

SKRIPSI

KUSTOMISASI SHARIF JUDGE UNTUK KEBUTUHAN PROGRAM STUDI TEKNIK INFORMATIKA



Stillmen Vallian

NPM: 2014730083

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

«tahun»

UNDERGRADUATE THESIS

«JUDUL BAHASA INGGRIS»



Stillmen Vallian

NPM: 2014730083

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY

«tahun»

LEMBAR PENGESAHAN

KUSTOMISASI SHARIF JUDGE UNTUK KEBUTUHAN PROGRAM STUDI TEKNIK INFORMATIKA

Stillmen Vallian

NPM: 2014730083

Bandung, «tanggal» «bulan» «tahun»

Menyetujui,

Pembimbing Utama

Pembimbing Pendamping

Pascal Alfadian, M.Comp.

«pembimbing pendamping/2»

Ketua Tim Penguji

Anggota Tim Penguji

«penguji 1»

«penguji 2»

Mengetahui,

Ketua Program Studi

Mariskha Tri Adithia, P.D.Eng

PERNYATAAN

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

KUSTOMISASI SHARIF JUDGE UNTUK KEBUTUHAN PROGRAM STUDI TEKNIK INFORMATIKA

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 Rp. 6000

Stillmen Vallian
NPM: 2014730083

ABSTRAK

«Tuliskan abstrak anda di sini, dalam bahasa Indonesia»

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

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

ABSTRACT

«Tuliskan abstrak anda di sini, dalam bahasa Inggris»

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

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

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

KATA PENGANTAR

«Tuliskan kata pengantar dari anda di sini ...»

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.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Bandung, «bulan» «tahun»

Penulis

DAFTAR ISI

KATA PENGANTAR	xv
DAFTAR ISI	xvii
DAFTAR GAMBAR	xix
DAFTAR TABEL	xxi
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	1
1.3 Tujuan	1
1.4 Batasan Masalah	2
1.5 Metodologi	2
1.6 Sistematika Pembahasan	2
2 LANDASAN TEORI	3
2.1 <i>CodeIgniter</i>	3
2.1.1 Fitur-fitur CodeIgniter	4
2.1.2 <i>Flow Chart</i> Aplikasi	5
2.1.3 <i>Model-View-Controller</i>	5
2.1.4 Desain dan Tujuan Arsitektur	9
2.2 <i>Sharif Judge</i>	10
2.2.1 Instalasi	10
2.2.2 <i>Clean URLs</i>	11
2.2.3 <i>Users</i>	12
2.2.4 Menambah Tugas	13
2.2.5 Struktur Pengujian	20
2.2.6 Deteksi Kecurangan	23
3 ANALISIS	25
3.1 Analisis Kebutuhan Perangkat Lunak Sharif Judge	25
3.1.1 <i>Security with PHP</i>	26
3.1.2 <i>Queue failed to process if submission take too long to complete?</i>	26
3.1.3 Menguji kemiripan kode antar mahasiswa	27
3.1.4 Satu Akun hanya dapat login satu waktu	27
3.1.5 Perlu ditambah petunjuk penamaan file input dan output	27
3.1.6 Sharif Judge diharapkan memiliki fungsi reminder	27
3.1.7 Membatasi soal (deskripsi & PDF) hanya bisa diakses saat assignment "open" dan setelah waktu mulai	27
3.1.8 Mengumpulkan file dengan format TXT	28
3.1.9 Mengumpulkan file JAR (java multi kelas)	28
3.1.10 Pendaftaran peserta disertai dengan <i>Display Name</i>	28

3.1.11 Nama pengguna Sharif Judge seharusnya tidak bisa diubah	28
3.1.12 Sharif Judge diharapkan memiliki fungsi dimana assignment dapat dikumpulkan tanpa adanya batasan waktu	29
3.1.13 Integrasi login ke RADIUS	29
3.1.14 Branding Teknik Informatika	29
3.1.15 Sharif Judge diharapkan memiliki Scoreboard global untuk semua assignment	30
DAFTAR REFERENSI	31
A KODE PROGRAM	33
B HASIL EKSPERIMEN	35

DAFTAR GAMBAR

2.1	<i>Flow Chart</i> Aplikasi	5
2.2	Tampilan Halaman <i>Assignments</i>	13
B.1	Hasil 1	35
B.2	Hasil 2	35
B.3	Hasil 3	35
B.4	Hasil 4	35

DAFTAR TABEL

2.1	<i>User Roles Table</i>	12
2.2	<i>Permission Table</i>	12
2.3	Masalah 1 (Penjumlahan)	16
2.4	Masalah 2 (<i>Max</i>)	17
3.1	<i>Kebutuhan Sharif Judge</i>	26

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Sharif Judge adalah *grader* otomatis yang mampu menilai ketepatan serta performansi program yang dikumpulkan mahasiswa. Perangkat lunak ini diciptakan oleh Mohammad Javad Naderi dan bersifat *open source*. *Web Interface* perangkat lunak ini dibuat menggunakan *framework CodeIgniter* dan *backend* menggunakan *BASH* [1]. Selain sebagai *grader*, perangkat lunak ini memiliki beberapa fungsi seperti deteksi plagiarisme jawaban para peserta. Cara kerja perangkat lunak ini dimulai dari dosen memasukkan data yang dibutuhkan berupa soal, peserta, dan kunci jawaban. Data yang dimasukan tersebut dapat diakses oleh para peserta. Peserta dapat mengumpulkan jawaban dalam bentuk kode program ke dalam *Sharif Judge*. *Sharif Judge* akan menjalankan kode program dan menyesuaikan dengan kunci jawaban, lalu *grader* akan menilai jawaban para peserta.

Sharif Judge digunakan oleh Jurusan Teknik Informatika Universitas Katolik Parahyangan pada mata kuliah seperti Algoritma dan Struktur Data serta Desain Analisis dan Algoritma. Perangkat lunak ini sangat membantu dosen dan mahasiswa dalam bidang akademik. Sistem penilaian otomatis merupakan salah satu fitur yang sering digunakan oleh para dosen. Dengan memanfaatkan fitur di atas, dosen dapat dengan mudah memberikan nilai tugas, kuis, atau ujian ke mahasiswa. Mahasiswa juga dapat melihat nilai secara langsung setelah jawaban dikumpulkan. Jika masih ada waktu, mahasiswa masih dapat memperbaiki jawaban yang salah. Ketika waktu sudah habis, jawaban terakhir yang dikumpulkan akan diambil sebagai jawaban final mahasiswa.

Pada prakteknya *Sharif Judge* masih butuh pengembangan, karena Jurusan Teknik Informatika memiliki kebutuhan yang lebih spesifik seperti *login* yang terintegrasi dengan *password* pada Teknik Informatika. Selain itu *Sharif Judge* terakhir dicommit pada *Github* pada bulan Juli 2015, dan masih ada beberapa bug yang belum diperbaiki. Hal tersebut menyebabkan *Sharif Judge* kurang memenuhi kebutuhan program studi Teknik Informatika.

Pada skripsi ini, peneliti akan mengembangkan *Sharif Judge* agar sesuai dengan kebutuhan yang disebutkan diatas. Dari kebutuhan yang disebutkan diatas, akan dirancang fitur-fitur untuk diimplementasikan pada *Sharif Judge*. Dengan pengimplementasian fitur yang baru, diharapkan kebutuhan mahasiswa dan dosen dapat terpenuhi.

1.2 Rumusan Masalah

1. Fitur-fitur apa saja yang dibutuhkan oleh Teknik Informatika?
2. Bagaimana mengembangkan *Sharif Judge* sehingga memenuhi kebutuhan Teknik Informatika?

1.3 Tujuan

1. Menganalisa dan mengetahui fitur-fitur yang dibutuhkan Teknik Informatika.
2. Mengimplementasi kebutuhan program studi Teknik Informatika pada *Sharif Judge*.

1.4 Batasan Masalah

Batasan masalah yang dibuat terkait dengan pengerjaan skripsi ini adalah sebagai berikut:

1. Perangkat lunak akan dikembangkan sesuai dengan kebutuhan yang telah dianalisis oleh para dosen pengguna dan daftar isu pada repositori *Sharif Judge*.

1.5 Metodologi

Metodologi yang dilakukan dalam pengerjaan skripsi ini adalah sebagai berikut:

1. Studi literatur mengenai :
 - *CodeIgniter* sebagai framework untuk mengembangkan perangkat lunak.
 - Dokumentasi *Sharif Judge* sebagai panduan untuk mengembangkan perangkat lunak.
2. Menganalisis kebutuhan-kebutuhan dari para dosen pengguna *Sharif Judge* dan daftar isu pada repositori *Sharif Judge* pada *Github*.
3. Merancang dan menentukan fitur yang akan diimplementasi.
4. Mengimplementasikan fitur terhadap perangkat lunak.
5. Mengujikan perangkat lunak ke mata kuliah selama satu semester.
6. Membuat dokumentasi perangkat lunak.

1.6 Sistematika Pembahasan

Setiap bab dalam skripsi ini memiliki sistematika penulisan yang dijelaskan kedalam poin-poin sebagai berikut:

1. Bab 1 : Pendahuluan
Bab 1 membahas mengenai gambaran umum penelitian ini. Berisi tentang latar belakang, rumusan masalah, tujuan, batasan masalah, metode penelitian, dan sistematika penulisan.
2. Bab 2 : Dasar Teori
Bab 2 membahas mengenai teori-teori yang mendukung berjalannya penelitian ini. Berisi tentang *CodeIgniter* dan dokumentasi *Sharif Judge*.
3. Bab 3 : Analisis
Bab 3 membahas mengenai analisa masalah.
4. Bab 4 : Perancangan
Bab 4 membahas mengenai perancangan yang dilakukan sebelum melakukan tahapan implementasi.
5. Bab 5 : Implementasi dan Pengujian
Bab 5 membahas mengenai implementasi dan pengujian yang telah dilakukan.
6. Bab 6 : Kesimpulan dan Saran
Bab 6 membahas hasil kesimpulan dari keseluruhan penelitian ini dan saran-saran yang dapat diberikan untuk penelitian berikutnya.

BAB 2

LANDASAN TEORI

Bab ini membahas tentang landasan teori yang akan digunakan dalam skripsi ini yang diambil dari dua sumber, yaitu "CodeIgniter Documentation" karya British Columbia Institute of Technology [2] dan "Sharif Judge Documentation" karya Mohammad Javad Naderi [3].

2.1 *CodeIgniter*

CodeIgniter merupakan sebuah *framework* bagi pengguna yang ingin membangun aplikasi web menggunakan PHP. Tujuan utamanya adalah memungkinkan para pengguna mengembangkan proyek-proyek menjadi lebih cepat dibandingkan menulis kode dari awal. *Framework* ini memiliki banyak library untuk tugas-tugas yang biasa diperlukan, serta antarmuka dan struktur logis yang sederhana untuk mengakses library ini. *CodeIgniter* membuat para pengguna lebih fokus pada proyek dengan meminimalkan jumlah kode yang dibutuhkan untuk tugas yang diberikan [2].

Beberapa keunggulan dari *CodeIgniter* yaitu:

- *Framework* yang Ringan
Inti dari sistem *CodeIgniter* hanya membutuhkan *library* yang kecil. Hal ini sangat berbeda dengan *framework* lain yang membutuhkan *resource* yang lebih. *Library* tambahan dimuat secara dinamis atau sesuai dengan permintaan sehingga sistem dapat berjalan cepat.
- Menggunakan Konsep M-V-C
CodeIgniter menggunakan pendekatan *Model-View-Controller* yang memungkinkan pemisahan anatara logika dan presentasi.
- Menghasilkan *Clean URLs*
URL yang dihasilkan oleh *CodeIgniter* berish dan *search-engine friendly*. *CodeIgniter* menggunakan pendekatan *segment-based* seperti:
example.com/news/article/345
- *Packs a Punch*
CodeIgniter dilengkapi dengan *library* yang umumnya diperlukan untuk tugas pengembangan web seperti mengakses database, mengirim email, memvalidasi data *form*, menjaga *session*, memanipulasi gambar, bekerja dengan *XML-RPC* data dan masih banyak lagi.
- *Extensible*
Sistem dapat dengan mudah diperluas dengan menggunakan *library* pengguna, *helper*, atau melalui *class extensions* dan *system hooks*.
- Tidak Membutuhkan *Template Engine*
CodeIgniter dilengkapi dengan *template parser* sederhana yang dapat digunakan secara opsional. *Template Engine* tidak dapat menandingi kinerja dari *native* PHP. Sintak yang harus dipelajari untuk menggunakan *Template Engine* biasanya lebih mudah dari mempelajari

dasar-dasar PHP. Perhatikan potongan kode PHP di bawah ini:

```
<ul>
<?php foreach ($addressbook as $name):?>
<li><?=$name?></li>
<?php endforeach; ?>
</ul>
```

Sangat berlawanan dengan *pseudo-code* yang digunakan oleh *Template Engine*:

```
<ul>
{foreach from=$addressbook item="name"}
<li>{$name}</li>
{/foreach}
</ul>
```

Terlihat *Template Engine* sedikit lebih bersih, namun harus ditukar dengan performa yang kurang baik karena *pseudo-code* harus dikonversi kembali menjadi PHP. Salah satu tujuan dari *CodeIgniter* adalah performa maksimal, oleh karena itu *CodeIgniter* tidak menggunakan *Template Engine*.

- Dokumentasi yang Baik
Dokumentasi merupakan salah satu bagian terpenting dari kode itu sendiri. *CodeIgniter* berkomitmen membuat kode yang sangat bersih dan terdokumentasi dengan baik.

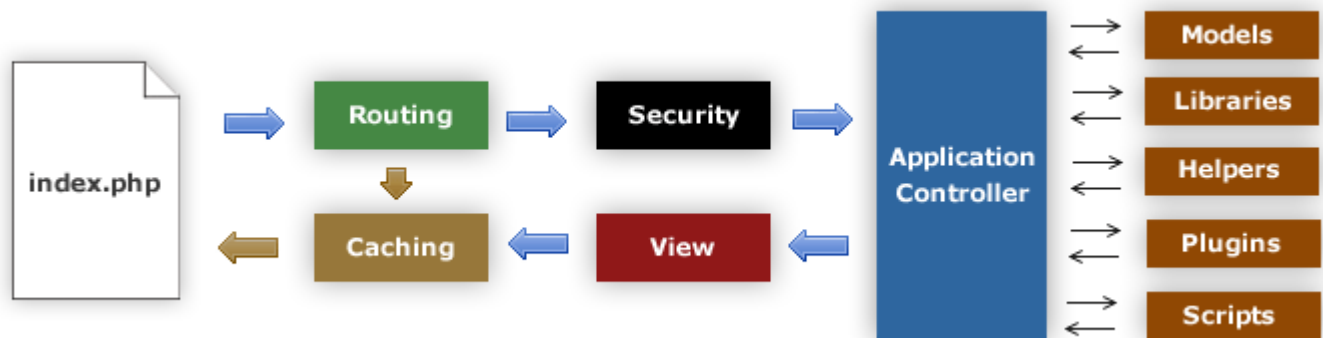
2.1.1 Fitur-fitur CodeIgniter

Berikut beberapa fitur utama yang terdapat pada *framework CodeIgniter* seperti:

- Sistem berbasis *MVC*
- *Framework* yang ringan
- *Database Class* yang lengkap dengan dukungan untuk beberapa platform
- Dukungan *query builder* untuk database
- *Form* dan validasi data
- Keamanan dan *XSS Filtering*
- *Session Management*
- *Email Sending Class*
- *Image Manipulation Library*
- *File Uploading Class*
- *Calendaring Class*
- *Unit Testing Class*

2.1.2 *Flow Chart Aplikasi*

Gambar 2.1 menunjukkan bagaimana data mengalir ke seluruh sistem [2]:



Gambar 2.1: *Flow Chart Aplikasi*

1. File *index.php* berfungsi sebagai *front controller* dan menginisialisasi *resource* utama yang dibutuhkan untuk menjalankan *CodeIgniter*.
2. *Router* memeriksa *HTTP request* untuk menentukan apa yang harus dilakukan.
3. Jika terdapat *file cache*, maka akan langsung dikirimkan ke *browser*.
4. *HTTP request* dan data pengguna yang dikirim akan terlebih dahulu disaring untuk alasan keamanan. *Application controller* akan dimuat setelah proses penyaringan selesai.
5. *Controller* akan memuat *model*, *core libraries*, *helpers* dan *resource* lain yang dibutuhkan untuk memproses permintaan khusus.
6. *View* akan di *render* kemudian dikirim ke web *browser*. Jika proses *caching* diaktifkan, maka *View* akan di *cache* terlebih dahulu sehingga permintaan berikutnya dapat dilayani.

2.1.3 *Model-View-Controller*

CodeIgniter merupakan *framework* yang menggunakan pola pengembangan *Model-View-Controller*. *MVC* adalah pendekatan perangkat lunak yang memisahkan logika aplikasi dari presentasi. Hal tersebut memungkinkan halaman web pengguna memiliki *scripting* yang minimal karena presentasi terpisah dari *scripting* PHP [2].

2.1.3.1 *Model*

Model merepresentasikan bagian struktur data pengguna. Biasanya kelas *Model* akan berisikan fungsi-fungsi yang membantu pengguna untuk mengambil, menyimpan, dan memperbarui informasi pada database. Berikut beberapa hal penting yang terdapat pada *Model*:

- Susunan dari *Model*
Kelas model berada di direktori *application/models/*. Model dapat dikelompokkan di dalam sub direktori bila model jika pengguna menginginkannya. Bentuk dasar kode pada kelas model digambarkan seperti berikut ini:

```
class Model_name extends CI_Model {
    public function _construct ()
```

```

    {
        parent::__construct();
        //constructor code
    }
}

```

Model_name adalah nama kelas dari kelas model yang pengguna buat. Penamaan kelas harus dimulai dengan huruf kapital. Pastikan kelas model merupakan turunan dari base Model (class CI_Model atau MY_Model).

- Menghubungkan Sebuah Model

Pada dasarnya model akan dimuat dan dipanggil dari method atau fungsi yang ada pada controller. Untuk menghubungkan model, pengguna harus menggunakan method berikut:

```
$this->model->model('model_name');
```

Jika model yang pengguna buat terletak di dalam sebuah sub-direktori, maka pengguna harus menyertakan alamat relatif (relative path) dari model yang dibuat. Sebagai contoh, jika model yang pengguna miliki berlokasi di application/models/blog/Queries.php pengguna dapat menghubungkannya dengan cara:

```
$this->load->model('blog/queries');
```

Pengguna dapat mengakses method yang terdapat pada model menggunakan sebuah objek dengan nama yang sama dengan nama kelas yang pengguna buat sebelumnya:

```
$this->load->model('model_name');
```

```
$this->model_name->method();
```

Jika pengguna ingin menggunakan objek yang berbeda untuk sebuah model, maka pengguna dapat menggunakan penamaan (alias) di parameter kedua:

```
$this->load->model('model_name', 'foobar');
```

```
$this->foobar->method();
```

Berikut merupakan contoh sebuah controller yang terhubung dengan sebuah model dan menampilkan data hasil olahan model ke view:

```

class Blog_controller extends CI_Controller {

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

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

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

```

- Auto-loading Model

Auto-loading (menghubungkan secara otomatis) model tertentu secara global dapat pengguna

lakukan dengan menggunakan pengaturan yang ada pada berkas `application/config/autoload.php`. Tambahkan model yang ingin pengguna hubungkan secara otomatis selama sistem berjalan: `$autoload['model'] = array('model_name');`

- Koneksi ke Database

Ketika sebuah model dipanggil, model tidak akan terhubung ke database secara otomatis. Beberapa opsi yang dapat digunakan untuk menghubungkan model ke database:

- Pengguna dapat menghubungkan dengan menggunakan standar metode database antara class Controller atau class Model.
- Pengguna dapat mengatur sebuah model melakukan *auto-connect* dengan menambahkan nilai *TRUE* (boolean) di parameter ketiga atau mengatur konektivitas sebagaimana yang telah didefinisikan di dalam berkas `application/config/database.php`:

```
\ $this->load->model('model_name', '', TRUE);
```

- pengguna dapat mengatur koneksi secara manual dengan menambahkan *item-item* berupa *array* pada *parameter* ketiga seperti contoh berikut:

```
$config['hostname'] = 'localhost';
$config['username'] = 'username';
$config['password'] = 'katasandi';
$config['database'] = 'database_name';
$config['dbdriver'] = 'mysqli';
$config['dbprefix'] = '';
$config['pconnect'] = FALSE;
$config['db_debug'] = TRUE;

$this->load->model('model_name', '', $config);
```

2.1.3.2 View

View merupakan informasi yang akan ditampilkan kepada pengguna. Umumnya *View* merupakan sebuah halaman web, namun pada *CodeIgniter*, *View* dapat berupa bagian-bagian halaman seperti *header* atau *footer*. Selain itu *View* juga dapat berupa halaman *RSS* atau jenis "halaman" lainnya. *View* tidak pernah dipanggil secara langsung, melainkan harus melalui controller karena dalam framework MVC controller berfungsi sebagai pengatur. Untuk memuat tampilan tertentu, pengguna dapat menggunakan *method* berikut:

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

CodeIgniter dapat menangani beberapa panggilan method `$this->load->view()` dari dalam controller. Jika lebih dari satu panggilan terjadi, maka panggilan tersebut ditambahkan bersama. Contohnya pengguna ingin memiliki header view, menu view, content view dan footer view. Kode program yang digunakan seperti ini:

```
class Page{
    public function INDEX(){
        $data['page_title'] = 'Your title';
        $load->view('header');
        $load->view('menu');
        $load->view('content', $data);
        $load->view('footer');
    }
}
```

2.1.3.3 Controller

Controller berfungsi sebagai perantara antara *Model*, *View*, dan *resource* lain yang dibutuhkan untuk memproses *HTTP request* dan menghasilkan halaman web. Controller merupakan sebuah kelas yang dinamakan demikian agar dapat dikaitkan dengan URI. Sebagai contoh URI "example.com/index.php/blog/", CodeIgniter akan mencari controller bernama Blog.php dan menjalankannya. Nama controller harus diawali dengan huruf kapital. Selain itu controller juga harus extend kelas "CI_Controller". Perhatikan contoh berikut: Contoh yang benar :

```
<?php
class Blog extends CI_Controller {

}
```

Contoh yang salah :

```
<?php
class blog extends CI_Controller {

}
```

Berikut beberapa hal penting yang terdapat pada Controller:

- **Method**
Untuk menjalankan suatu method, pengembang perlu menuliskannya pada segmen kedua URI. Contoh "example.com/index.php/blog/comments" maka method comments() akan dijalankan pada controller blog.php. Method index() akan dijalankan jika bagian kedua URI kosong. Jika URI mengandung lebih dari dua segment, maka segment-segment tersebut akan dimasukkan ke dalam method sebagai parameter.
- **Default Controller**
CodeIgniter dapat diperintahkan untuk menjalankan *default controller* jika tidak terdapat URI. Hal ini umumnya terjadi ketika terdapat permintaan menggunakan URL dasar *website*. Penentuan *default controller* terdapat pada file "application/config/routes.php". Perhatikan contoh berikut:

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

Blog merupakan nama kelas controller yang ingin digunakan. Jika pengguna mengakses file index.php utama tanpa menentukan segmen URI, maka akan dijalankan controller blog.

- **Processing Output**
CodeIgniter memiliki kelas *output* yang menangani pengiriman data ke *web browser* secara otomatis. Dalam beberapa kasus saat pengguna ingin mengubah cara pengiriman data tersebut, CodeIgniter akan menambahkan method bernama "__output()" ke *controller* terkait. Jika controller memiliki method bernama "__output()" maka controller tersebut akan selalu dipanggil oleh kelas "output". Contoh penggunaan method "__output()" :

```
public function __output(\ $output)
{
    echo $output;
}
```

- **Private Method** Method-method dengan tipe *private* tidak dapat diakses oleh publik. Method ini hanya dapat diakses oleh method lain dalam *controller* yang sama dan method ini juga tidak dapat diakses melalui URL. Contoh penulisan *private method*:

```
private function _utility()  
{  
    // kode program  
}
```

Method di atas tidak dapat diakses dengan cara pemanggilan method pada umum seperti berikut:

```
example.com/index.php/blog/_utility/
```

- Mengorganisir Controller ke Dalam Sub Direktori

CodeIgniter mengizinkan pengguna untuk mengorganisir *controller* ke dalam sub direktori. Pengguna dapat membuat sub direktori di dalam direktori *application/controllers/* dan simpan kelas *controller* ke dalamnya. Ketika menggunakan fitur ini, pengguna harus menspesifikasikan folder tersebut ke dalam URI. Perhatikan contoh berikut: Sebuah controller berlokasi pada direktori

```
application/controllers/products/Shoes.php
```

Untuk memanggil controller di atas, URI pengguna akan terlihat seperti ini:

```
example.com/index.php/products/shoes/show/123
```

CodeIgniter memiliki pendekatan yang cukup fleksibel terhadap *MVC* karena *Model* tidak selalu diperlukan. Para pengguna dapat membangun aplikasi minimal menggunakan *Controller* dan *View*. Hal tersebut dapat dilakukan jika pengguna tidak memerlukan adanya pemisahan tambahan atau pengguna merasa bahwa mempertahankan sebuah *Model* membutuhkan kompleksitas yang lebih tinggi [2].

2.1.4 Desain dan Tujuan Arsitektur

Dari sudut pandang teknis, dan arsitektural, CodeIgniter dibuat dengan tujuan sebagai berikut:

- *Dynamic Instation*
Dalam *CodeIgniter*, komponen dimuat dan rutinitas dieksekusi hanya jika diminta. Tidak ada asumsi yang dibuat oleh sistem tentang apa yang mungkin diperlukan di luar resource utama, sehingga sistem ini sangat ringan secara *default*. *Event*, *Controller* dan *View* yang pengguna rancang akan menentukan apa yang dipanggil.
- *Loose Coupling*
Coupling adalah sejauh mana komponen-komponen dari sistem saling mengandalkan satu sama lain. Semakin sedikit komponen yang bergantung satu sama lain, maka komponen tersebut lebih dapat digunakan kembali dan sistem menjadi fleksibel. Tujuan dari *framework* ini adalah sistem yang sangat longgar (*very loosely coupled system*).
- *Component Singularity*
Singularity adalah sejauh mana komponen memiliki tujuan yang difokuskan secara sempit. Dalam CodeIgniter, setiap kelas dan fungsinya sangat otonom. Hal tersebut memungkinkan fungsi dapat berjalan secara maksimal.

CodeIgniter merupakan sistem yang loosely coupled dengan singularitas komponen yang tinggi (*dynamically instantiated*). Codeigniter berusaha untuk sederhana, fleksible, dan kinerja tinggi dengan paket yang sekecil mungkin [2].

2.2 *Sharif Judge*

Sharif Judge adalah *grader* otomatis yang mampu menilai ketepatan serta performansi program yang dikumpulkan mahasiswa. Perangkat lunak ini diciptakan oleh Mohammad Javad Naderi dan bersifat *open source*. *Web Interface* perangkat lunak ini ditulis menggunakan PHP (*framework CodeIgniter*) dan *backend* menggunakan *BASH* [1]. Selain sebagai *grader* otomatis, *Sharif Judge* juga memiliki beberapa fitur lainnya seperti:

- Beberapa peran *users* (*admin, head instructor, instructor, student*)
- Sandboxing (belum diterapkan untuk *python*)
- Deteksi kecurangan (mendeteksi kode yang mirip) menggunakan *Moss*
- Pengaturan untuk menilai keterlambatan pengiriman
- Antrian Pengiriman
- Mengunduh hasil dalam bentuk *file excel*
- Mengunduh kode yang telah dikirim dalam bentuk *file zip*
- Metode "*Output Comparison*" dan "*Tester Code*" untuk memeriksa kebenaran dari hasil keluaran.
- Menambahkan beberapa pengguna sekaligus
- Diskripsi Masalah (*PDF/Markdown/HTML*)
- Penilaian ulang (*rejudge*)
- Papan nilai
- Notifikasi

2.2.1 Instalasi

Untuk menjalankan *Sharif Judge* membutuhkan sebuah *server Linux* dengan persyaratan berikut [3]:

- *Webserver* menjalankan PHP versi 5.3 atau yang lebih baru
- Pengguna harus dapat menjalankan php dari *command line*. Pada *Ubuntu*, pengguna perlu menginstal paket *php5-cli*
- *Mysql database* (dengan ekstensi *mysqli* untuk PHP) atau *PostgreSQL database*
- PHP harus memiliki akses untuk menjalankan *shell commands* menggunakan fungsi *shell_exec*. Contohnya seperti *command* di bawah ini:

```
echo shell_exec("php -v");
```

- Perkakas yang digunakan untuk *compiling* dan menjalankan kode yang dikumpulkan (*gcc, g++, javac, java, python2, python3 commands*)
- *Perl* lebih baik diinstal untuk ketepatan waktu, batas memori dan memaksimalkan batas ukuran pada *output* kode yang dikirimkan

Jika persyaratan di atas telah terpenuhi, maka akan masuk tahap instalasi sebagai berikut:

- Mengunduh versi terakhir dari Sharif Judge dan *unpack* hasil download di direktori *public html*
- (Pilihan) Pindahkan folder *system* dan *application* keluar dari *public directory* dan masukan *path* lengkap di file *index.php*

```
$system_path = '/home/mohammad/secret/system';
$application_folder = '/home/mohammad/secret/application';
```

- Buat sebuah *Mysql* atau *PostgreSql* database untuk Sharif Judge. Jangan menginstall paket koneksi database untuk *C/C++*, *Java*, atau *Python*
- Atur pengaturan koneksi database di file *application/config/database.php*. Pengguna dapat menggunakan awalan untuk nama tabel.

```
/* Enter database connection settings here: */
'dbdriver' => 'postgre', // database driver (mysqli, postgre)
'hostname' => 'localhost', // database host
'username' => '', // database username
'password' => '', // database password
'database' => '', // database name
'dbprefix' => 'shj_', // table prefix
/*****/
```

- Buat *application/cache/Twig* dapat ditulis oleh PHP
- Buka halaman utama *Sharif Judge* pada web *browser* dan ikuti proses instalasi berikutnya
- *Log in* menggunakan akun *admin*
- Pindahkan folder *tester* dan *assignments* di luar *public directory* lalu simpan *path* lengkap di halaman *Settings*. Dua folder tersebut harus dapat ditulis oleh PHP. File-file yang diserahkan akan disimpan di folder *assignments* sehingga tidak dapat diakses publik.

2.2.2 Clean URLs

Secara *default*, *index.php* merupakan bagian dari seluruh *urls* yang ada pada *Sharif Judge* seperti [3]:

```
http://example.mjnaderi.ir/index.php/dashboard
http://example.mjnaderi.ir/index.php/users/add
```

Pengguna dapat menghilangkan *index.php* dan memiliki *urls* yang baik jika sistem pengguna mendukung aturan *rewrite* seperti:

```
http://example.mjnaderi.ir/dashboard
http://example.mjnaderi.ir/users/add
```

Untuk memungkinkan *clean urls*, ubah isi file *.htaccess2* menjadi *.htaccess* yang berlokasi di direktori utama *Sharif Judge*. Berikut isi file *.htaccess2*:

```
# You also need to change
# $config['index_page'] = 'index.php';
# to
# $config['index_page'] = '';
# in application/config/config.php
# in order to enable clean urls.
```

```
RewriteEngine on
```

```
RewriteCond $1 !^(index\.php|assets|robots\.txt)
RewriteRule ^(.*)$ index.php?/$1 [L]
```

Lalu buka file *application/config/config.php* dan ubah

```
$config['index_page'] = 'index.php';
```

menjadi

```
$config['index_page'] = '';
```

2.2.3 Users

Pada *Sharif Judge*, *users* dikelompokkan menjadi 4 yaitu *Admins*, *Head Instructor*, *Instructor*, dan *Students* Tabel 2.1 menunjukan *level users* [3].

Tabel 2.1: User Roles Table	
User Role	User Level
<i>Admin</i>	3
<i>Head Instructor</i>	2
<i>Instructor</i>	1
<i>Student</i>	0

Setiap *users* memiliki aksi yang berbeda-beda. Aksi yang dapat dilakukan para *users* akan disesuaikan dengan *level* masing-masing. Perhatikan tabel 2.2 berikut:

Tabel 2.2: Permission Table				
Aksi	<i>Admin</i>	<i>Head Instructor</i>	<i>Instructor</i>	<i>Student</i>
Mengubah <i>Settings</i>	✓	×	×	×
Menambah/Menghapus <i>users</i>	✓	×	×	×
Mengubah Peran <i>users</i>	✓	×	×	×
Menambah/Menghapus/Mengubah Tugas	✓	✓	×	×
Mengunduh <i>Test</i>	✓	✓	×	×
Menambah/Menghapus/Mengubah Notifikasi	✓	✓	×	×
<i>Rejudge</i>	✓	✓	×	×
Melihat/ <i>Pause</i> /Melanjutkan/ <i>Submission Queue</i>	✓	✓	×	×
Mendeteksi Kode yang Mirip	✓	✓	×	×
Melihat Semua Kode	✓	✓	✓	×
Mengunduh Kode Final	✓	✓	✓	×
Memilih Tugas	✓	✓	✓	✓
<i>Submit</i>	✓	✓	✓	✓

Pengguna dapat menambahkan *users* dengan mengklik pada bagian *Add Users* di halaman *Users*. Pengguna harus mengisi semua informasi yang ada pada *textarea*. Baris dimulai dengan komentar *#*. Setiap baris lainnya mewakili pengguna dengan sintaks berikut:

```
USERNAME EMAIL PASSWORD ROLE
```

- * Usernames dapat berisikan huruf kecil atau nomor dan harus terdiri antara 3 sampai 20 karakter.
- * Passwords harus terdiri antara 6 sampai 30 karakter.
- * Pengguna dapat menggunakan `RANDOM[n]` untuk menghasilkan password

acak yang terdiri dari n -digit karakter.
 * ROLE harus terdiri dari salah satu ini: 'admin', 'head_instructor', 'instructor', 'student'

Contoh:

```
# This is a comment!
# This is another comment!
instructor instructor@sharifjudge.ir 123456 head_instructor
instructor2 instructor2@sharifjudge.ir random[7] instructor
student1 st1@sharifjudge.ir random[6] student
student2 st2@sharifjudge.ir random[6] student
student3 st3@sharifjudge.ir random[6] student
student4 st4@sharifjudge.ir random[6] student
student5 st5@sharifjudge.ir random[6] student
student6 st6@sharifjudge.ir random[6] student
student7 st7@sharifjudge.ir random[6] student
```

2.2.4 Menambah Tugas

Pengguna dapat menambahkan tugas dengan cara mengklik *Add* di halaman *Assignments* [3]. Pengguna akan melihat halaman seperti gambar 2.2.

Gambar 2.2: Tampilan Halaman *Assignments*

Berikut beberapa pengaturan yang terdapat pada halaman *Add Assignments*:

- *Assignment Name*
Tugas akan ditampilkan dengan nama ini dalam daftar tugas.
- *Start Time*
Users tidak dapat mengumpulkan tugas sebelum "*Start Time*". Gunakan format ini untuk *start time*: *MM/DD/YYYY HH:MM:SS*. Contoh: *08/31/2013 12:00:00*
- *Finish Time, Extra Time*
Users tidak dapat mengumpulkan tugas setelah *Finish Time* + *Extra Time*. Tugas yang telat akan dikalikan dengan koefisien tertentu. Pengguna harus menulis *script* PHP untuk menghitung koefisien pada bidang "*Coefficient Rule*". Gunakan format ini untuk *finish time*: *MM/DD/YYYY HH:MM:SS*. Contoh: *08/31/2013 23:59:59*. Waktu ekstra harus dalam menit. Pengguna dapat menggunakan *. Contoh *120* (2 jam) atau *48*60* (2 hari).

- *Participants*

Masukan *username* dari partisipan disini. Gunakan tanda koma untuk memisah *username* antar partisipan. Hanya *users* ini yang dapat mengumpulkan tugas. Pengguna dapat menggunakan kata kunci *ALL* untuk mengijinkan semua *users* agar dapat mengumpulkan tugas. Contoh: *admin, instructor1 , instructor2 ,student1 , student2,student3 , student4*.

- *Open*

Pengguna dapat membuka atau menutup tugas untuk *students* menggunakan pilihan ini. Jika pengguna menutup tugas, *non-student users* masih dapat mengumpulkan tugas.

- *Scoreboard*

Pengguna dapat mengaktifkan atau mematikan papan nilai dengan menggunakan pilihan ini.

- *Java Exceptions* Pengguna dapat mengaktifkan dan mematikan *java exceptions* yang ditunjukkan kepada *students*. Perubahan pada pilihan ini tidak berdampak pada kode yang sebelumnya sudah dinilai. Nama *exception* akan muncul jika *tester/java_exceptions_list* berisikan nama tersebut. Jika pengguna mengaktifkan fitur ini, kode di bawah ini akan ditampilkan kepada *students* saat *exception* dilemparkan:

```

Test 1
ACCEPT
Test 2
Runtime Error (java.lang.ArrayIndexOutOfBoundsException)
Test 3
Runtime Error (java.lang.ArrayIndexOutOfBoundsException)
Test 4
ACCEPT
Test 5
ACCEPT
Test 6
ACCEPT
Test 7
ACCEPT
Test 8
Runtime Error (java.lang.ArrayIndexOutOfBoundsException)
Test 9
Runtime Error (java.lang.StackOverflowError)
Test 10
Runtime Error (java.lang.ArrayIndexOutOfBoundsException)

```

- *Coefficient Rule*

Pengguna dapat menulis *script* PHP pada bagian ini. Pengguna harus memasukan koefisien (dari 100) pada variabel *\$coefficient*. Pengguna dapat menggunakan variabel *\$extra_time* dan *\$delay*. *\$extra_time* merupakan total waktu ekstra yang diberikan kepada *users* dalam satuan detik dan *\$delay* merupakan jumlah detik berlalu dari waktu selesai (bisa negatif). *Script* PHP pada bagian ini tidak mengandung *tags* `<?php`, `<?`, `?>`. Berikut contoh *\$extra_time* 172800 (2 hari):

```

if ($delay<=0)
// no delay
$coefficient = 100;

elseif ($delay<=3600)
// delay less than 1 hour
$coefficient = ceil(100-((30*$delay)/3600));

elseif ($delay<=86400)
// delay more than 1 hour and less than 1 day
$coefficient = 70;

elseif (($delay-86400)<=3600)
// delay less than 1 hour in second day
$coefficient = ceil(70-((20*($delay-86400))/3600));

elseif (($delay-86400)<=86400)
// delay more than 1 hour in second day
$coefficient = 50;

elseif ($delay > $extra_time)
// too late
$coefficient = 0;

```

- *Time Limit*
Pengguna dapat mengatur batas waktu untuk menjalankan kode dalam satuan milisekon. *Python* dan *Java* biasanya lebih lambat dari *C/C++*. Oleh karena itu mereka membutuhkan waktu yang lebih.
- *Memory Limit*
Pengguna dapat mengatur batas memori dalam satuan *kilobyte*, namun penggunaan *Memory Limit* tidak terlalu akurat.
- *Allowed Languages*
Pengguna dapat mengatur bahasa untuk setiap permasalahan (dipisahkan menggunakan koma). Bahasa yang tersedia seperti *C*, *C++*, *Java*, *Python 2*, *Python 3*, *zip*, *PDF*. Pengguna dapat menggunakan *zip* atau *PDF* jika mengaktifkan pilihan *Upload Only*. Contoh: *C*, *C++*, *zip* atau *Python 2*, *Python 3* atau *Java*, *C*.
- *Diff Command*
Command ini digunakan untuk membandingkan keluaran dengan keluaran yang benar. Secara default *Sharif Judge* menggunakan *diff*, namun pengguna dapat mengubah *command* pada bagian ini.
- *Diff Arguments*
Pengguna dapat mengatur argumen dari *Diff Command* disini. Untuk melihat daftar lengkap *diff* argumen, pengguna dapat melihat *man diff*. *Sharif Judge* menambahkan dua pilihan baru yaitu *ignore* dan *identical*. *Ignore* akan menghiraukan semua baris baru dan spasi. *Identical* tidak akan menghiraukan apapun namun keluaran dari file yang dikumpulkan harus identik dengan keluaran *test case* agar dapat diterima.
- *Upload Only*
Jika pengguna mengatur masalah sebagai *Upload-Only*, maka *Sharif Judge* tidak akan menilai tugas pada masalah tersebut. Pengguna dapat menggunakan *zip* dan *PDF* pada *allowed languages* jika mengaktifkan pilihan ini.

2.2.4.1 Contoh Tugas

Berikut contoh tugas untuk mencoba *Sharif Judge*. Menambah tugas ini dengan mengklik *Add* di halaman *Assignment*. Tugas dibagi menjadi 3 permasalahan:

1. Masalah 1 (Penjumlahan)

Program pengguna akan menerima masukan bilangan *integer* *n*, kemudian menerima masukan lagi sebanyak *n* buah bilangan *integer* dan menampilkan hasil penjumlahan dari *n* nomor tersebut. Untuk lebih jelas, perhatikan tabel 2.3.

Tabel 2.3: Masalah 1 (Penjumlahan)

Sample Input	Sample Output
5 54 78 0 4 9	145

2. Masalah 2 (*Max*)

Program pengguna akan menerima masukan bilangan *integer* *n*, kemudian menerima masukan lagi sebanyak *n* buah bilangan *integer* dan menampilkan hasil penjumlahan dari dua nilai tertinggi. Untuk lebih jelas, perhatikan tabel 2.4.

Tabel 2.4: Masalah 2 (*Max*)

Sample Input	Sample Output
7 162 173 159 164 181 158 175	356

3. Masalah 2 (*Upload!*)

Pengguna diharuskan mengunggah sebuah *file C* atau *zip*. Masalah ini menggunakan pilihan "*Upload Only*" sehingga tidak akan dinilai oleh Sharif Judge.

Pengguna dapat menemukan *file zip* pada *folder Assignments*. Perhatikan susunan pohon dari tugas ini:

```
.
|-- p1
|   |-- in
|       |-- input1.txt
|       |-- input2.txt
|       |-- input3.txt
|       |-- input4.txt
|       |-- input5.txt
|       |-- input6.txt
|       |-- input7.txt
|       |-- input8.txt
|       |-- input9.txt
|       |-- input10.txt
|   |-- out
|       |-- output1.txt
|   |-- tester.cpp
|   |-- desc.md
|-- p2
|   |-- in
|       |-- input1.txt
|       |-- input2.txt
|       |-- input3.txt
|       |-- input4.txt
|       |-- input5.txt
|       |-- input6.txt
|       |-- input7.txt
|       |-- input8.txt
|       |-- input9.txt
|       |-- input10.txt
|   |-- out
|       |-- output1.txt
|       |-- output2.txt
|       |-- output3.txt
|       |-- output4.txt
|       |-- output5.txt
|       |-- output6.txt
|       |-- output7.txt
|       |-- output8.txt
|       |-- output9.txt
```

```
| | ——— output10.txt
| |--- desc.md
| ——— Problem2.pdf
|--- p3
| ——— desc.md
|—— SampleAssignment.pdf
```

Masalah 1 menggunakan metode "*Tester*" untuk mengecek keluaran sehingga memiliki *file tester.cpp* (*Tester Script*). Masalah 2 menggunakan metode *Output Comparison* untuk mengecek keluaran sehingga memiliki dua *folder* (*in* dan *out*) yang berisikan *test case*. Masalah 3 merupakan masalah yang menggunakan pilihan *Upload-Only*.

2.2.4.2 Contoh Solusi

Permasalahan diatas dapat diselesaikan menggunakan contoh solusi berikut ini:

- Solusi Masalah 1

Menggunakan bahasa *C*

```
#include<stdio.h>
int main(){
    int n;
    scanf("%d",&n);
    int i;
    int sum =0 ;
    int k;
    for(i=0 ; i<n ; i++){
        scanf("%d",&k);
        sum+=k;
    }
    printf("%d\n",sum);
    return 0;
}
```

Menggunakan bahasa *C++*

```
#include <iostream>
using namespace std;
int main(){
    int n, sum=0;
    cin >> n;
    for (int i=0 ; i<n ; i++){
        int a;
        cin >> a;
        sum += a;
    }
    cout << sum << endl;
    return 0;
}
```

Menggunakan bahasa *Java*

```
import java.util.Scanner;
class sum
{
```



```

public static void main(String [] args)
{
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    int sum =0;
    for (int i=0 ; i<n ; i++)
    {
        int a = sc.nextInt();
        sum += a;
    }
    System.out.println(sum);
}

```

- Solusi Masalah 2
Menggunakan bahasa C

```

#include<stdio.h>
int main(){
    int n , m1=0, m2=0;
    scanf("%d",&n);
    for (;n--;){
        int k;
        scanf("%d",&k);
        if (k>=m1){
            m2=m1;
            m1=k;
        }
        else if (k>m2)
            m2=k;
    }
    printf("%d",m1+m2);
    return 0;
}

```

Menggunakan bahasa C++

```

#include<iostream>
using namespace std;
int main(){
    int n , m1=0, m2=0;
    cin >> n;
    for (;n--;){
        int k;
        cin >> k;
        if (k>=m1){
            m2=m1;
            m1=k;
        }
        else if (k>m2)
            m2=k;
    }
    cout << (m1+m2) << endl ;
}

```

```

        return 0;
    }

```

2.2.5 Struktur Pengujian

Pengguna harus menyediakan sebuah *file zip* yang berisikan *test cases* ketika menambahkan tugas. *File zip* ini dapat berisikan folder-folder untuk setiap masalah. Pengguna harus memberikan nama pada folder sesuai aturan seperti *p1*, *p2*, *p3*, *dst*. Tugas yang menggunakan pilihan *Upload-Only* tidak membutuhkan *folder* [3].

2.2.5.1 Metode Pengecekan

Sharif Judge memiliki dua metode pengecekan untuk setiap permasalahan yaitu metode "*Input/Output*" *Comparison* dan metode *Tester*.

- *Metode Input/Output Comparison*

Dengan metode ini, pengguna harus memasukan beberapa *file input* dan *output* pada *folder* masalah. *Sharif Judge* akan memasukan nilai dari *file input* ke kode *users* dan membandingkan hasil keluaran dari kode *users* dengan *file output*. *Input files* harus berada dalam folder "*in*" dengan nama *input1.txt*, *input2.txt*, *dst*. *Output files* harus berada dalam folder "*out*" dengan nama *output1.txt*, *output2.txt*, *dst*.

- *Metode Tester*

Dengan metode ini, pengguna harus menyediakan beberapa *file input* dan sebuah *file C++* (*tester.cpp*) dan beberapa *file output*. *Sharif Judge* akan memasukan nilai dari *file input* ke kode *users* dan mengambil keluaran dari kode *users*. *tester.cpp* akan mengambil nilai dari *file input*, *file output*, dan keluaran *users*. Jika keluaran dari kode *users* benar akan mengembalikan nilai 0, sebaliknya akan mengeluarkan nilai 1. Berikut contoh kode untuk menulis *tester.cpp*:

```

/*
* tester.cpp
*/

#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main(int argc, char const *argv[])
{

    ifstream test_in(argv[1]); /* Stream ini membaca
    isi file input */
    ifstream test_out(argv[2]); /* Stream ini membaca
    isi file output */
    ifstream user_out(argv[3]); /* Stream ini membaca
    isi keluaran users */

    /* Kode Pengguna */
    /* Jika keluaran kode user benar, mengembalikan nilai 0,
    sebaliknya mengembalikan 1 */

    ...

```

```
}

```

2.2.5.2 Contoh *File*

Pengguna dapat menemukan contoh *file* pengujian pada *folder Assignments*. Perhatikan susunan pohon dari *file* tersebut:

```
.
|-- p1
|   |-- in
|       |-- input1.txt
|       |-- input2.txt
|       |-- input3.txt
|       |-- input4.txt
|       |-- input5.txt
|       |-- input6.txt
|       |-- input7.txt
|       |-- input8.txt
|       |-- input9.txt
|       --- input10.txt
|   |-- out
|       --- output1.txt
|   --- tester.cpp
--- p2
    |-- in
    |   |-- input1.txt
    |   |-- input2.txt
    |   |-- input3.txt
    |   |-- input4.txt
    |   |-- input5.txt
    |   |-- input6.txt
    |   |-- input7.txt
    |   |-- input8.txt
    |   |-- input9.txt
    |   --- input10.txt
    --- out
        |-- output1.txt
        |-- output2.txt
        |-- output3.txt
        |-- output4.txt
        |-- output5.txt
        |-- output6.txt
        |-- output7.txt
        |-- output8.txt
        |-- output9.txt
        --- output10.txt

```

Masalah 1 menggunakan metode "*Tester*" untuk mengecek hasil keluaran, sehingga memiliki *file tester.cpp*. Berikut isi dari *file tester.cpp* untuk masalah 1:

```
/*
* tester.cpp

```

```

*/
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main(int argc, char const *argv[])
{
    ifstream test_in(argv[1]); /* Stream ini membaca
    isi file input */
    ifstream test_out(argv[2]); /* Stream ini membaca
    isi file output */
    ifstream user_out(argv[3]); /* Stream ini membaca
    isi keluaran users */

    /* Kode Pengguna */
    /* Jika keluaran kode user benar, mengembalikan nilai 0,
    sebaliknya mengembalikan 1 */
    /* e.g.: Permasalahan: membaca n nomor dan keluarkan
    hasil penjumlahannya: */

    int sum, user_output;
    user_out >> user_output;

    if ( test_out.good() ) // if test's output file exists
    {
        test_out >> sum;
    }
    else
    {
        int n, a;
        sum=0;
        test_in >> n;
        for (int i=0 ; i<n ; i++){
            test_in >> a;
            sum += a;
        }
    }

    if (sum == user_output)
        return 0;
    else
        return 1;
}

```

Masalah 2 menggunakan metode *"Input/Output Comparison"* untuk mengecek hasil keluaran, sehingga memiliki dua folder *in* dan *out* yang berisikan *test cases*. Masalah 3 menggunakan pilihan *Upload-Only*, sehingga tidak memiliki folder apapun.

2.2.6 Deteksi Kecurangan

Sharif Judge menggunakan *Moss* untuk mendeteksi kode yang mirip. *Moss (Measure Of Software Similarity)* merupakan sistem otomatis untuk menentukan kemiripan program. Pada saat ini, aplikasi utama *Moss* telah digunakan untuk mendeteksi plagiarisme pada kelas *programming*. Pengguna dapat mengirimkan kode final (yang dipilih oleh *students* sebagai *Final Submission*) ke server *Moss* dengan satu klik [3].

Sebelum menggunakan *Moss* ada beberapa hal yang harus diperhatikan yaitu:

- Pengguna harus mendapatkan *Moss user id* dan mengaturnya di *Sharif Judge*. Untuk mendapatkan *Moss user id*, pengguna harus terlebih dahulu daftar pada halaman <http://theory.stanford.edu/aiken/moss/>. Pengguna akan mendapatkan sebuah *email* yang berisikan *script perl*. *Moss user id* berada pada *script* tersebut. Berikut potongan *perl script* yang berisikan user id:

```
...

$server = 'moss.stanford.edu';
$port = '7690';
$noreq = "Request not sent.";
$usage = "usage: moss [-x] [-l language] [-d]
           [-b basefile1] ... [-b basefileN] [-m #]
           [-c \"string\"] file1 file2 file3 ...";

#
# The userid is used to authenticate your queries to the server;
# don't change it!
#
$userid=YOUR_MOSS_USER_ID;

#
# Process the command line options. This is done in a non-standard
# way to allow multiple -b's.
#
$opt_l = "c"; # default language is c
$opt_m = 10;
$opt_d = 0;

...
}
```

- Dapatkan *user id* tersebut lalu gunakan pada *Sharif Judge* untuk mendeteksi kecurangan. Pengguna dapat menyimpan *user id* di *Sharif Judge* pada halaman *Moss* dan *Sharif Judge* akan menggunakan *user id* tersebut di *Moss perl script*.
- Server pengguna harus menginstal *perl* untuk menggunakan *Moss*.
- Pengguna dianjurkan untuk mendetek kode yang mirip setelah waktu tugas berakhir, karena *students* masih dapat mengubah *Final Submissions* mereka sebelum waktu habis. Dengan cara tersebut *Sharif Judge* dapat mengirimkan *Final submissions student* ke *Moss*.

BAB 3

ANALISIS

Bab ini membahas tentang analisis kebutuhan Sharif Judge yang diperlukan oleh Teknik Informatika dan solusi yang ditawarkan untuk memenuhi kebutuhan tersebut. Kebutuhan-kebutuhan tersebut didapat dari daftar isu repositori Sharif Judge di *GitHub* dan dari para dosen pengguna Sharif Judge. Hasil dari analisis kebutuhan tersebut dicatat ke dalam *Google Sheets* dan didiskusikan dengan dosen pembimbing. Selain analisis kebutuhan, pada bab ini juga akan dibahas analisis sistem terkini pada perangkat lunak Sharif Judge.

3.1 Analisis Kebutuhan Perangkat Lunak Sharif Judge

Analisis dilakukan dengan menganalisis setiap isu terbuka yang ada pada repositori di GitHub. Setiap isu di GitHub terdapat kode unik yang dimulai dengan tanda '#' dan diikuti dengan angka. Kode unik tersebut menunjukkan urutan isu yang dibuat oleh para pengguna GitHub. Dari analisa setiap isu tersebut, didapatkan beberapa pertanyaan dan usulan pengembangan. Beberapa isu yang memiliki usulan pengembangan akan dijadikan pertimbangan untuk mengembangkan Sharif Judge.

Analisis kebutuhan dari para dosen pengguna Sharif Judge dilakukan dalam bentuk wawancara secara langsung dan melalui email. Dosen-dosen yang telah diwawancarai antara lain:

1. Husnul Hakim
2. Claudio Franciscus
3. Vania Natali
4. Luciana Abednego
5. Joanna Helga

Tabel 3.1: *Kebutuhan Sharif Judge*

No	Deskripsi	Sumber	Issue Number / Nama Mata Kuliah	Pembuat Issue Nama Dosen	Status
1	Security with PHP	GitHub	#61	kathiedart	Diimplemntasi
2	Securing Assignment	GitHub	#55	wojcik13	Tidak diimplementasi
3	New Function	GitHub	#53	wojcik13	Tidak diimplementasi
4	Solved Problem Indicator	GitHub	#46	atiabjobayer	Tidak diimplementasi
5	Some Problem & Sugestion	GitHub	#45	atiabjobayer	Tidak diimplementasi
6	Queue failed to process if submission take too long to complete?	GitHub	#32	truongan	Diimplemntasi
7	Compilation Error on all language	GitHub	#34	Eririn07	Tidak diimplementasi
8	Membatasi soal (deskripsi dan PDF) hanya bisa diakses saat assignment "open" dan setelah waktu mulai	Dosen	ASD	Husnul Hakim	Diimplementasi
9	Menguji kemiripan kode antar mahasiswa (Contek)	Dosen	ASD	Husnul Hakim	Tidak diimplementasi
10	1 Akun hanya dapat login 1 waktu (Jika suatu akun sedang login, tidak ada lagi yang bisa login akun tersebut)	Dosen	ASD	Husnul Hakim	Diimplementasi
11	asd	Dosen	ASD	Vania Natali	Diimplementasi
12	asd	Dosen	ASD	Vania Natali	Diimplementasi
13	asd	Dosen	DAA	Luciana Abednego	Diimplementasi
14	asd	Dosen	DAA	Luciana Abednego	Diimplementasi
15	asd	Dosen	DAA	Luciana Abednego	Tidak diimplementasi
16	asd	Dosen	DAA	Joanna Helga	Diimplementasi
17	asd	Dosen	DAA	Joanna Helga	Diimplementasi
18	asd	Dosen	DAA	Joanna Helga	Diimplementasi
19	asd	Dosen	DAA	Joanna Helga	Diimplementasi
20	asd	Dosen	ASD & DAA	Joanna Helga	Diimplementasi
21	asd	Dosen	ASD & DAA	Claudio Fransiscus	Diimplementasi
22	asd	Dosen	ASD & DAA	Claudio Fransiscus	Diimplementasi
23	asd	Dosen	ASD & DAA	Claudio Fransiscus	Tidak diimplementasi
24	asd	Dosen	ASD & DAA	Claudio Fransiscus	Tidak diimplementasi
25	asd	Dosen	ASD & DAA	Claudio Fransiscus	Tidak diimplementasi
26	asd	Dosen	ASD & DAA	Pascal Alfadian	Diimplementasi
27	asd	Dosen	AIF401	Pascal Alfadian	Diimplementasi
28	asd	Dosen	AIF401	Pascal Alfadian	Diimplementasi
29	asd	Dosen	AIF401	Pascal Alfadian	Diimplementasi
30	asd	Dosen	AIF401	Pascal Alfadian	Diimplementasi

3.1.1 *Security with PHP*

Isu ini dibuat oleh pengguna *GitHub* dengan username *danwdart*. Pada isu tersebut dikatakan bahwa seseorang pengguna Sharif Judge dapat mengubah parameter PHP `shell_exec()` yang mengakibatkan pengeksekusian kode bisa dilakukan secara sewenang-wenang. Solusi yang ditawarkan untuk mengatasi hal diatas adalah mengganti penggunaan PHP `shell_exec()` dengan method lain. Contohnya seperti perintah `shell_exec("rm ...")` yang memiliki fungsi untuk menghapus sebuah file. Perintah tersebut dapat diganti menggunakan method `unlink()` yang memiliki fungsi sama yaitu menghapus sebuah file.

3.1.2 *Queue failed to process if submission take too long to complete?*

Isu ini dibuat oleh pengguna *GitHub* dengan username *truongan*. Pada isu tersebut dikatakan bahwa assignment yang memiliki masalah dengan test case besar akan menyebabkan *submission status* menjadi *pending*. User *truongan* memperkirakan hal diatas terjadi dikarenakan *database connection times out*. Solusi yang ditawarkan untuk mengatasi hal diatas adalah menambahkan

method reconnect database. *Method reconnect database* akan menghubungkan kembali database ketika terjadi kasus seperti diatas.

3.1.3 Menguji kemiripan kode antar mahasiswa

Perangkat lunak Sharif Judge terkini sudah memiliki fitur untuk menguji kemiripan kode antar peserta dengan menggunakan Moss (*Measure Of Software Similarity*). Moss adalah sistem otomatis untuk mendeteksi kemiripan program. Aplikasi Moss telah berkembang dari tahun 1994 hingga sekarang. Algoritma yang digunakan pada aplikasi Moss sangat efektif dibandingkan algoritma deteksi kecurangan lainnya [4]. Namun untuk sekarang ini, Moss tidak dapat digunakan karena aplikasi Moss membutuhkan port 7690 yang diblok oleh UNPAR. Oleh sebab itu, kebutuhan ini tidak diimplementasikan.

3.1.4 Satu Akun hanya dapat login satu waktu

Para peserta Sharif Judge dapat login menggunakan akunnya di beberapa komputer. Peserta yang mengetahui user dan password peserta lain dapat dengan mudah login ke Sharif Judge. Hal tersebut sering dijadikan celah bagi beberapa peserta untuk melakukan tindak kecurangan. Peserta yang sudah login menggunakan akun peserta lainnya, dapat melihat dan menyalin kode yang telah dikumpulkan ke Sharif Judge. Tindak kecurangan ini sering terjadi pada saat kuis maupun ujian. Bapak Husnul Hakim menginginkan perangkat lunak Sharif Judge dimana akun para peserta hanya dapat login satu waktu. Jika sebuah akun telah login di satu komputer, maka akun tersebut tidak dapat login di komputer lainnya. Diharapkan dengan adanya fitur tersebut dapat menekan jumlah tindak kecurangan yang terjadi.

Solusi yang ditawarkan untuk mengurangi tingkat kecurangan seperti diatas yaitu membuat halaman baru yang berisikan *logs* username yang berhasil login ke Sharif Judge. Halaman logs tersebut akan mencatat username yang login menggunakan ip berbeda dalam waktu dibawah 24 jam. Dengan adanya halaman Logs ini, para dosen dapat memantau username yang login pada dua tempat berbeda.

3.1.5 Perlu ditambah petunjuk penamaan file input dan output

Dalam membuat sebuah assignment pada perangkat lunak Sharif Judge terdapat file test case yang harus disertakan. File test case yang disertakan memiliki beberapa ketentuan. Beberapa ketentuan tersebut seperti struktur direktori dan penamaan dalam file test case. Pada halaman add assignment telah disediakan link menuju dokumentasi Sharif Judge di *GitHub* yang menjelaskan ketentuan dalam menyertakan file test case. Ketentuan tersebut harus terpenuhi agar sebuah assignment dapat berjalan dengan baik. Oleh sebab itu, kebutuhan ini tidak diimplementasikan.

3.1.6 Sharif Judge diharapkan memiliki fungsi reminder

Setiap assignment pada perangkat lunak Sharif Judge memiliki batas pengumpulan. Jika assignment telah melewati batas pengumpulan maka para peserta tidak dapat mengumpulkan tugasnya. Banyak peserta sering kali lupa untuk mengerjakan assignment dan pada akhirnya melewati batas pengumpulan. Bapak Claudio Fransiscus menginginkan perangkat lunak Sharif Judge yang memiliki fitur reminder. Fitur reminder akan mengirimkan email ke setiap peserta yang berisikan peringatan bahwa ada assignment yang harus diselesaikan. Namun kebutuhan ini belum dapat diimplementasikan karena masih belum ada sistem *scheduler* dan tingkat kesulitan yang terlalu tinggi.

3.1.7 Membatasi soal (deskripsi & PDF) hanya bisa diakses saat assignment "open" dan setelah waktu mulai

Kebutuhan ini merupakan salah satu kebutuhan yang paling banyak disebut oleh para dosen pengguna Sharif Judge. Perangkat lunak Sharif Judge terkini masih belum dapat memenuhi kebutuhan diatas. Para peserta dapat mengunduh deskripsi soal & PDF sebelum waktu assignment dimulai. Untuk menangani hal tersebut para dosen harus mengunggah file PDF tepat pada saat assignment dimulai.

Solusi yang ditawarkan untuk memenuhi kebutuhan diatas yaitu membatasi soal agar dapat diunduh pada saat assignment "open" dan setelah waktu mulai. Jika ada peserta yang mencoba untuk mengunduh soal (deskripsi & PDF) pada saat assignment belum dimulai, maka akan muncul pesan error "*Selected assignment has not started*". Deskripsi & PDF hanya dapat diunduh tepat setelah waktu assignment dimulai.

3.1.8 Mengumpulkan file dengan format TXT

Pengumpulan file dengan format TXT dibutuhkan untuk *Pre-test*. Perangkat lunak Sharif Judge yang sekarang hanya dapat menerima file C, C++, Java, Python 2, Python 3, Zip, dan PDF. Para peserta yang ingin mengumpulkan jawaban *Pre-test*, harus terlebih dahulu mengubah ekstensi file menjadi Java atau mengompres file ke dalam Zip.

Solusi yang ditawarkan untuk kebutuhan diatas yaitu menambahkan file dengan format TXT agar dapat dikumpul ke Sharif Judge. Assignment yang digunakan merupakan assignment yang bersifat "Upload Only". Dosen dapat menambahkan format TXT pada bagian "Allowed Language" sehingga para peserta dapat mengumpulkan jawaban *Pre-test* menggunakan file dengan ekstensi TXT.

3.1.9 Mengumpulkan file JAR (java multi kelas)

JAR (Java ARchive) adalah format file platform-independen yang menggabungkan banyak file menjadi satu. File-file seperti kelas, gambar dan suara dapat digabungkan dalam file JAR. ==On Progress==

3.1.10 Pendaftaran peserta disertai dengan *Display Name*

Pendaftaran peserta ke dalam Sharif Judge terkini tidak disertai dengan *Display Name*. Perangkat lunak Sharif Judge membutuhkan empat buah parameter yang dipisah menggunakan spasi untuk mendaftarkan peserta. Parameter tersebut antara lain, username, email, password dan role. Contoh penggunaannya seperti "i14085 i14085@unpar.ac.id random85 student" yang artinya peserta didaftarkan menggunakan username i14085, email i14085@unpar.ac.id, password random85 dan role sebagai student. Setiap peserta yang berhasil didaftarkan masih belum memiliki *Display Name*. Para peserta harus memasukan *Display Name* masing-masing secara manual.

Solusi yang ditawarkan untuk memenuhi kebutuhan diatas yaitu menambahkan parameter *Display Name* pada saat pendaftaran peserta. Parameter yang digunakan akan menjadi lima buah parameter dan dipisah menggunakan tanda koma. Parameter tersebut antara lain, username, email, display name, password, dan role. Contoh penggunaan parameter diatas seperti "i14085,i14085@unpar.ac.id,Budi Simon,random85,student" yang artinya peserta didaftarkan menggunakan username i14085, email i14085@unpar.ac.id, display name Budi Simon, password random85 dan role sebagai student. Dengan pengimplementasian fitur ini, setiap peserta yang didaftarkan akan langsung memiliki Display Name masing-masing.

3.1.11 Nama pengguna Sharif Judge seharusnya tidak bisa diubah

Display Name pada perangkat lunak Sharif Judge berfungsi sebagai nama peserta. Selain itu, nama peserta akan muncul pada halaman Scoreboard sebuah assignment yang dapat dilihat oleh seluruh

peserta. Sharif Judge yang terkini mengizinkan para peserta untuk mengubah *Display Name* pada halaman Profile. Hal tersebut dapat dijadikan sebuah "mainan" dan tindakan kecurangan karena dapat memberikan *hint* untuk peserta lain. Oleh karena itu, Ibu Joanna Helga menginginkan nama peserta yang terdaftar pada Sharif Judge tidak dapat diubah.

Solusi yang ditawarkan untuk memenuhi kebutuhan diatas yaitu menambahkan sebuah fitur dimana fitur tersebut dapat mengunci *Display Name* peserta Sharif Judge. Fitur ini akan diletakan pada halaman Settings yang dapat diatur oleh admin. Jika admin mengaktifkan fitur tersebut, maka *input text Display Name* pada halaman Profile menjadi nonaktif sehingga para peserta tidak dapat mengubah *Display Name*. Sebaliknya jika admin menonaktifkan fitur tersebut, maka *input text Display Name* pada halaman Profile akan kembali aktif.

3.1.12 Sharif Judge diharapkan memiliki fungsi dimana assignment dapat dikumpulkan tanpa adanya batasan waktu

Pada masa Pra UTS dan Pra UAS biasanya para dosen akan memberikan assignment sebagai bahan pembelajaran. Arsip-arsip soal ujian dan latihan tahun lalu akan dijadikan sebuah assignment yang dapat dikerjakan oleh para peserta. Assignment tersebut memiliki waktu pengumpulan yang cenderung lama. Ibu Joanna Helga menginginkan sebuah fitur dimana Sharif Judge dapat mengatur assignment tertentu menjadi tidak memiliki batasan waktu dan dapat dikumpulkan kapan saja.

Solusi yang ditawarkan untuk memenuhi kebutuhan diatas yaitu membuat sebuah fitur tambahan pada assignment. Fitur tersebut akan membuat batas waktu pengumpulan menjadi tanggal 18 Januari 2038. Assignment yang mengaktifkan fitur tidak akan muncul pada kalender yang terdapat di halaman Dashboard.

3.1.13 Integrasi login ke RADIUS

RADIUS (*Remote Authentication Dial In User Service*) merupakan protokol jaringan klien dan server. Klien mengirimkan informasi pengguna ke server RADIUS yang ditunjuk dan akan bertindak berdasarkan respons yang dikembalikan. Server RADIUS akan menerima permintaan koneksi pengguna, mengautentikasi pengguna dan kemudian mengembalikan informasi konfigurasi yang diperlukan agar klien dapat memberikan layanan kepada pengguna. Server RADIUS dapat bertindak sebagai klien proxy ke server RADIUS lain atau server autentikasi jenis lainnya¹. Lab FTIS UNPAR memiliki server RADIUS yang dapat memverifikasi ID mahasiswa terhadap kata sandinya. Server RADIUS juga berguna untuk autentikasi ID mahasiswa agar menggunakan komputer di Lab FTIS UNPAR. Dengan pengimplementasian integrasi login RADIUS pada Sharif Judge, para peserta dapat login ke Sharif Judge menggunakan akun yang terdapat pada server RADIUS.

3.1.14 Branding Teknik Informatika

Branding Teknik Informatika akan dilakukan dengan cara mengubah logo dan ikon Sharif Judge menjadi logo Teknik Informatika. Hal tersebut dapat dilakukan karena Sharif Judge sendiri menggunakan lisensi GPL versi 3. GPL merupakan kepanjangan dari General Public License yang memberikan beberapa kebebasan pada setiap penggunaanya [5]. Kebebasan tersebut antara lain:

- Kebebasan untuk menggunakan perangkat lunak dengan tujuan apapun
- Kebebasan untuk mengubah perangkat lunak sesuai dengan kebutuhan

¹Cisco, "How Does RADIUS Work?" How Does RADIUS Work? - Cisco.
<https://www.cisco.com/c/en/us/support/docs/security-vpn/remote-authentication-dial-user-service-radius/12433-32.html> (diakses 22 Februari 2018)

- Kebebasan untuk membagikan perangkat lunak kepada teman dan kerabat
- Kebebasan untuk membagikan perubahan yang telah dilakukan

3.1.15 Sharif Judge diharapkan memiliki Scoreboard global untuk semua assignment

Sharif Judge terkini memiliki halaman Scoreboard yang berfungsi menampilkan seluruh nilai akhir para peserta dari sebuah assignment. Pada halaman Scoreboard juga menampilkan nilai dari setiap problem yang ada pada sebuah assignment. Nilai yang muncul pada halaman ini adalah nilai para peserta yang telah mengumpulkan jawabannya. Nilai yang muncul tersebut akan diurutkan mulai dari yang tertinggi hingga terendah. Para dosen menginginkan sebuah Scoreboard global untuk semua assignment. Scoreboard global tersebut akan menampilkan berapa problem yang telah dikerjakan para peserta diseluruh assignment yang ada.

Solusi yang ditawarkan untuk memenuhi kebutuhan diatas yaitu membuat halaman baru yang diberi nama Hall of Fame. Halaman Hall of Fame akan menampilkan berapa problem yang telah dikerjakan oleh para peserta diseluruh assignment yang ada. Nama peserta yang muncul pada halaman ini diurutkan sesuai dengan banyaknya problem yang berhasil diselesaikan oleh para peserta.

DAFTAR REFERENSI

- [1] Naderi, M. J. (2014) Sharif judge. <https://github.com/mjnaderi/Sharif-Judge/>. 6 Oktober 2017.
- [2] of Technology, B. C. I. (2017) Codeigniter documentation. https://codeigniter.com/user_guide/overview/index.html. 6 Oktober 2017.
- [3] Naderi, M. J. (2014) Sharif judge documentation. <https://github.com/mjnaderi/Sharif-Judge/tree/docs/v1.4>. 6 Oktober 2017.
- [4] Aiken, A. (2010) A system for detecting software similarity. <http://theory.stanford.edu/~aiken/moss/>. 25 Februari 2018.
- [5] Smith, B. (2009) The gnu operating system and the free software movement. <https://www.gnu.org/>. 22 Februari 2018.

LAMPIRAN A

KODE PROGRAM

Listing A.1: MyCode.c

```
1 // This does not make algorithmic sense,
2 // but it shows off significant programming characters.
3
4 #include<stdio.h>
5
6 void myFunction( int input, float* output ) {
7     switch ( array[i] ) {
8         case 1: // This is silly code
9             if ( a >= 0 || b <= 3 && c != x )
10                 *output += 0.005 + 20050;
11             char = 'g';
12             b = 2^n + ~right_size - leftSize * MAX_SIZE;
13             c = (--aaa + &daa) / (bbb++ - ccc % 2 );
14             strcpy(a,"hello_$@?");
15         }
16         count = ~mask | 0x00FF00AA;
17     }
18 }
19
20 // Fonts for Displaying Program Code in LATEX
21 // Adrian P. Robson, nepsweb.co.uk
22 // 8 October 2012
23 // http://nepsweb.co.uk/docs/progfonts.pdf
```

Listing A.2: MyCode.java

```
1 import java.util.ArrayList;
2 import java.util.Collections;
3 import java.util.HashSet;
4
5 //class for set of vertices close to furthest edge
6 public class MyFurSet {
7     protected int id; //id of the set
8     protected MyEdge FurthestEdge; //the furthest edge
9     protected HashSet<MyVertex> set; //set of vertices close to furthest edge
10    protected ArrayList<ArrayList<Integer>> ordered; //list of all vertices in the set for each trajectory
11    protected ArrayList<Integer> closeID; //store the ID of all vertices
12    protected ArrayList<Double> closeDist; //store the distance of all vertices
13    protected int totaltrj; //total trajectories in the set
14
15    /*
16     * Constructor
17     * @param id : id of the set
18     * @param totaltrj : total number of trajectories in the set
19     * @param FurthestEdge : the furthest edge
20     */
21    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
22        this.id = id;
23        this.totaltrj = totaltrj;
24        this.FurthestEdge = FurthestEdge;
25        set = new HashSet<MyVertex>();
26        ordered = new ArrayList<ArrayList<Integer>>();
27        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
28        closeID = new ArrayList<Integer>(totaltrj);
29        closeDist = new ArrayList<Double>(totaltrj);
30        for (int i = 0;i <totaltrj;i++) {
31            closeID.add(-1);
32            closeDist.add(Double.MAX_VALUE);
33        }
34    }
35
36 }
```


LAMPIRAN B

HASIL EKSPERIMEN

Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4