

**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 consectetuer.

**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
2.3 Twitter Bootstrap . . . . .	15
2.3.1 Grid System . . . . .	16
2.3.2 Form Class . . . . .	17
<b>3 ANALISIS</b>	<b>19</b>
3.1 Analisis Data Penilaian Skripsi . . . . .	19
3.1.1 Analisis Kode Sistem Kini . . . . .	19
3.1.2 Analisis Database Sistem Kini . . . . .	22
3.2 Analisis Tampilan Sistem Informasi Penilaian Skripsi . . . . .	24
<b>4 PERANCANGAN</b>	<b>27</b>
4.1 Perancangan Kelas . . . . .	27
4.2 <i>Routes</i> . . . . .	27

4.3	<i>Controllers</i>	27
4.4	<i>Models</i>	28
4.5	Perancangan Antarmuka	28
<b>DAFTAR REFERENSI</b>		<b>29</b>
<b>A FORM PENILAIAN SKRIPSI</b>		<b>31</b>
<b>B THE SOURCE CODE</b>		<b>33</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 . . . . .	16
2.16	Contoh Pembagian Grid Columns . . . . .	17
2.17	Contoh Penggunaan Kelas Form . . . . .	17
2.18	Contoh Hasil Penggunaan Kelas Form . . . . .	18
3.1	Struktur penyimpanan data program . . . . .	20
3.2	Struktur file application . . . . .	20
3.3	Diagram kelas . . . . .	21
3.4	<i>Use case</i> diagram . . . . .	22
3.5	Data yang disimpan di database . . . . .	24
3.6	Perkiraan Tampilan . . . . .	25
3.7	Perkiraan Tampilan . . . . .	26
4.1	Isi <i>file routes</i> . . . . .	27
4.2	Gambar diagram kelas <i>file controllers</i> . . . . .	28
4.3	Gambar isi dari <i>file models</i> . . . . .	28
A.1	Form Penilaian Skripsi saat sidang . . . . .	31
A.2	Form Rekapitulasi Penilaian Skripsi saat sidang . . . . .	32

## 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

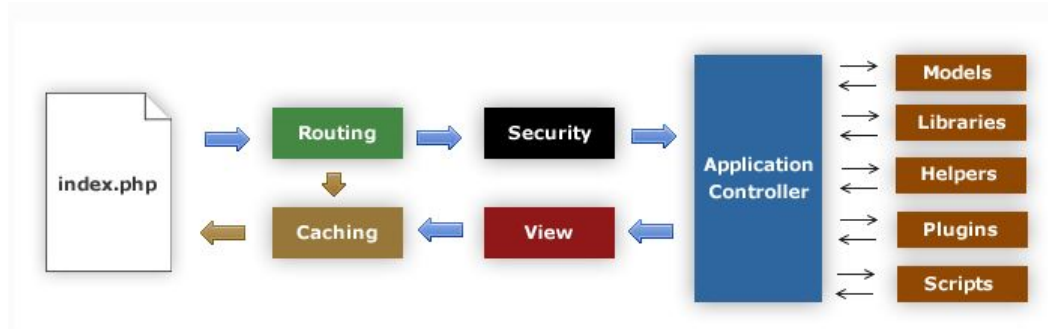
### DASAR TEORI

#### 2.1 CodeIgniter

CodeIgniter 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 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

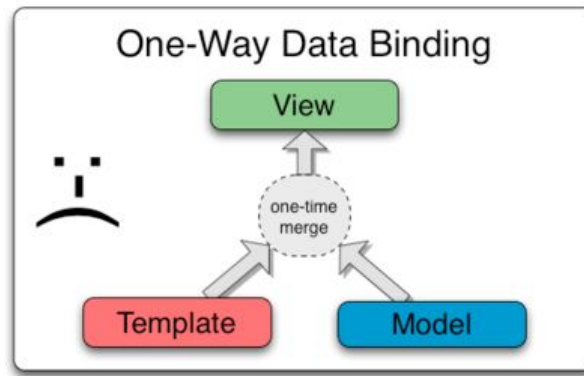
### 2.2.2 Directives

*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.

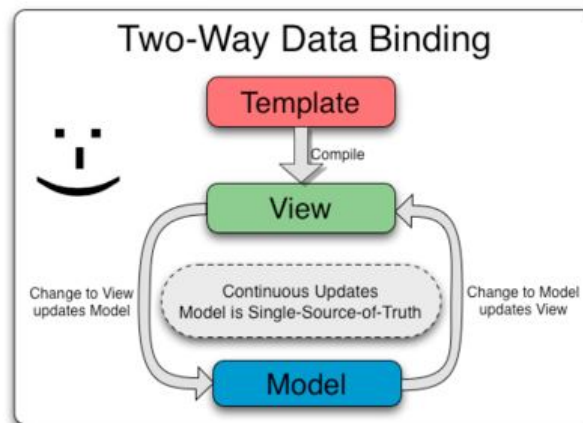
### 2.2.3 Data Binding

*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*).

## 2.3 Twitter Bootstrap

*Twitter Bootstrap* 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.

```

<form>
  <div class="form-group">
    <label for="exampleInputEmail">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail" 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 Data Penilaian Skripsi

Pada bab ini akan dijelaskan mengenai analisis kode dari sistem ini, analisis tampilan sistem ini, analisis *database* sistem ini, dan analisis sistem usulan.

##### 3.1.1 Analisis Kode Sistem Kini

Berdasarkan hasil analisis kode yang telah dibuat, berikut ini adalah penjelasan secara umum mengenai struktur kode Sistem Informasi Penilaian Skripsi 2 Gambar 3.1:

1. *Folder* "application" merupakan tempat penyimpanan *Model*, *View*, *Controller* dan fungsi-fungsi penting pada *codeigniter*
2. *Folder* "nbproject" merupakan tempat yang dibutuhkan agar sistem yang tersedia dapat dipakai pada *platform netbeans*.
3. *Folder* "public" merupakan tempat penyimpanan *file-file* Javascript dan css yang diperlukan oleh program.
4. *Folder* "system" merupakan inti dari semua sistem *codeigniter*.
5. *Folder* "user\_guide" merupakan tempat penyimpanan instruksi-instruksi yang menjadi bantuan bagi pengguna *codeigniter*.

Name	Date modified	Type	Size
application	3/17/2016 1:25 PM	File folder	
nbproject	3/10/2016 10:06 AM	File folder	
public	4/27/2016 5:48 PM	File folder	
system	3/10/2016 10:06 AM	File folder	
user_guide	3/10/2016 10:06 AM	File folder	
.gitignore	1/12/2016 4:09 PM	Text Document	1 KB
beritaacarasidangskripsi.sql	9/28/2016 10:58 AM	SQL File	6 KB
composer.json	1/12/2016 4:09 PM	JSON File	1 KB
contributing.md	1/12/2016 4:09 PM	MD File	7 KB
index.php	1/12/2016 4:09 PM	PHP File	10 KB
license.txt	1/12/2016 4:09 PM	Text Document	2 KB
readme.rst	1/12/2016 4:09 PM	RST File	3 KB

Gambar 3.1: Struktur penyimpanan data program

Name	Date modified	Type	Size
cache	3/10/2016 10:06 AM	File folder	
config	3/10/2016 10:06 AM	File folder	
controllers	11/19/2016 3:18 AM	File folder	
core	2/17/2017 2:25 AM	File folder	
helpers	3/10/2016 10:06 AM	File folder	
hooks	3/10/2016 10:06 AM	File folder	
language	3/10/2016 10:06 AM	File folder	
libraries	3/10/2016 10:06 AM	File folder	
logs	11/21/2016 11:54 ...	File folder	
models	9/20/2016 3:58 AM	File folder	
third_party	3/10/2016 10:06 AM	File folder	
views	11/19/2016 3:18 AM	File folder	
.htaccess	1/12/2016 4:09 PM	HTACCESS File	1 KB
index.html	1/12/2016 4:09 PM	Chrome HTML Do...	1 KB

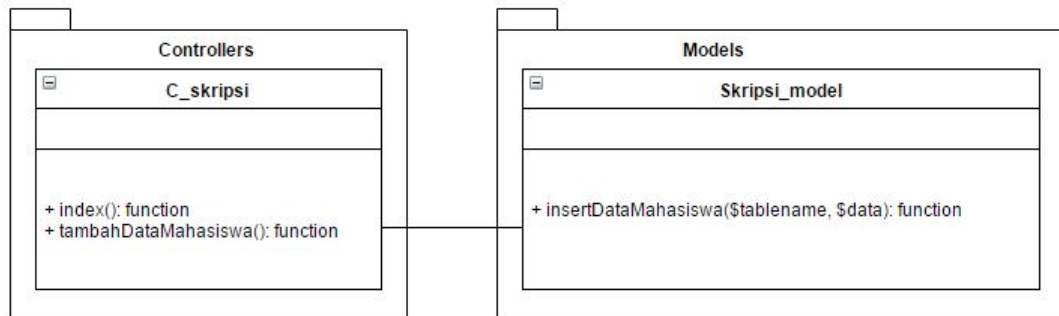
Gambar 3.2: Struktur file application

Pembuatan sistem informasi penilaian skripsi 2 berfokus pada *folder "application"* (Gambar 3.2) yang merupakan tempat penyimpanan *file-file* MVC yang menjadi dasar dari sistem ini.

1. *Folder "config"* merupakan tempat penyimpanan *file* yang berguna untuk mengatur berbagai fungsi yang terdapat pada codeigniter.
2. *Folder "controller"* merupakan tempat penyimpanan *file* yang berfungsi untuk menyambungkan *view* dengan *model* dan mengatur *routing*.



3. *Folder "models"* merupakan tempat penyimpanan *file model* yang menyambungkan program dengan perintah-perintah sql untuk *database*.
4. *Folder "views"* merupakan tempat penyimpanan *file-file* yang mengatur tampilan program.



Gambar 3.3: Diagram kelas

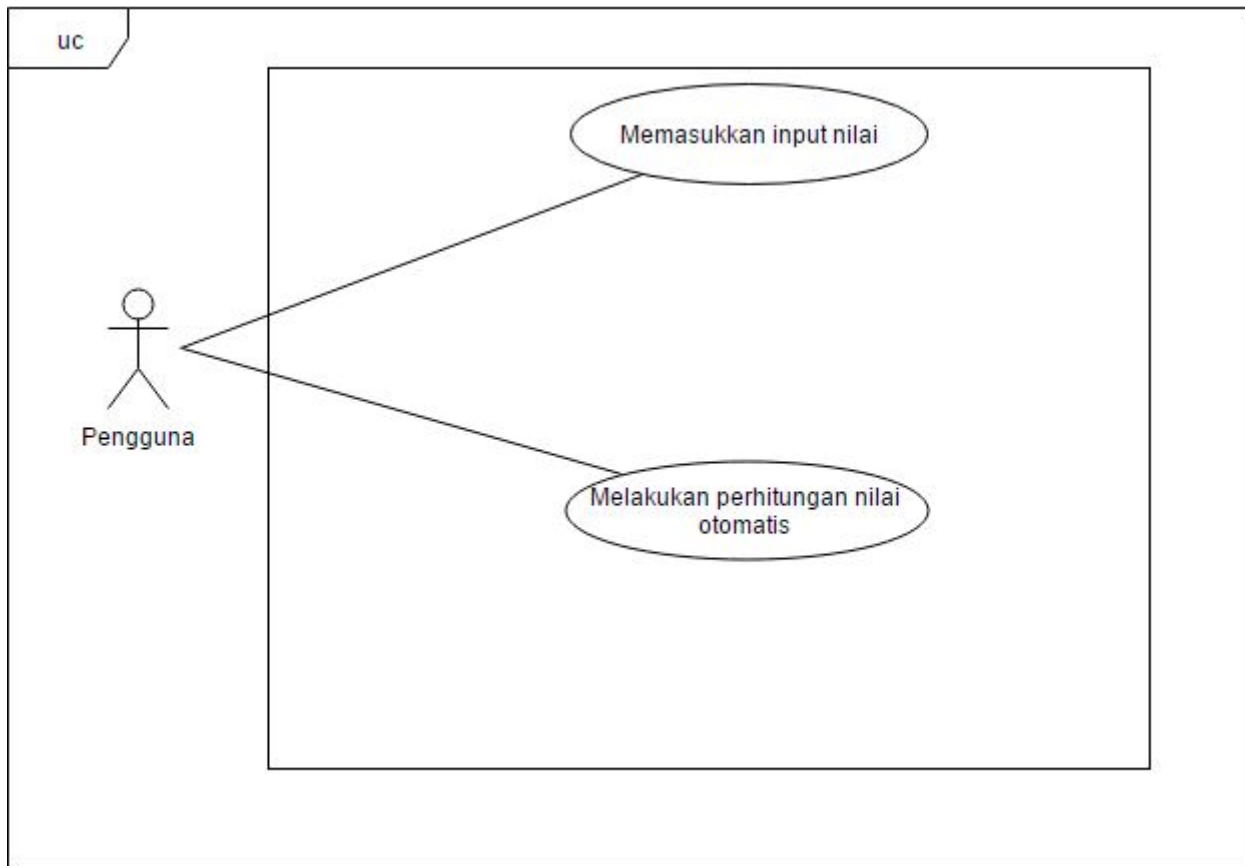
Berdasarkan diagram kelas dari sistem ini (Gambar 3.3), maka selanjutnya akan dijelaskan fungsi dari masing-masing metode yang ada:

Kelas C\_Skripsi:

- public function `index()`  
 Berfungsi untuk mengarahkan pengguna ke halaman "application/views/"  
 Nilai kembalian: halaman sistem kini
- public function `tambahDataMahasiswa()`  
 Berfungsi untuk menerjemahkan *input data* yang telah dimasukkan pengguna pada *view* sehingga dapat diterima oleh *model* sistem ini, yang pada akhirnya akan diproses untuk dimasukkan ke dalam *database* sistem ini.  
 Nilai kembalian: data yang diperlukan oleh model dalam fungsi `insertData`.

Kelas Skripsi\_model:

- public function `insertDataMahasiswa($tablename, $data)`  
 Berfungsi untuk memproses *input data* yang sudah diproses oleh *controller* untuk dimasukkan ke dalam *database*  
 Parameter: . . .
  - `tablename` adalah nama tabel.
  - `data` adalah kumpulan *input* yang sudah diolah sehingga dapat dimasukkan ke dalam fungsi sql.

Gambar 3.4: *Use case diagram*

### 3.1.2 Analisis Database Sistem Kini

Sistem informasi penilaian sidang skripsi 2 menggunakan perangkat lunak phpmyadmin sebagai sarana penyimpanan dan pengolahan *database*. Seperti yang dijelaskan pada subbab 3.1.1, didalam *folder* "application" terdapat *folder* "models" (Gambar??) yang berfungsi menghubungkan *database* 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.

#	Nama	Jenis	Penyortiran
<input type="checkbox"/> 1	<u>id</u>	int(11)	
<input type="checkbox"/> 2	tahun	year(4)	
<input type="checkbox"/> 3	semester	int(1)	
<input type="checkbox"/> 4	npm	varchar(10)	latin1_swedish_ci
<input type="checkbox"/> 5	nama	varchar(256)	latin1_swedish_ci
<input type="checkbox"/> 6	judul	varchar(256)	latin1_swedish_ci
<input type="checkbox"/> 7	namaPembimbing	varchar(256)	latin1_swedish_ci
<input type="checkbox"/> 8	namaPembimbingPendamping	varchar(256)	latin1_swedish_ci
<input type="checkbox"/> 9	namaKetuaTimPenguji	varchar(256)	latin1_swedish_ci
<input type="checkbox"/> 10	namaAnggotaTimPenguji	varchar(256)	latin1_swedish_ci
<input type="checkbox"/> 11	bobotKetuaTimPenguji	int(2)	
<input type="checkbox"/> 12	bobotAnggotaTimPenguji	int(2)	
<input type="checkbox"/> 13	bobotPembimbing	int(2)	
<input type="checkbox"/> 14	nilaiKoordinatorSkripsi	int(2)	
<input type="checkbox"/> 15	bobotKoordinatorSkripsi	int(2)	
<input type="checkbox"/> 16	bobotTataTulisLaporanAnggota	int(2)	
<input type="checkbox"/> 17	bobotKelengkapanMateriAnggota	int(2)	
<input type="checkbox"/> 18	bobotPenguasaanMateriAnggota	int(2)	
<input type="checkbox"/> 19	bobotPresentasiAnggota	int(2)	
<input type="checkbox"/> 20	bobotPencapaianTujuanAnggota	int(2)	
<input type="checkbox"/> 21	bobotTataTulisLaporanKetua	int(2)	
<input type="checkbox"/> 22	bobotKelengkapanMateriKetua	int(2)	
<input type="checkbox"/> 23	bobotPenguasaanMateriKetua	int(2)	
<input type="checkbox"/> 24	bobotPresentasiKetua	int(2)	
<input type="checkbox"/> 25	bobotPencapaianTujuanKetua	int(2)	
<input type="checkbox"/> 26	bobotTataTulisLaporanPembimbing	int(2)	
<input type="checkbox"/> 27	bobotKelengkapanMateriPembimbing	int(2)	
<input type="checkbox"/> 28	bobotPenguasaanMateriPembimbing	int(2)	
<input type="checkbox"/> 29	prosesBimbinganPembimbing	int(2)	
<input type="checkbox"/> 30	nilaiAkhirMahasiswa	int(2)	

Gambar 3.5: Data yang disimpan di database

## 3.2 Analisis Tampilan Sistem Informasi Penilaian Skripsi

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 3.6 adalah bayangan awal tampilan untuk sistem informasi penilaian skripsi:

**Berita Acara Sidang Skripsi**

Semester: Ganjil ▼ 2016 2017 a

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

Nama:  NPM:

Judul:

Dengan pembimbing dan penguji:

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

Berikut ini 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 Penyajian		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	
Presentasi		15	
Pencapaian 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	
Presentasi		15	
Pencapaian Tujuan		30	
Total		100	

Gambar 3.6: Perkiraan Tampilan

Dari bayangan awal itulah, saya mendesain tampilan dari aplikasi sistem informasi penilaian skripsi 2 ini. Gambar 3.7 merupakan tampilan pada aplikasi sistem informasi penilaian skripsi 2.

Berita Acara Sidang Skripsi

Lembar Rekapitulasi Ketua Tim Penguji

Lembar Rekapitulasi Anggota Tim Penguji

Lembar Rekapitulasi Pembimbing

Selesai

## Berita Acara Sidang Skripsi

Semester:  / 

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

NPM:  Nama: Judul: 

dengan pembimbing dan penguji:

Pembimbing: Pembimbing Pendamping: Ketua Tim Penguji: Anggota Tim Penguji: 

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

No	Pembimbing/Penguji	Nilai	Bobot(%)	Nilai Akhir
1	Ketua Tim Penguji	<input type="text"/>	35	<input type="text"/>
2	Anggota Tim Penguji	<input type="text"/>	35	<input type="text"/>
3	Pembimbing	<input type="text"/>	20	<input type="text"/>
4	Koordinator Skripsi	<input type="text"/>	10	<input type="text"/>
	Total		100	<input type="text"/>

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: 

Komponen Penilaian	Nilai	Bobot(%)	Nilai Akhir
Tata Tulis Laporan	<input type="text"/>	15	<input type="text"/>
Kelengkapan Materi	<input type="text"/>	10	<input type="text"/>
Penguasaan Materi	<input type="text"/>	30	<input type="text"/>
Presentasi	<input type="text"/>	15	<input type="text"/>
Pencapaian Tujuan	<input type="text"/>	30	<input type="text"/>
Total		100	<input type="text"/>

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: 

Komponen Penilaian	Nilai	Bobot(%)	Nilai Akhir
Tata Tulis Laporan	<input type="text"/>	15	<input type="text"/>
Kelengkapan Materi	<input type="text"/>	10	<input type="text"/>
Penguasaan Materi	<input type="text"/>	30	<input type="text"/>
Presentasi	<input type="text"/>	15	<input type="text"/>
Pencapaian Tujuan	<input type="text"/>	30	<input type="text"/>
Total		100	<input type="text"/>

Berita Acara Sidang Skripsi

Lembar Rekapitulasi Ketua Tim Penguji

Lembar Rekapitulasi Anggota Tim Penguji

Lembar Rekapitulasi Pembimbing

Selesai

## Rekapitulasi Penilaian Skripsi 2 (Pembimbing)

NPM: 

Komponen Penilaian	Nilai	Bobot(%)	Nilai Akhir
Tata Tulis Laporan	<input type="text"/>	20	<input type="text"/>
Kelengkapan Materi	<input type="text"/>	20	<input type="text"/>
Penguasaan Materi	<input type="text"/>	30	<input type="text"/>
Proses Bimbingan	<input type="text"/>	30	<input type="text"/>
Total		100	<input type="text"/>

## 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":

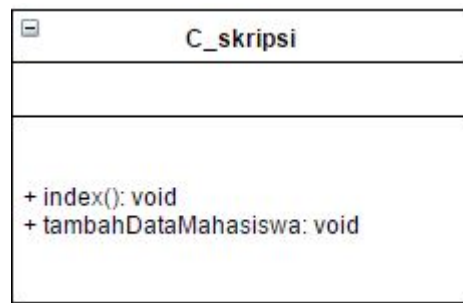
```
$route['default_controller'] = 'C_skripsi';  
$route['404_override'] = '';  
$route['translate_uri_dashes'] = FALSE;
```

Gambar 4.1: Isi *file routes*

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 penggunaannya adalah "\$route['404\_override'] = 'errors/page\_missing'". Baris ketiga mempunyai fungsi mengganti seluruh nama *file* yang mengandung '-' menjadi '\_', contoh penggunaannya 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 4.2: 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 *database*. Pada penggunaan *codeigniter*, *model* dibuat dengan sangat sederhana. Berikut adalah isi dari kelas model:

```

<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Skripsi_model extends CI_Model {

    public function insertDataMahasiswa($tableName, $data){
        $res = $this->db->insert($tableName, $data);
    }

}
  
```

Gambar 4.3: Gambar isi dari *file models*

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

#### 4.5 Perancangan Antarmuka

Perancangan antarmuka dibangun berdasarkan lembar surat penilaian yang dipakai pada sidang skripsi 2 fakultas teknologi informasi yang terlampir pada Gambar [A.1](#) dan Gambar [A.2](#).




## DAFTAR REFERENSI



## 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     }
59 }

```