terletak di *folder controllers* yang akan diambil. Baris kedua merupakan kode untuk menangani *error* yang terjadi jika *file* yang dicari tidak ditemukan, contoh penggunaanya adalah "\$route['404\_override'] = 'errors/page\_missing'". Baris ketiga mempunyai fungsi mengganti seluruh nama *file* yang mengandung '-' menjadi '\_', contoh penggunaanya adalah: "my-controller/index" menjadi "my\_controller/index".

#### 4.3 Controllers

*Controller* terdiri dari sebuah kelas yang dinamakan "C\_Skripsi". Keseluruhan aktivitas dari sistem informasi penilaian skripsi diatur oleh kelas ini. Berikut adalah gambar kelas diagram dari *controllers*:



Gambar 4.1: Gambar diagram kelas *file controllers*

- public function `index()` Berfungsi untuk mengarahkan pengguna ke *file views default* dari aplikasi.
- public function `tambahDataMahasiswa()` Berfungsi untuk mengambil data dari *view* yang tersedia, untuk kemudian diolah menjadi bahasa sql oleh *models*.

## 4.4 Models

*Models* mempunyai fungsi menghubungkan *views* dan *controllers* pada basis data. Pada penggunaan *codeigniter*, *model* dibuat dengan sangat sederhana. Berikut adalah isi dari kelas model:

```

1  <?php
2      defined('BASEPATH') OR exit('No direct script access allowed');
3
4      class Skripsi_model extends CI_Model {
5
6          public function insertDataMahasiswa($tableName, $data){
7              $res = $this->db->insert($tableName, $data);
8          }
9      }
  
```

- public function `insertDataMahasiswa($tablename, $data)` Berfungsi untuk mengolah data yang sudah diolah oleh *controllers* menjadi kueri sql *insert data*.

## 4.5 Perancangan Basis Data

Berdasarkan analisis basis data pada bab 3.2.3, maka dibuat tabel basis data berisi:

No	Nama Tabel	Jenis Data
1	<u>id</u>	int(11)
2	tahun	year(4)
3	semester	int(1)
4	npm	varchar(10)
5	nama	varchar(256)
6	judul	varchar(256)
7	namaPembimbing	varchar(256)
8	namaPembimbingPendamping	varchar(256)
9	namaKetuaTimPenguji	varchar(256)
10	namaAnggotaTimPenguji	varchar(256)
11	bobotKetuaTimPenguji	int(2)
12	bobotAnggotaTimPenguji	int(2)
13	bobotPembimbing	int(2)
14	nilaiKoordinatorSkripsi	int(2)
15	bobotKoordinatorSkripsi	int(2)
16	bobotTataTulisLaporanAnggota	int(2)
17	bobotKelengkapanMateriAnggota	int(2)
18	bobotPenguasaanMateriAnggota	int(2)
19	bobotPresentasiAnggota	int(2)
20	bobotPencapaianTujuanAnggota	int(2)
21	bobotTataTulisLaporanKetua	int(2)
22	bobotKelengkapanMateriKetua	int(2)
23	bobotPenguasaanMateriKetua	int(2)
24	bobotPresentasiKetua	int(2)
25	bobotPencapaianTujuanKetua	int(2)
26	bobotTataTulisLaporanPembimbing	int(2)
27	bobotKelengkapanMateriPembimbing	int(2)
28	bobotPenguasaanMateriPembimbing	int(2)
29	prosesBimbinganPembimbing	int(2)
30	nilaiAkhirMahasiswa	int(2)

## 4.6 Perancangan Tampilan

Tampilan pada sistem informasi penilaian skripsi haruslah dibuat semirip mungkin dengan form penilaian skripsi yang sudah ada seperti pada lampiran gambar [A.1](#) dan gambar [A.2](#).

Perbedaan yang akan ditampilkan adalah dengan adanya otomatisasi penghitungan nilai sesuai dengan bobot yang diberikan kepada penilai. Hal ini akan memberikan kemudahan penilai untuk melakukan penilaian.

Gambar [4.2](#) adalah bayangan awal tampilan untuk sistem informasi penilaian skripsi:

**Berita Acara Sidang Skripsi**

Semester: Ganjil 2016 2017 a

Telah dilaksanakan Sidang Skripsi untuk mata kuliah AIF402-E Skripsi 2 bagi:

Nama:  NPM:

Judul:

Dengan pembimbing dan penguji:

- Pembimbing Utama/Tunggal :
- Pembimbing Pendamping :
- Ketua Tim Penguji :
- Anggota Tim Penguji :

Reskapitulasi nilai Sidang Skripsi 2 yang diberikan oleh pembimbing, penguji, & koordinator skripsi :

No	Pembimbing/Penguji	Nilai	Bobot(%)	Nilai Akhir
1	Ketua Tim Penguji		35	
2	Anggota Tim Penguji		35	
3	Pembimbing		20	
4	Koordinator Skripsi		10	
	Total		100	

#### **Rekapitulasi Penilaian Skripsi 2 (PEMBIMBING)**

NPM Mahasiswa:

Komponen Penilaian	Nilai	Bobot(%)	Nilai Akhir
Tata Tulis Laporan		20	
Kelengkapan Materi		20	
Penggunaan Materi		30	
Proses Bimbingan		30	
Total		100	

#### **Rekapitulasi Penilaian Skripsi 2 (Ketua Tim Penguji)**

NPM Mahasiswa:

Komponen Penilaian	Nilai	Bobot(%)	Nilai Akhir
Tata Tulis Laporan		15	
Kelengkapan Materi		10	
Penggunaan Materi		30	
Persiapan		15	
Penjelasan Tujuan		30	
Total		100	

#### **Rekapitulasi Penilaian Skripsi 2 (Anggota Tim Penguji)**

NPM Mahasiswa:

Komponen Penilaian	Nilai	Bobot(%)	Nilai Akhir
Tata Tulis Laporan		15	
Kelengkapan Materi		10	
Penggunaan Materi		30	
Persiapan		15	
Penjelasan Tujuan		30	
Total		100	

Gambar 4.2: Perkiraan Tampilan



## BAB 5

### IMPLEMENTASI DAN PENGUJIAN

Pada bagian ini merupakan rincian atau penjelasan lanjut mengenai lingkungan implementasi perangkat keras maupun perangkat lunak sistem informasi penilaian sidang skripsi 2. Bagian terakhir akan membahas tentang pengujian yang telah dilakukan pada sistem informasi.

#### 5.1 Implementasi

Pada bagian ini akan dijabarkan lingkungan pengembangan sistem informasi dan pengujian.

##### 5.1.1 Lingkungan Implementasi dan Pengujian

Implementasi dilakukan dengan menggunakan sebuah laptop. Berikut adalah spesifikasi laptop yang digunakan:

1. Processor: Intel(R) Core(TM) i5-3230M CPU @ 2.60GHz (4CPUs), 2.6GHz
2. RAM : 4096 MB
3. Sistem operasi : Windows 7 Ultimate 64-bit (6.1, Build 7601)
4. Versi AngularJS : Version 1.5.2
5. Versi Codeigniter : Version 3.1.3
6. Versi TwitterBootstrap : Version 2.3.2
7. Versi Google Chrome : Version 55.0.2883.87 m(64-bit)

##### 5.1.2 Hasil Implementasi

Hasil implementasi dari penelitian ini adalah sebuah sistem informasi berbasis web yang menggunakan *codeigniter*, *AngularJS*, dan *Twitter Bootstrap* sebagai dasar pembuatan. Aplikasi dapat diakses melalui jaringan *global* dengan URL " <http://sipskripsi.com> ". Sistem informasi terdiri dari bagian-bagian sebagai berikut:

1. Bagian formulir berita acara sidang skripsi  
Bagian ini adalah halaman yang bersangkutan dalam pengisian data diri mahasiswa yang bersangkutan, sekaligus sebagai halaman akhir yang menyimpulkan perhitungan nilai akhir

mahasiswa. Kolom penilaian pada halaman ini tidak dapat diisi secara manual kecuali kolom penilaian milik koordinator skripsi. Kolom penilaian yang lain didapatkan berdasarkan perhitungan nilai akhir masing-masing penguji.

Berita Acara Sidang Skripsi
Lembar Rekapitulasi Ketua Tim Penguji
Lembar Rekapitulasi Anggota Tim Penguji
Lembar Rekapitulasi Pembimbing
Selesai

### Berita Acara Sidang Skripsi

Semester: Genap 2017 / 2018

---

Telah diselenggarakan Sidang Skripsi untuk mata kuliah AIF402-6 Skripsi 2 bagi:

**NPM:** 2012730017 **Nama:** Billy Yanuar

**Judul:** Sistem Informasi Penilaian Sidang Skripsi 2 dengan AngularJS

dengan pembimbing dan penguji:

**Pembimbing:** Pascal Alfadian

**Pembimbing Pendamping:** -

**Ketua Tim Penguji:** Ketua tim penguji

**Anggota Tim Penguji:** Anggota tim penguji

Rekapitulasi nilai Sidang Skripsi 2 yang diberikan oleh pembimbing, penguji & koordinator skripsi:

No	Pembimbing/Penguji	Nilai	Bobot(%)	Nilai Akhir
1	Ketua Tim Penguji	76.7	35	26.8
2	Anggota Tim Penguji	77.4	35	27.1
3	Pembimbing	77.8	20	15.5
4	Koordinator Skripsi	100	10	10
<b>Total</b>			100	79.5

Gambar 5.1: Formulir berita acara sidang skripsi 2 terisi

## 2. Bagian formulir rekapitulasi penilaian sidang skripsi 2.

Bagian ini adalah halaman yang bersangkutan dalam menampung nilai-nilai yang diberikan oleh ketua tim penguji, anggota tim penguji, dan pembimbing pada mahasiswa.

Berita Acara Sidang Skripsi
Lembar Rekapitulasi Ketua Tim Penguji
Lembar Rekapitulasi Anggota Tim Penguji
Lembar Rekapitulasi Pembimbing
Selesai

### Rekapitulasi Penilaian Skripsi 2 (Ketua Tim Penguji)

NPM: 2012730017

---

Komponen Penilaian	Nilai	Bobot(%)	Nilai Akhir
Tata Tulis Laporan	70	15	10.5
Kelengkapan Materi	80	10	8
Penguasaan Materi	75	30	22.5
Presentasi	80	15	12
Pencapaian Tujuan	79	30	23.7
<b>Total</b>		100	76.7

Gambar 5.2: Formulir rekapitulasi ketua tim penguji terisi

Berita Acara Sidang Skripsi
Lembar Rekapitulasi Ketua Tim Penguji
Lembar Rekapitulasi Anggota Tim Penguji
Lembar Rekapitulasi Pembimbing
Selesai

### Rekapitulasi Penilaian Skripsi 2 (Anggota Tim Penguji)

NPM:2012730017

Komponen Penilaian	Nilai	Bobot(%)	Nilai Akhir
Tata Tulis Laporan	78	15	11.7
Kelengkapan Materi	80	10	8
Penguasaan Materi	79	30	23.7
Presentasi	77	15	11.5
Pencapaian Tujuan	75	30	22.5
Total		100	77.4

Gambar 5.3: Formulir rekapitulasi anggota tim penguji terisi

Berita Acara Sidang Skripsi
Lembar Rekapitulasi Ketua Tim Penguji
Lembar Rekapitulasi Anggota Tim Penguji
Lembar Rekapitulasi Pembimbing
Selesai

### Rekapitulasi Penilaian Skripsi 2 (Pembimbing)

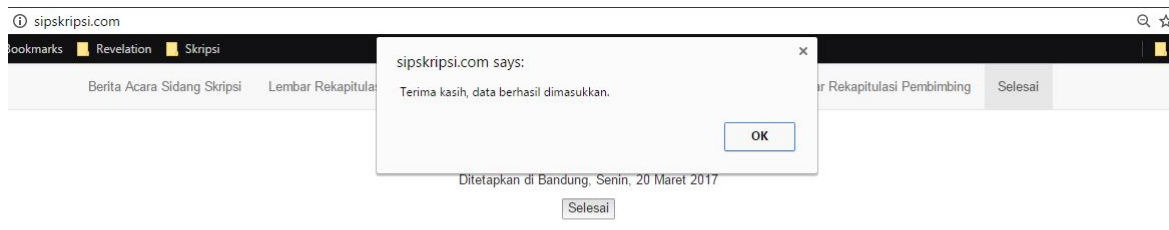
NPM:2012730017

Komponen Penilaian	Nilai	Bobot(%)	Nilai Akhir
Tata Tulis Laporan	78	20	15.6
Kelengkapan Materi	77	20	15.4
Penguasaan Materi	79	30	23.7
Proses Bimbingan	77	30	23.1
Total		100	77.8

Gambar 5.4: Formulir rekapitulasi pembimbing terisi

### 3. Bagian selesai.

Bagian ini adalah bagian terakhir dari sistem informasi. Ketika formulir sudah selesai diisi, maka dengan menekan tombol selesai pada bagian ini, *data* yang telah terisi akan dimasukkan ke dalam *database*.



Gambar 5.5: Ketika tombol selesai di klik

id	tahun	semester	npm	nama	judul	namaPembimbing	namaPembimbingPendamping	namaKetuaTimPenguji	namaAnggotaTimPenguji	bobotKetuaTimPenguji	bobotAnggotaTimPenguji
32	2017	2	2012730017	Billy Yanuar	Sistem Informasi Penilaian Penilaian Sidang Skripsi 2 dengan...	Pascal Alfadian	-	Ketua tim penguji	Anggota tim penguji	35	35

Gambar 5.6: Sebagian hasil pada database

## 5.2 Hasil Pengujian

Pengujian pada sistem informasi penilaian sidang skripsi 2 merupakan pengujian bersifat fungsional, dan pengujian eksperimental. Berikut penjelasannya:

### 5.2.1 Pengujian Eksperimental

Pengujian eksperimental dilakukan dengan cara mengikuti sidang skripsi 2 yang dilakukan pada semester ganjil 2016/2017. Pada sidang yang diujikan, penilaian dilakukan dengan dua cara, yaitu dengan sistem kini yang bekerja secara manual dan dengan sistem usulan menggunakan laptop. Sehubungan dengan sifat kerahasiaan *data* pengujian, maka dengan persetujuan pembimbing pengujian eksperimental dilakukan dengan merahasiakan identitas mahasiswa yang berhubungan.

Pada saat melakukan pengujian eksperimental, sistem kini memiliki kekurangan kecerobohan manusia yang mengakibatkan kesalahan dalam perhitungan nilai baik dari lembar rekapitulasi maupun lembar berita acara sidang skripsi. Hal tersebut diketahui pada saat membandingkan nilai perhitungan nilai akhir yang didapatkan oleh mahasiswa pada sistem kini dan sistem usulan. Pada beberapa kesalahan tersebut, penguji kembali melakukan perhitungan secara manual dengan menggunakan mesin hitung berupa kalkulator pada *gadget* penguji. Setelah perhitungan dilakukan, didapatkan bahwa sistem usulan memiliki hasil yang benar.

Percobaan eksperimental menghasilkan kesimpulan sistem usulan dapat menutupi kekurangan sistem kini yang berfokus pada perhitungan penilaian mahasiswa. Dengan menggunakan sistem usulan, perhitungan nilai dibuktikan lebih akurat dibandingkan dengan sistem kini.

### 5.2.2 Pengujian Fungsional

Pengujian fungsional dilakukan untuk mengetahui apakah sistem informasi dapat menjalankan seluruh fungsi-fungsi yang dimiliki dengan baik. Hasil pengujian fungsional sistem informasi akan dijabarkan pada tabel berikut:

No	Aksi Pengguna	Reaksi yang diharapkan	Keterangan
1	Pengguna menjalankan sistem informasi	Halaman berita acara ditampilkan	Reaksi sesuai
2	Pengguna memasukkan nilai	Menampilkan hasil dari perhitungan otomatis	Reaksi sesuai
3	Pengguna menekan tombol selesai	Menampilkan notifikasi data telah tersimpan	Reaksi sesuai
4	Pengguna menekan tombol ok pada notifikasi	Menampilkan kembali halaman lembar formulir berita acara awal sebelum terisi	Reaksi sesuai



## **BAB 6**

### **KESIMPULAN DAN SARAN**

#### **6.1 Kesimpulan**

Berdasarkan hasil penelitian yang dilakukan, didapatkan kesimpulan-kesimpulan sebagai berikut:

1. Tampilan Sistem Penilaian Sidang Skripsi 2 dibuat seperti sistem kini yang dipakai program studi teknik informatika universitas katolik parahyangan agar memudahkan penyesuaian.
2. Berdasarkan hasil pengujian eksperimental, pengisian perhitungan pada sistem kini masih terdapat faktor kesalahan manusia yang mengakibatkan kesalahan pemberian nilai kepada mahasiswa bersangkutan. Pada saat terjadi kesalahan di pengujian, maka dilakukan perbandingan hasil dari sistem kini dan sistem usulan yang kemudian membuktikan bahwa fitur perhitungan otomatis dari sistem usulan dapat menanggulangi faktor kesalahan manusia tersebut.

#### **6.2 Saran**

Berdasarkan pengujian yang dilakukan, berikut adalah beberapa saran untuk pengembang:





## DAFTAR REFERENSI


- [1] "Codeigniter documentation." [https://www.codeigniter.com/user\\_guide/general/index.html](https://www.codeigniter.com/user_guide/general/index.html).
- [2] "Angularjs documentation." <https://docs.angularjs.org/guide/introduction>.
- [3] B. Green and S. Seshadri, *AngularJS*. " O'Reilly Media, Inc.", 2013.
- [4] "Twitter bootstrap documentation." <http://getbootstrap.com/css/>.



## LAMPIRAN A

### FORM PENILAIAN SKRIPSI

Berikut adalah lembaran penilaian Skripsi yang di pakai di Program Studi Teknik Informatika Universitas Katolik Parahyangan ??:



Program Studi Teknik Informatika  
 Fakultas Teknologi Informasi dan Sains  
 Universitas Katolik Parahyangan

### Berita Acara Sidang Skripsi

Semester: Ganjil/Genap\* 20..... /20.....

Telah diselenggarakan Sidang Skripsi untuk mata kuliah AIF402-6 Skripsi 2 bagi :

NPM :  Nama:

Judul :

dengan pembimbing dan penguji :

- Pembimbing Utama/Tunggal\* :
- Pembimbing Pendamping :
- Ketua Tim Penguji :
- Anggota Tim Penguji :

Rekapitulasi nilai Sidang Skripsi 2 yang diberikan oleh pembimbing, penguji & koordinator skripsi:

No	Pembimbing/Penguji	Nilai	Bobot	Nilai Akhir
1	Ketua Tim Penguji		35%	
2	Anggota Tim Penguji		35%	
3	Pembimbing		20%	
4	Koordinator Skripsi		10%	
	<b>Total</b>		<b>100%</b>	

Ditetapkan di Bandung,   20

Ketua Tim Penguji	Anggota Tim Penguji	Pembimbing**	Koordinator Skripsi

Petunjuk pengisian :

1. \* = coret yang tidak perlu, \*\* = salah satu pembimbing saja, jika pembimbing utama hadir maka harus pembimbing utama
2. Jika minimal satu penguji dan/atau seluruh pembimbing tidak hadir, maka sidang harus dibatalkan dan dijadwalkan ulang, tetapi berita acara harus tetap dilaporkan kepada koordinator skripsi atau kepada pimpinan jurusan.

Gambar A.1: Form Penilaian Skripsi saat sidang

Berikut adalah lembaran rekapitulasi penilaian Skripsi yang di pakai di Program Studi Teknik Informatika Universitas Katolik Parahyangan ??:

**Rekapitulasi Penilaian SKRIPSI 2 (PEMBIMBING)**

NPM Mahasiswa:

Komponen Penilaian	Nilai	Bobot	Nilai Akhir
Tata Tulis Laporan		20%	
Kelengkapan Materi		20%	
Penguasaan Materi		30%	
Proses Bimbingan		30%	
		<b>Total</b>	

Tgl:    /    / 20  
 Ttd:  
  
 Nama:

---

**Rekapitulasi Penilaian SKRIPSI 2 (KETUA TIM PENGUJI)**

NPM Mahasiswa:

Komponen Penilaian	Nilai	Bobot	Nilai Akhir
Tata Tulis Laporan		15%	
Kelengkapan Materi		10%	
Penguasaan Materi		30%	
Presentasi		15%	
Pencapaian Tujuan		30%	
		<b>Total</b>	

Tgl:    /    / 20  
 Ttd:  
  
 Nama:

---

**Rekapitulasi Penilaian SKRIPSI 2 (ANGGOTA TIM PENGUJI)**

NPM Mahasiswa:

Komponen Penilaian	Nilai	Bobot	Nilai Akhir
Tata Tulis Laporan		15%	
Kelengkapan Materi		10%	
Penguasaan Materi		30%	
Presentasi		15%	
Pencapaian Tujuan		30%	
		<b>Total</b>	

Tgl:    /    / 20  
 Ttd:  
  
 Nama:

Gambar A.2: Form Rekapitulasi Penilaian Skripsi saat sidang

# LAMPIRAN B

## THE SOURCE CODE

Listing B.1: MyFurSet.java

```

1  |
2  | import java.util.ArrayList;
3  | import java.util.Collections;
4  | import java.util.HashSet;
5  |
6  | /**
7  |  *
8  |  * @author Lionov
9  |  */
10 |
11 | //class for set of vertices close to furthest edge
12 | public class MyFurSet {
13 |     protected int id; //id of the set
14 |     protected MyEdge FurthestEdge; //the furthest edge
15 |     protected HashSet<MyVertex> set; //set of vertices close to furthest edge
16 |     protected ArrayList<ArrayList<Integer>> ordered; //list of all vertices in the set for each
17 |         trajectory
18 |     protected ArrayList<Integer> closeID; //store the ID of all vertices
19 |     protected ArrayList<Double> closeDist; //store the distance of all vertices
20 |     protected int totaltrj; //total trajectories in the set
21 |
22 |     /**
23 |      * Constructor
24 |      * @param id : id of the set
25 |      * @param totaltrj : total number of trajectories in the set
26 |      * @param FurthestEdge : the furthest edge
27 |      */
28 |     public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
29 |         this.id = id;
30 |         this.totaltrj = totaltrj;
31 |         this.FurthestEdge = FurthestEdge;
32 |         set = new HashSet<MyVertex>();
33 |         ordered = new ArrayList<ArrayList<Integer>>();
34 |         for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
35 |         closeID = new ArrayList<Integer>(totaltrj);
36 |         closeDist = new ArrayList<Double>(totaltrj);
37 |         for (int i = 0;i < totaltrj;i++) {
38 |             closeID.add(-1);
39 |             closeDist.add(Double.MAX_VALUE);
40 |         }
41 |     }
42 |
43 |     /**
44 |      * set a vertex into the set
45 |      * @param v : vertex to be added to the set
46 |      */
47 |     public void add(MyVertex v) {
48 |         set.add(v);
49 |     }
50 |
51 |     /**
52 |      * check whether vertex v is a member of the set
53 |      * @param v : vertex to be checked
54 |      * @return true if v is a member of the set , false otherwise
55 |      */
56 |     public boolean contains(MyVertex v) {
57 |         return this.set.contains(v);
58 |     }

```