

**SKRIPSI**

**KONVERSI SHARIF JUDGE DARI CODEIGNITER 3 KE  
CODEIGNITER 4**



**Filipus**

**NPM: 6181901074**

**PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS  
UNIVERSITAS KATOLIK PARAHYANGAN  
2023**



# DAFTAR ISI

DAFTAR ISI	iii
DAFTAR GAMBAR	v
<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 Metodologi . . . . .	2
1.6 Sistematika Pembahasan . . . . .	3
<b>2 LANDASAN TEORI</b>	<b>5</b>
2.1 CodeIgniter 3[1] . . . . .	5
2.1.1 <i>Model-View-Controller</i> . . . . .	5
2.1.2 <i>CodeIgniter URLs</i> . . . . .	7
2.1.3 <i>Helpers</i> . . . . .	8
2.1.4 <i>Libraries</i> . . . . .	9
2.1.5 <i>Database</i> . . . . .	12
2.1.6 <i>URI Routing</i> . . . . .	14
2.1.7 <i>Auto-loading</i> . . . . .	15
2.2 SharIF Judge[2] . . . . .	15
2.2.1 Struktur Aplikasi . . . . .	15
2.2.2 Instalasi . . . . .	15
2.2.3 <i>Clean URLs</i> . . . . .	16
2.2.4 Users . . . . .	17
2.2.5 Menambah <i>Assignment</i> . . . . .	18
2.2.6 <i>Sample Assignment</i> . . . . .	21
2.2.7 <i>Test Structure</i> . . . . .	24
2.2.8 Deteksi Kecurangan . . . . .	26
2.2.9 Keamanan . . . . .	27
2.2.10 <i>Sandboxing</i> . . . . .	28
2.2.11 <i>Shield</i> . . . . .	29
2.3 CodeIgniter 4[3] . . . . .	30
2.3.1 <i>Models-Views-Controllers</i> . . . . .	31
2.3.2 <i>CodeIgniter URLs</i> . . . . .	34
2.3.3 <i>URI Routing</i> . . . . .	34
2.3.4 <i>Databases</i> . . . . .	35
2.3.5 <i>Library</i> . . . . .	37
2.3.6 <i>Helpers</i> . . . . .	39
2.4 Koversi CodeIgniter 3 ke CodeIgniter 4[3] . . . . .	39
2.4.1 Struktur Aplikasi . . . . .	39
2.4.2 <i>Routing</i> . . . . .	39

2.4.3	<i>Model, View, and Controller</i>	40
2.4.4	<i>Libraries</i>	41
2.4.5	<i>Helpers</i>	43
2.4.6	<i>Events</i>	43
2.4.7	<i>Framework</i>	44
<b>3</b>	<b>ANALISIS</b>	<b>45</b>
3.1	Analisis konversi menuju <i>CodeIgniter 4</i>	45
3.1.1	Persiapan <i>CodeIgniter 4</i>	45
3.1.2	Pemindahan <i>file</i> dari <i>CodeIgniter 3</i> ke <i>CodeIgniter 4</i>	45
3.1.3	Pembaharuan <i>file-file CodeIgniter 3</i>	45
3.1.4	Struktur Aplikasi	45
3.1.5	<i>Routing</i>	47
3.1.6	<i>Model, View, and Controller</i>	47
3.1.7	<i>Libraries</i>	49
3.1.8	<i>Configuration</i>	51
3.1.9	<i>Database</i>	51
3.1.10	<i>Helpers</i>	51
3.2	Analisis Sistem Kini	52
3.2.1	<i>Model</i>	52
3.2.2	<i>View</i>	57
3.2.3	Halaman pada <i>SharIF Judge</i>	57
3.2.4	<i>View</i>	57
3.2.5	<i>Controller</i>	66
3.3	Analisis Sistem Usulan	71
3.3.1	<i>Model</i>	71
3.3.2	<i>View</i>	71
3.3.3	<i>Controller</i>	71
<b>4</b>	<b>PERANCANGAN</b>	<b>73</b>
4.1	Perancangan Konversi Menuju <i>CodeIgniter 4</i>	73
4.1.1	Implementasi Persiapan <i>CodeIgniter 4</i>	73
4.1.2	Pemindahan <i>file</i> dari <i>CodeIgniter 3</i> ke <i>CodeIgniter 4</i>	74
4.1.3	Pembaharuan <i>file-file CodeIgniter 3</i>	74
4.2	Implementasi Analisis Konversi <i>CodeIgniter 3</i> ke <i>CodeIgniter 4</i>	74
	<b>DAFTAR REFERENSI</b>	<b>75</b>
	<b>A KODE PROGRAM</b>	<b>77</b>
	<b>B HASIL EKSPERIMEN</b>	<b>79</b>

## DAFTAR GAMBAR

1.1	Tampilan halaman <i>SharIF Judge</i> . . . . .	1
2.1	<i>Flow Chart</i> Aplikasi <i>CodeIgniter 3</i> . . . . .	5
2.2	Tampilan halaman <i>SharIF Judge</i> untuk menambahkan <i>assignment</i> . . . . .	18
3.1	<i>Pemindahan struktur aplikasi menuju CodeIgniter 4</i> . . . . .	46
3.2	Tampilan Halaman <i>Dashboard</i> . . . . .	58
3.3	Tampilan Halaman <i>Profile</i> . . . . .	58
3.4	Tampilan Halaman <i>Settings</i> . . . . .	59
3.5	Tampilan Halaman <i>Users</i> . . . . .	59
3.6	Tampilan Halaman <i>Notifications</i> . . . . .	60
3.7	Tampilan Halaman <i>Assignments</i> . . . . .	60
3.8	Tampilan Halaman <i>Problems</i> . . . . .	61
3.9	Tampilan Halaman <i>Submit</i> . . . . .	61
3.10	Tampilan Halaman <i>Final Submission</i> . . . . .	62
3.11	Tampilan Halaman <i>All Submission</i> . . . . .	62
3.12	Tampilan Halaman <i>Scoreboard</i> . . . . .	63
3.13	Tampilan Halaman <i>Hall of Fame</i> . . . . .	63
3.14	Tampilan Halaman <i>24-hour Log</i> . . . . .	64
3.15	Tampilan Halaman <i>ReJudge</i> . . . . .	64
3.16	Tampilan Halaman <i>Submission Queue</i> . . . . .	65
3.17	Tampilan Halaman <i>Cheat Detection</i> . . . . .	65
B.1	Hasil 1 . . . . .	79
B.2	Hasil 2 . . . . .	79
B.3	Hasil 3 . . . . .	79
B.4	Hasil 4 . . . . .	79





1 *framework* CodeIgniter 3 yang merupakan *framework* berbasis PHP (*Hypertext Preprocessor*) dan  
2 dimodifikasi sesuai dengan kebutuhan Informatika Unpar untuk mengumpulkan tugas dan ujian  
3 mahasiswa[2].

4 CodeIgniter 3 merupakan sebuah *framework opensource* yang bertujuan untuk mempermudah  
5 dalam membangun sebuah aplikasi *website* menggunakan PHP. CodeIgniter 3 menggunakan struktur  
6 MVC yang membagi file menjadi 3 buah yaitu Model, View, Controller. Selain itu, CodeIgniter 3  
7 merupakan *framework* ringan dan menyediakan banyak *library* untuk digunakan oleh penggunanya[1].  
8 Namun, CodeIgniter 3 sudah memasuki fase *maintenance*<sup>1</sup> sehingga tidak akan mendapatkan *update*  
9 lebih lanjut dari pembentuknya. CodeIgniter 3 pada akhirnya akan tidak dapat dipakai dan  
10 akan hilangnya dokumentasi dari situs web resmi. Sehingga, perangkat lunak yang menggunakan  
11 CodeIgniter 3 perlu dikonversi ke *framework* CodeIgniter dengan versi terbaru yakni CodeIgniter 4.

12 CodeIgniter 4 merupakan versi terbaru dari *framework* CodeIgniter yang memiliki banyak  
13 perubahan fitur dari versi sebelumnya. CodeIgniter 4 dapat dijalankan menggunakan versi PHP 7.4  
14 atau lebih baru sedangkan CodeIgniter 3 dapat dijalankan menggunakan versi PHP 5.6 atau lebih  
15 baru. CodeIgniter 4 juga membagi file menggunakan struktur MVC namun, memiliki struktur  
16 folder berbeda dengan versi sebelumnya[3].

17 Pada skripsi ini, akan dilakukan konversi SharIF Judge dari CodeIgniter 3 sehingga dapat  
18 berjalan pada CodeIgniter 4.

## 19 1.2 Rumusan Masalah

- 20 • Bagaimana cara melakukan konversi SharIF Judge pada CodeIgniter 3 menjadi CodeIgniter  
21 4?
- 22 • Bagaimana mengevaluasi kode SharIF Judge dan mengubahnya agar dapat berjalan di Code-  
23 Igniter 4?

## 24 1.3 Tujuan

25 Tujuan dari skripsi ini adalah sebagai berikut:

- 26 • Melakukan konversi dengan mengubah kode sesuai dengan CodeIgniter 4.
- 27 • Melakukan evaluasi kode SharIF Judge dan mengubahnya agar dapat berjalan di CodeIgniter  
28 4.

## 29 1.4 Batasan Masalah

## 30 1.5 Metodologi

31 Metodologi yang dilakukan dalam melakukan penelitian ini adalah sebagian berikut:

- 32 1. Melakukan analisis dan eksplorasi fungsi-fungsi perangkat lunak SharIF Judge.
- 33 2. Melakukan studi literatur kebutuhan konversi dari CodeIgniter 3 menjadi CodeIgniter 4.
- 34 3. Melakukan konversi perangkat lunak dari CodeIgniter 3 menjadi CodeIgniter 4.

---

<sup>1</sup>Pemberitahuan fase *maintenance* CodeIgniter 3 <https://codeigniter.com/download>(19 Maret 2023)



4. Melakukan pengujian dan eksperimen terhadap perangkat lunak yang sudah di konversi.
5. Menyelesaikan pembentukan dokumen

## 1.6 Sistematika Pembahasan

Penelitian ini akan dibahas dalam enam bab yang masing-masing berisi:

1. **Bab 1:** Pendahuluan

Bab ini berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi, dan sistematika pembahasan.

2. **Bab 2:** Landasan Teori

Bab ini berisi pembahasan dasar-dasar teori yang akan digunakan dalam melakukan konversi SharIF Judge dari CodeIgniter 3 ke CodeIgniter 4. Landasan Teori yang digunakan diantaranya adalah SharIF Judge, CodeIgniter 3, CodeIgniter 4, dan Konversi CodeIgniter 3 ke CodeIgniter 4.

3. **Bab 3:** Analisis

Bab ini berisi analisis *SharIF Judge* dan analisis kebutuhan konversi menuju *CodeIgniter 3*.

4. **Bab 4:** Perancangan

Bab ini berisikan mengenai rancangan yang perangkat lunak yang akan dikonversi.

5. **Bab 5:** Implementasi dan Pengujian

Bab ini berisikan hasil implementasi dan pengujian yang telah dilakukan untuk melakukan konversi *SharIF Judge* dari *CodeIgniter 3* ke *CodeIgniter 4*.

6. **Bab 6:** Kesimpulan dan Saran

Bab ini berisikan kesimpulan dari hasil konversi yang telah dilakukan dan saran-saran terhadap perangkat lunak.

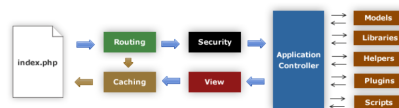


## BAB 2

### LANDASAN TEORI

#### 2.1 CodeIgniter 3[1]

CodeIgniter 3 merupakan sebuah *framework opensource* yang berfungsi untuk mempermudah pengguna dalam membangun aplikasi *website* menggunakan bahasa PHP. CodeIgniter 3 memiliki tujuan untuk membantu pengguna dalam membangun aplikasi web lebih cepat dengan menyediakan beragam *library* dan tampilan dan *logic* yang simpel. *CodeIgniter 3* juga merupakan *framework* ringan yang menggunakan struktur *Model-View-Controller*, dan menghasilkan *URLs* yang bersih. *Code Igniter 3* memiliki *flow chart* aplikasi yang dapat dilihat pada gambar 2.1.



Gambar 2.1: *Flow Chart* Aplikasi *CodeIgniter 3*

Berikut merupakan pembagian *flow chart* aplikasi *CodeIgniter 3*:

1. `index.php` berfungsi sebagai *front controller* yang berguna untuk melakukan inisiasi
2. *Router* berfungsi dalam melakukan pemeriksaan dan menentukan penggunaan *HTTP Request*.
3. *Cache* berfungsi untuk mengirimkan *file cache*(apabila ada) kepada *browser* secara langsung.
4. *Security* berfungsi sebagai alat penyaringan setiap data dan *HTTP Request* yang masuk. Penyaringan data tersebut dilakukan sebelum *controller* aplikasi dimuat agar aplikasi menjadi lebih aman.
5. *Controller* berguna sebagai alat untuk memuat *model*, *libraries*, dan sumber daya yang dibutuhkan untuk menjalankan permintaan spesifik.
6. *View* akan dikirimkan menuju *browser* untuk dilihat oleh pengguna. Apabila *caching* dinyalakan, maka *view* akan dilakukan *cached* terlebih dahulu sehingga permintaan selanjutnya dapat diberikan.

##### 2.1.1 Model-View-Controller

*CodeIgniter 3* merupakan *framework* berbasis arsitektur *Model-View-Controller* atau yang selanjutnya akan disebut sebagai MVC. MVC merupakan sebuah pendekatan perangkat lunak yang memisahkan antara logika dengan presentasi atau tampilannya. Penggunaan struktur ini mengurangi penggunaan skrip pada halaman web karena tampilan terpisah dengan skrip PHP. Berikut merupakan penjelasan mengenai struktur MVC:

## Model

*Model* berfungsi dalam mewakili struktur data perangkat lunak. *Model* berfungsi dalam mengambil, memasukan, dan memperbarui data pada *database*. Berikut merupakan contoh *file Model CodeIgniter 3* pada direktori `application/models/`:

Kode 2.1: Contoh *model* pada *CodeIgniter 3*

```

5 class Blog_model extends CI_Model {
6
7
8     public $title;
9     public $content;
10    public $date;
11
12    public function get_last_ten_entries()
13    {
14        $query = $this->db->get('entries', 10);
15        return $query->result();
16    }
17
18    public function insert_entry()
19    {
20        $this->title   = $_POST['title']; // please read the below note
21        $this->content = $_POST['content'];
22        $this->date    = time();
23
24        $this->db->insert('entries', $this);
25    }
26
27    public function update_entry()
28    {
29        $this->title   = $_POST['title'];
30        $this->content = $_POST['content'];
31        $this->date    = time();
32
33        $this->db->update('entries', $this, array('id' => $_POST['id']));
34    }
35 }
36
37

```

Kode 2.1 terdapat beberapa fungsi yaitu:

- `get_last_ten_entries()` yang berfungsi untuk mengambil 10 data terakhir dari tabel `entries` menggunakan *query builder*.
- `insert_entry()` yang berfungsi untuk memasukan data *title*, *content*, dan *date* menuju tabel `entries`.
- `update_entry()` yang berfungsi untuk memperbaharui data *title*, *content*, dan *date* pada tabel `entries`.

*Model* biasanya digunakan pada *file controller* dan dapat dipanggil menggunakan:

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

## View

*View* berfungsi dalam menyajikan informasi kepada pengguna. *View* biasanya merupakan halaman web namun, pada *CodeIgniter 3* *view* dapat berupa pecahan halaman seperti *header* atau *footer*. Pecahan halaman dapat dimasukan pada halaman lain agar mempermudah dan membangun kode yang lebih bersih.

Kode 2.2: Contoh *view* pada *CodeIgniter 3*

```

52
53 1 <?php
54 2 <html>
55 3 <head>
56 4     <title>My Blog</title>

```

```

15 </head>
26 <body>
37     <h1>Welcome to my Blog!</h1>
48 </body>
59 </html>

```

Kode 2.2 merupakan contoh *file view CodeIgniter 3* pada direktori `application/views/` yang berisikan judul My Blog dan *heading* Welcome to my Blog!. Pengguna dapat memanggil halaman yang sudah dibentuk pada *file controller* dengan cara sebagai berikut:

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

## 11 **Controller**

*Controller* berfungsi sebagai perantara antara *Model*, *View*, dan sumber daya yang dibutuhkan untuk melakukan proses *HTTP Request* dan menjalankan halaman web. Penamaan *controller* biasanya digunakan sebagai *url* pada perangkat lunak pengguna. Berikut merupakan contoh *controller CodeIgniter 3* pada direktori `application/controllers/`:

Kode 2.3: Contoh *controller* pada *CodeIgniter 3*

```

16 <?php
17 1 class Blog extends CI_Controller {
18 2
19 3
20 4     public function index()
21 5     {
22 6         echo 'Hello World!';
23 7     }
24 8
25 9     public function comments()
26 0     {
27 1         echo 'Look at this!';
28 2     }
29 3 }

```

Kode 2.3 berfungsi dalam mengembalikan *string* sesuai dengan *controller* yang dipanggil. Nama *controller* dan metode diatas akan dijadikan segmen pada *URL* seperti berikut:

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

Metode *index* akan secara otomatis dipanggil menjadi *URL* dan pengguna juga dapat memberi parameter untuk metode *controller* yang nantinya akan menjadi *URL*.

### 36 **2.1.2 CodeIgniter URLs**

*CodeIgniter 3* menggunakan pendekatan *segment-based* dibandingkan menggunakan *query string* untuk membangun *URL* yang mempermudah mesin pencari dan pengguna. Berikut merupakan contoh *URL* pada *CodeIgniter 3*:

```
example.com/news/article/my_article
```

Struktur segmen pada MVC menghasilkan *URL* sebagai berikut :

```
example.com/class/function/ID
```

Segmen tersebut dibagi menjadi tiga buah yakni:

1. Segmen pertama merepresentasikan kelas *controller* yang dipanggil.
2. Segmen kedua merepresentasikan kelas fungsi atau metode yang digunakan.
3. Segmen ketiga dan segmen lainnya merepresentasikan *ID* dari variabel yang akan dipindahkan menuju *controller*.

Secara asali *URL* yang dihasilkan *CodeIgniter 3* terdapat nama file `index.php` seperti contoh dibawah ini:

```
example.com/index.php/news/article/my_article
```

Pengguna dapat menghapus file `index.php` file pada *url* menggunakan file `.htaccess` apabila server *Apache* pengguna menghidupkan `mod_rewrite`. Berikut merupakan contoh file `.htaccess` menggunakan metode *negative*:

Kode 2.4: Contoh file `.htaccess` pada halaman `index.php`

```
11 RewriteEngine On
12 RewriteCond %{REQUEST_FILENAME} !-f
13 RewriteCond %{REQUEST_FILENAME} !-d
14 RewriteRule ^(.*)$ index.php/$1 [L]
```

Aturan diatas menyebabkan *HTTP Request* selain yang berasal dari direktori atau file diperlakukan sebagai sebuah permintaan pada file `index.php`. Selain itu, pengguna juga dapat menambahkan akhirkan pada *URL* agar halaman pengguna dapat menampilkan halaman sesuai dengan tipe yang diinginkan. Berikut merupakan contoh *URL* sebelum dan sesudah ditambahkan akhiran berupa:

```
example.com/index.php/products/view/shoes
example.com/index.php/products/view/shoes.html
```

Pengguna juga dapat menyalakan fitur *query strings* dengan cara sebagai mengubah file `application/config.php` seperti:

Kode 2.5: File `application/config.php`

```
25
26 1 $config['enable_query_strings'] = FALSE;
27 2 $config['controller_trigger'] = 'c';
28 3 $config['function_trigger'] = 'm';
29
```

Pengguna dapat mengubah `enable_query_strings` menjadi `TRUE`. *Controller* dan *function* dapat diakses menggunakan kata yang sudah diatur.

### 2.1.3 Helpers

*Helpers* merupakan fungsi pada *CodeIgniter 3* yang mempermudah pengguna dalam membangun aplikasi web. Setipa file *helpers* terdiri dari banyak fungsi yang membantu sesuai kategori dan tidak ditulis dalam format *Object Oriented*. File *helpers* terdapat pada direktori `system/helpers` atau `application/helpers`. Pengguna dapat memakai fitur *helpers* dengan cara memuatnya seperti berikut:

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

Pemanggilan *helper* tidak menggunakan ekstensi `.php` melainkan hanya menggunakan nama dari *helper* tersebut. Pengguna dapat memanggil satu atau banyak *helper* pada metode *controller* ataupun *view* sesudah dimuat.

### 2.1.4 Libraries

*CodeIgniter 3* menyediakan *library* yang dapat dipakai pengguna untuk mempermudah pembentukan aplikasi web. *Library* merupakan kelas yang tersedia pada direktori *application/libraries* dan dapat ditambahkan, diperluas, dan digantikan.

Kode 2.6: Contoh kelas *library* pada *CodeIgniter 3*

```

5
61 <?php
72 defined('BASEPATH') OR exit('No direct script access allowed');
83
94 class Someclass {
105
116     public function some_method()
127     {
138     }
149 }
```

Kode 2.6 merupakan contoh *file library* pada *CodeIgniter 3*. Setiap pembentukan *file library* diperlukan huruf kapital dan harus sama dengan nama kelasnya. Berikut merupakan contoh pemanggilan *file library* pada *file controller*:

```

19         $params = array('type' => 'large', 'color' => 'red');
20         $this->load->library('someclass', $params);
```

Pemanggilan kelas ini dapat dilakukan melalui *controller* manapun dan dapat diberikan parameter sesuai dengan metode yang dibentuk pada *library*. *CodeIgniter 3* menyediakan berbagai *library* yang dapat digunakan oleh pengguna seperti berikut:

#### JavaScript

Penggunaan kelas *javascript* dapat dipanggil pada konstruktor *controller* dengan cara berikut:

```

26         $this->load->library('javascript');
```

Pengguna selanjutnya harus melakukan inisiasi *library* pada halaman *view tag <head>* seperti berikut:

```

29         <?php echo $library_src;?>
30         <?php echo $script_head;?>
```

Sintaks `$library_src` merupakan lokasi *library* yang akan dimuat sedangkan `$script_head` merupakan lokasi untuk fungsi yang akan dijalankan. Selanjutnya *javascript* dapat diinisiasikan pada *controller* menggunakan sintaks berikut:

```

34         $this->javascript
```

Selain menggunakan *javascript*, pengguna dapat memakai *jQuery* dengan menambahkan *jQuery* pada akhir inisiasi kelas *javascript* seperti berikut:

```

37         $this->load->library('javascript/jquery');
```

Pengguna dapat memakai berbagai fungsi *library jquery* menggunakan sintaks berikut:

```

39         $this->jquery
```

## Kelas *Email*

*CodeIgniter 3* menyediakan kelas *email* dengan fitur sebagai berikut:

- Beberapa Protokol: *Mail*, *Sendmail*, dan *SMTP*
- Enkripsi *TLS* dan *SSL* untuk *SMTP*
- Beberapa Penerima
- *CC* dan *BCCs*
- *HTML* atau *email* teks biasa
- Lampiran
- Pembungkus kata
- Prioritas
- Mode *BCC Batch*, memisahkan daftar *email* besar menjadi beberapa *BCC* kecil.
- Alat *Debugging email*

Penggunaan *library email* dapat dikonfigurasi pada *file config*. Pengguna dapat mengirim *email* dengan mudah menggunakan fungsi-fungsi yang telah disediakan *library email*. Kode 2.7 merupakan contoh pengiriman email melalui *controller*.

Kode 2.7: Contoh pengiriman email melalui *controller*

```

16 $this->load->library('email');
17
18
19 $this->email->from('your@example.com', 'Your Name');
20 $this->email->to('someone@example.com');
21 $this->email->cc('another@example.com');
22 $this->email->bcc('them@their-example.com');
23
24 $this->email->subject('Email Test');
25 $this->email->message('Testing the email class.');
```

Kode 2.7 merupakan contoh penggunaan *library email* untuk mengirim *email* dari *your@example.com* menuju *someone@example.com*. Konfigurasi ini juga mengirim dua buah salinan menuju *another@example.com* dan *them@their-example.com* dengan subjek berupa *Email Test* dengan pesan *Testing the email class..* Selain itu, pengguna juga dapat melakukan konfigurasi preferensi *email* melalui dua puluh satu preferensi. Pengguna dapat melakukan konfigurasi secara otomatis melalui *file config* atau melakukan konfigurasi secara manual. Kode 2.8 merupakan contoh konfigurasi preferensi secara manual.

Kode 2.8: Contoh konfigurasi preferensi *library email* secara manual

```

36
37 $config['protocol'] = 'sendmail';
38 $config['mailpath'] = '/usr/sbin/sendmail';
39 $config['charset'] = 'iso-8859-1';
40 $config['wordwrap'] = TRUE;
41
42 $this->email->initialize($config);
```

Kode 2.8 merupakan contoh konfigurasi pengiriman *email* dengan protokol *sendmail* dan tujuan *usr/sbin/sendmail*.

## Kelas *File Uploading*

Pengunggahan *file* terdapat empat buah proses sebagai berikut:

1. Dibentuk sebuah form untuk pengguna memilih dan mengunggah *file*.



2. Setelah *file* diunggah, *file* akan dipindahkan menuju direktori yang dipilih.
  3. Pada pengiriman dan pemindahan *file* dilakukan validasi sesuai dengan ketentuan yang ada.
  4. Setelah *file* diterima akan dikeluarkan pesan berhasil.
- Perangkat lunak akan memindahkan *file* yang sudah diunggah pada *form* menuju *controller* untuk dilakukan validasi dan penyimpanan.

Kode 2.9: Contoh *controller* untuk melakukan validasi dan penyimpanan

```

6
7 1 <?php
8 2
9 3 class Upload extends CI_Controller {
10 4
11 5     public function __construct()
12 6     {
13 7         parent::__construct();
14 8         $this->load->helper(array('form', 'url'));
15 9     }
16 0
17 1     public function index()
18 2     {
19 3         $this->load->view('upload_form', array('error' => ' '));
20 4     }
21 5
22 6     public function do_upload()
23 7     {
24 8         $config['upload_path']       = './uploads/';
25 9         $config['allowed_types']     = 'gif|jpg|png';
26 0         $config['max_size']         = 100;
27 1         $config['max_width']        = 1024;
28 2         $config['max_height']       = 768;
29 3
30 4         $this->load->library('upload', $config);
31 5
32 6         if ( ! $this->upload->do_upload('userfile'))
33 7         {
34 8             $error = array('error' => $this->upload->display_errors());
35 9
36 0             $this->load->view('upload_form', $error);
37 1         }
38 2         else
39 3         {
40 4             $data = array('upload_data' => $this->upload->data());
41 5
42 6             $this->load->view('upload_success', $data);
43 7         }
44 8     }
45 9 }
46 0 ?>

```

Kode 2.9 merupakan contoh kode dengan dua buah metode yakni:

1. `index()` yang digunakan untuk mengembalikan *view* bernama `upload_form`
2. `do_upload` yang digunakan untuk melakukan validasi berupa tipe, ukuran, lebar, dan panjang maksimal sebuah file. Metode ini juga mengembalikan *error* dan menyimpan *file* sesuai dengan direktori yang sudah diatur.

Direktori penyimpanan dapat diubah sesuai dengan kebutuhan namun perlu pengubahan izin direktori menjadi 777 agar dapat di baca, di tulis, dan di eksekusi oleh seluruh pengguna.

### Kelas *Zip Encoding*

*Zip* merupakan format sebuah *file* yang berguna untuk melakukan kompress terhadap *file* untuk mengurangi ukuran dan menjadikannya sebuah *file*. *CodeIgniter 3* menyediakan *library Zip Encoding* yang dapat digunakan untuk membangun arsip *Zip* yang dapat diunduh menuju *desktop* atau disimpan pada direktori. *Library* ini dapat diinsiasi dengan kode sebagai berikut:

```
$this->load->library('zip');
```

1 Setelah diinisiasi, pengguna dapat memanggil *library* tersebut menggunakan kode sebagai  
 2 berikut:

```
3 $this->zip
```

4 Kode 2.10 merupakan contoh penggunaan *library Zip Encoding* untuk menyimpan dan menunduh  
 5 data.

Kode 2.10: Contoh penggunaan *library Zip Encoding*

```
6
71 $name = 'mydata1.txt';
82 $data = 'A Data String!';
93
104 $this->zip->add_data($name, $data);
115
126 // Write the zip file to a folder on your server. Name it "my_backup.zip"
137 $this->zip->archive('/path/to/directory/my_backup.zip');
148
159 // Download the file to your desktop. Name it "my_backup.zip"
160 $this->zip->download('my_backup.zip');
```

18 Kode 2.10 merupakan contoh untuk melakukan penyimpanan *Zip file* pada direktori dan dapat  
 19 mengunduh *file* menuju *desktop* pengguna. Selain menggunakan *library* yang sudah disediakan,  
 20 pengguna dapat membangun dan memperluas *libraries* sendiri sesuai dengan kebutuhan. Kode  
 21 merupakan contoh *library* yang dibentuk.

Kode 2.11: Contoh *library* yang dibentuk

```
22
231 class Example_library {
242
253     protected $CI;
264
275     // We'll use a constructor, as you can't directly call a function
286     // from a property definition.
297     public function __construct()
308     {
319         // Assign the CodeIgniter super-object
320         $this->CI =& get_instance();
331     }
342
353     public function foo()
364     {
375         $this->CI->load->helper('url');
386         redirect();
397     }
408 }
41
```

42 *Library* diatas merupakan contoh *library* yang dibentuk oleh pengguna digunakan untuk me-  
 43 manggil *helper* bernama *url*. Pengguna dapat memanggil kelas tersebut seperti memanggil kelas  
 44 *library* lainnya. Selain itu, pengguna juga dapat mengganti *library* yang sudah ada dengan *library*  
 45 yang dibentuk pengguna dengan mengubah nama kelas sama persis dengan nama *library* yang ingin  
 46 digantikan.

## 47 2.1.5 Database

48 *CodeIgniter 3* memiliki konfigurasi *database* yang menyimpan data-data terkait aturan *database*.

Kode 2.12: Contoh konfigurasi *database*

```
49
501 $db['default'] = array(
512     'dsn' => '',
523     'hostname' => 'localhost',
534     'username' => 'root',
545     'password' => '',
556     'database' => 'database_name',
```

```

17     'dbdriver' => 'mysqli',
18     'dbprefix' => '',
19     'pconnect' => TRUE,
20     'db_debug' => TRUE,
21     'cache_on' => FALSE,
22     'cachedir' => '',
23     'char_set' => 'utf8',
24     'dbcollat' => 'utf8_general_ci',
25     'swap_pre' => '',
26     'encrypt' => FALSE,
27     'compress' => FALSE,
28     'stricton' => FALSE,
29     'failover' => array()
30 );

```

Kode 2.12 merupakan contoh konfigurasi pada file *database* untuk *database* bernama *database\_name* dengan *username* *root* tanpa sebuah *password*. *CodeIgniter 3* menyediakan fitur *query* untuk menyimpan, memasukan, memperbarui, dan menghapus data pada *database* sesuai dengan konfigurasi *database* yang sudah diatur. Kode 2.13 merupakan contoh *query* untuk melakukan *select* dan *join* pada *CodeIgniter 3*:

Kode 2.13: Contoh penggunaan *query*

```

21 $this->db->select('*');
22 $this->db->from('blogs');
23 $this->db->join('comments', 'comments.id = blogs.id');
24 $query = $this->db->get();

```

Kode 2.13 merupakan contoh kode untuk melakukan *query* pada tabel *blogs* yang melakukan *join* dengan tabel *comments*. Pengguna dapat mengambil hasil dari *query* menjadi *object* atau *array*. Selain itu, *database* pada *CodeIgniter 3* juga dapat digunakan untuk membangun, menghapus, dan mengubah *database* ataupun menambahkan kolom pada *table*. Penggunaan *database* untuk membentuk, menghapus, atau mengubah *database* harus dilakukan inisiasi sebagai berikut:

```

32 $this->load->dbforge()

```

Setelah dilakukan inisiasi pengguna dapat membangun *database* menggunakan kelas *Forge*. Kode 2.14 merupakan contoh untuk membangun *database*.

Kode 2.14: Contoh membangun *database* menggunakan *CodeIgniter3*

```

35 $this->dbforge->create_database('db_name')

```

Selain itu, pengguna juga dapat menambahkan kolom dengan konfigurasinya. Kode 2.15 merupakan contoh penambahan kolom sesuai dengan kebutuhan pengguna.

Kode 2.15: Contoh menambahkan kolom dengan konfigurasinya menggunakan *CodeIgniter3*

```

40 $fields = array(
41     'blog_id' => array(
42         'type' => 'INT',
43         'constraint' => 5,
44         'unsigned' => TRUE,
45         'auto_increment' => TRUE
46     ),
47     'blog_title' => array(
48         'type' => 'VARCHAR',
49         'constraint' => '100',
50         'unique' => TRUE,
51     ),
52     'blog_author' => array(
53         'type' => 'VARCHAR',
54         'constraint' => '100',
55         'default' => 'King of Town',
56     ),
57     'blog_description' => array(

```

```

119         'type' => 'TEXT',
120         'null' => TRUE,
121     ),
122 );
123 $this->dbforge->add_field($fields)
124 $this->dbforge->create_table('table_name');

```

Kode 2.15 merupakan contoh penggunaan *database* untuk menambahkan kolom sesuai dengan tipenya pada tabel `table_name`.

### 2.1.6 URI Routing

*URL string* biasanya menggunakan nama atau metode *controller* seperti pada berikut:

```
example.com/class/function/id/
```

Namun, pengguna dapat melakukan pemetaan ulang terhadap *url* yang dibentuk agar dapat memanggil beberapa metode.

Kode 2.16: Contoh *url* yang sudah dimetakan

```

15 example.com/product/1/
16 example.com/product/2/
17 example.com/product/3/
18 example.com/product/4/

```

Kode 2.16 merupakan contoh *url* yang sudah dimetakan ulang. Pengguna dapat menambahkan kode pemetaan pada *file application/config/routes.php* yang terdapat *array* bernama `$route`. Berikut merupakan beberapa cara melakukan pemetaan terhadap *url*:

#### WildCards

*Route wildcard* biasanya berisikan kode seperti berikut:

```
$route['product/:num'] = 'catalog/product_lookup';
```

*Route* diatas dibagi menjadi dua buah yakni:

#### 1. Bagian segmen *URL*

Bagian pertama merupakan segmen pertama *url* yang akan tampil pada *url*. Bagian kedua merupakan segmen kedua dapat berisikan angka atau karakter.

#### 2. Bagian kelas dan metode

Bagian kedua berisikan kelas dan metode dari *controller* yang akan digunakan pada *url*.

#### Ekspresi Reguler

Pengguna dapat memakai ekspresi reguler untuk melakukan pemetaan ulang *route*. Berikut merupakan contoh ekspresi reguler yang biasa digunakan:

```
$route['products/([a-z]+)/(\d+)'] = '$1/id_$2';
```

Ekspresi ini menghasilkan *URI products/shirts/123* yang memanggil kelas *controller* dan metode *id\_123*. Pengguna juga dapat mengambil segmen banyak seperti berikut:

```
$route['login/(.+)'] = 'auth/login/$1';
```

### 2.1.7 Auto-loading

*CodeIgniter 3* menyediakan sebuah fungsi untuk memuat berbagai kelas seperti *libraries*, *helpers*, dan *models*. Kelas dapat dimasukkan pada `application/config/autoload.php` sesuai dengan petunjuk yang ada. *File autoload* akan di inisiasikan setiap aplikasi dijalankan sehingga pengguna tidak perlu memuat kelas tersebut berulang kali.

## 2.2 SharIF Judge[2]

*SharIF Judge* merupakan sebuah *Online Judge* percabangan dari *Sharif Judge* yang dibentuk oleh Mohammed Javad Naderi. *Sharif Judge* dibentuk menggunakan *CodeIgniter 3* dan dimodifikasi sesuai dengan kebutuhan di Informatika Universitas Katolik Parahyangan menjadi nama *SharIF Judge*. *SharIF Judge* dapat menilai kode berbahasa *C*, *C++*, *Java*, dan *Python* dengan mengunggah file ataupun mengetiknya secara langsung.

### 2.2.1 Struktur Aplikasi

```

13 .
14 |
15 |-- application
16 | |-- cache
17 | |-- config
18 | |-- controllers
19 | |-- core
20 | |-- helpers
21 | |-- hooks
22 | |-- language
23 | |-- libraries
24 | |-- logs
25 | |-- models
26 | |-- third_party
27 | |-- vendor
28 | |-- views
29 |-- assets
30 | |-- images
31 | |-- js
32 | |-- styles
33 |-- restricted
34 | |-- tester
35 |-- system
36 |-- assignments

```

### 2.2.2 Instalasi

Berikut merupakan persyaratan dan langkah-langkah melakukan *instalasi SharIF Judge*:

#### Persyaratan

*SharIF Judge* dapat dijalankan pada sistem operasi *Linux* dengan syarat sebagai berikut:

- Diperlukan *webserver* dengan versi PHP 5.3 atau lebih baru.
- Pengguna dapat menjalankan PHP pada *command line*. Pada *Ubuntu* diperlukan instalasi paket *php5-cli*.
- *MySQL database* dengan ekstensi *MySQLi* untuk PHP atau *PostgreSQL database*.
- PHP harus memiliki akses untuk menjalankan perintah melalui fungsi *shell\_exec*.

Kode 2.17: Kode untuk melakukan pengetesan fungsi

```

1 echo shell_exec("php -v");

```

- *Tools* untuk melakukan kompilasi dan menjalankan kode yang dikumpulkan (*gcc*, *g++*, *javac*, *java*, *python2*, *python3*).
- *Perl* disarankan untuk diinstalasi untuk alasan ketepatan waktu, batas memori, dan memaksimalkan batas ukuran pada hasil kode yang dikirim.

## Instalasi

1. Mengunduh versi terakhir dari *SharIF Judge* dan melakukan *unpack* pada direktori *public html*.
2. Memindahkan *folder system* dan *application* diluar direktori *public* dan mengubah *path* pada *index.php*(Opsional).

Kode 2.18: Contoh *path* pada halaman *index.php*

```
1 $system_path = '/home/mohammad/secret/system';
2 application_folder = '/home/mohammad/secret/application';
```

3. Membangun *database MySQL* atau *PostgreSql* untuk *SharIF Judge*. Jangan melakukan instalasi paket koneksi *database* apapun untuk *C*, *C++*, *Java*, atau *Python*.
4. Mengatur koneksi *database* pada file *application/config/database.example.php* dan menyimpannya dengan nama *database.php*. Pengguna dapat menggunakan awalan untuk nama tabel.

Kode 2.19: Contoh pengaturan koneksi untuk *database*

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

5. Mengatur *RADIUS server* dan *mail server* pada file *application/config/secrets.example.php* dan menyimpannya dengan nama *secrets.php*.
6. Mengatur *application/cache/Twig* agar dapat ditulis oleh PHP.
7. Membuka halaman utama *SharIF Judge* pada *web browser* dan mengikuti proses instalasi.
8. Melakukan *Log in* dengan akun admin.
9. Memindahkan direktori *tester* dan *assignments* diluar direktori publik dan mengatur kedua direktori agar dapat ditulis oleh PHP. Selanjutnya Menyimpan *path* kedua direktori pada halaman *Settings*. Direktori *assignments* digunakan untuk menyimpan *file-file* yang diunggah agar tidak dapat diakses publik.

### 2.2.3 Clean URLs

Secara asali, *index.php* merupakan bagian dari seluruh *urls* pada *SharIF judge*. Berikut merupakan contoh dari *urls* *SharIF Judge*.

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

Pengguna dapat menghapus *index.php* pada *url* dan mendapatkan *url* yang baik apabila sistem pengguna mendukung *URL rewriting*.

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

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

### Cara Mengaktifkan *Clean URLs*

- Mengganti nama *file* `.htaccess2` pada direktori utama menjadi `.htaccess`.
- Mengganti `$config['index_page'] = 'index.php';` menjadi `$config['index_page'] = '';` pada *file* `application\config\config.php`.

#### 2.2.4 Users

Pada perangkat lunak SharIF Judge, pengguna dibagi menjadi 4 buah. Keempat pengguna tersebut adalah *Admins*, *Head Instructors*, *Instructors*, dan *Students*. Tabel 2.1 merupakan pembagian tingkat setiap pengguna.

Tabel 2.1: Tabel tingkat pengguna

<i>User Role</i>	<i>User Level</i>
<i>Admin</i>	3
<i>Head Instructor</i>	2
<i>Instructor</i>	1
<i>Student</i>	0

Setiap pengguna memiliki akses untuk aksi yang berbeda berdasarkan tingkatnya. Tabel 2.2 merupakan aksi yang dapat dilakukan oleh setiap pengguna.

Tabel 2.2: Tabel izin aksi setiap pengguna

Aksi	<i>Admin</i>	<i>Head Instructor</i>	<i>Instructor</i>	<i>Student</i>
Mengubah <i>Settings</i>	✓	×	×	×
Menambah/Menghapus Pengguna	✓	×	×	×
Mengubah Peran Pengguna	✓	×	×	×
Menambah/Menghapus/Mengubah <i>Assignment</i>	✓	✓	×	×
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 <i>Assignment</i>	✓	✓	✓	✓
<i>Submit</i>	✓	✓	✓	✓

#### Menambahkan Pengguna

Admin dapat menambahkan pengguna melalui bagian *Add User* pada halaman *Users*. Admin harus mengisi setiap informasi dimana baris yang diawali # merupakan komen dan setiap baris lainnya mewakili pengguna dengan sintaks berikut:

Kode 2.20: Contoh sintaks untuk menambahkan pengguna

```

11 USERNAME,EMAIL,DISPLAY-NAME,PASSWORD,ROLE
12
13 * Usernames may contain lowercase letters or numbers and must be between 3 and 20 characters in length.
14 * Passwords must be between 6 and 30 characters in length.
15 * You can use RANDOM[n] for password to generate random n-digit password.
16 * ROLE must be one of these: 'admin', 'head_instructor', 'instructor', 'student'

```

Dengan contoh sebagai berikut:

Kode 2.21: Contoh kode untuk menambahkan pengguna

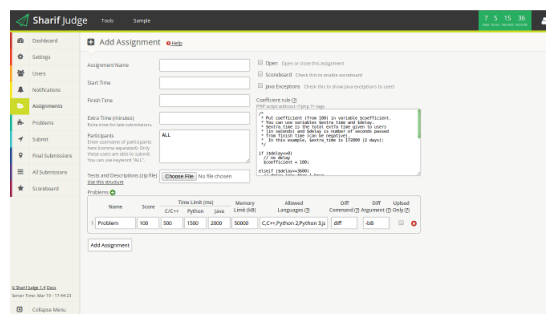
```

9
10 # This is a comment!
11 # This is another comment!
12 instructor,instructor@sharifjudge.ir,Instructor One,123456,head_instructor
13 instructor2,instructor2@sharifjudge.ir,Instructor Two,random[7],instructor
14 student1,st1@sharifjudge.ir,Student One,random[6],student
15 student2,st2@sharifjudge.ir,Student Two,random[6],student
16 student3,st3@sharifjudge.ir,Student Three,random[6],student
17 student4,st4@sharifjudge.ir,Student Four,random[6],student
18 student5,st5@sharifjudge.ir,Student Five,random[6],student
19 student6,st6@sharifjudge.ir,Student Six,random[6],student
20 student7,st7@sharifjudge.ir,Student Seven,random[6],student
21

```

## 2.2.5 Menambah Assignment

Pengguna dapat menambahkan *assignment* baru melalui bagian *Add* pada halaman *Assignments* (dapat dilihat pada Gambar 2.2).



Gambar 2.2: Tampilan halaman *SharIF Judge* untuk menambahkan *assignment*

Berikut merupakan beberapa pengaturan pada halaman *Add Assignments*:

- **Assignment Name**

*Assignment* akan ditampilkan sesuai dengan masukan pada daftar *assignment*.

- **Start Time**

Pengguna tidak dapat mengumpulkan *assignment* sebelum waktu dimulai ("Start Time"). Format pengaturan waktu untuk waktu mulai adalah MM/DD/YYYY HH:MM:SS dengan contoh 08/31/2013 12:00:00.

- **Finish Time, Extra Time**

Pengguna tidak dapat mengumpulkan *assignment* setelah *Finish Time + Extra Time*. Pengumpulan *Assignment* pada *Extra Time* akan dikalikan sesuai dengan koefisien. Pengguna harus menulis skrip PHP untuk menghitung koefisien pada *field Coefficient Rule*. Format



pengaturan waktu untuk waktu selesai sama seperti waktu mulai yakni MM/DD/YYYY HH:MM:SS dan format waktu tambahan menggunakan menit dengan contoh 120 (2 jam) atau 48\*60 (2 hari).

#### • *Participants*

Pengguna dapat memasukkan *username* setiap partisipan atau menggunakan kata kunci *ALL* untuk membiarkan seluruh pengguna melakukan pengumpulan. Contoh: *admin, instructor1, instructor2, student1, student2, student3, student4*.

#### • *Tests*

Pengguna dapat mengunggah *test case* dalam bentuk *zip file* sesuai dengan struktur pada [2.2.7](#).

#### • *Open*

Pengguna dapat membuka dan menutup *assignment* untuk pengguna *student* melalui pilihan ini. Pengguna selain *student* tetap dapat mengumpulkan *assignment* apabila sudah ditutup.

#### • *Score Board*

Pengguna dapat menghidupkan dan mematikan *score board* melalui pilihan ini.

#### • *Java Exceptions*

Pengguna dapat menghidupkan dan mematikan fungsi untuk menunjukan *java exceptions* kepada pengguna *students* dan tidak akan memengaruhi kode yang sudah di *judge* sebelumnya. Berikut merupakan tampilan apabila fitur *java exceptions* dinyalakan:

Kode 2.22: Contoh tampilan fitur *Java Exceptions*

```

1 Test 1
2 ACCEPT
3 Test 2
4 Runtime Error (java.lang.ArrayIndexOutOfBoundsException)
5 Test 3
6 Runtime Error (java.lang.ArrayIndexOutOfBoundsException)
7 Test 4
8 ACCEPT
9 Test 5
10 ACCEPT
11 Test 6
12 ACCEPT
13 Test 7
14 ACCEPT
15 Test 8
16 Runtime Error (java.lang.ArrayIndexOutOfBoundsException)
17 Test 9
18 Runtime Error (java.lang.StackOverflowError)
19 Test 10
20 Runtime Error (java.lang.ArrayIndexOutOfBoundsException)

```

#### • *Archived Assignment*

Pengguna dapat menghidupkan fitur ini dan *assignment* akan dibentuk dengan waktu selesai 2038-01-18 00:00:00 (UTC + 7) dengan kata lain pengguna memiliki waktu tidak terhingga

untuk mengumpulkan *assignment*.

- ***Coefficient Rule***

Pengguna dapat menuliskan skrip PHP pada bagian ini untuk menghitung koefisien dikalikan dengan skor. Pengguna harus memasukan koefisien (dari 100) dalam variabel `$coefficient`. Pengguna dapat menggunakan variabel `$extra_time` dan `$delay`. `$extra_time` merupakan total dari waktu tambahan yang diberikan kepada pengguna dalam detik dan `$delay` merupakan waktu dalam detik yang melewati waktu selesai(dapat berupa negatif). Skrip PHP pada bagian ini tidak boleh mengandung tag `<?php`, `<?`, dan `?>`. Berikut merupakan contoh skrip dimana `$extra_time` adalah 172800(2 hari):

Kode 2.23: Contoh skrip PHP

```

1 if ($delay<=0)
2     // no delay
3     $coefficient = 100;
4
5 elseif ($delay<=3600)
6     // delay less than 1 hour
7     $coefficient = ceil(100-((30*$delay)/3600));
8
9 elseif ($delay<=86400)
10    // delay more than 1 hour and less than 1 day
11    $coefficient = 70;
12
13 elseif (($delay-86400)<=3600)
14    // delay less than 1 hour in second day
15    $coefficient = ceil(70-((20*($delay-86400))/3600));
16
17 elseif (($delay-86400)<=86400)
18    // delay more than 1 hour in second day
19    $coefficient = 50;
20
21 elseif ($delay > $extra_time)
22    // too late
23    $coefficient = 0;

```

- ***Name***

Merupakan nama dari masalah pada *assignments*.

- ***Score***

Merupakan skor dari masalah pada *assignments*.

- ***Time Limit***

Pengguna dapat menentukan batas waktu untuk menjalankan kode dalam satuan milidetik. Bahasa *Python* dan *Java* biasanya memiliki waktu lebih lambat dari *C/C++* sehingga membutuhkan waktu lebih lama.

- ***Memory Limit***

Pengguna dapat menentukan batas memori dalam *kilobytes* namun, pengguna pembatasan memori tidak terlalu akurat.

### • *Allowed Languages*

Pengguna dapat menentukan bahasa setiap masalah pada *assignment* (dipisahkan oleh koma). Terdapat beberapa bahasa yang tersedia yaitu *C*, *C++*, *Java*, *Python 2*, *Python 3*, *Zip*, *PDF*, dan *TXT*. Pengguna dapat memakai *Zip*, *PDF*, dan *TXT* apabila opsi *Upload Only* dinyalakan. Contoh : *C*, *C++*, *Zip* atau *Python 2*, *Python 3*.

### • *Diff Command*

*Diff Command* digunakan untuk membandingkan keluaran dengan keluaran yang benar. Secara asali, *SharIF Judge* menggunakan *diff* namun, pengguna dapat menggantinya pada bagian ini dan bagian ini tidak boleh mengandung spasi.

### • *Diff Arguments*

Pengguna dapat mengatur argumen untuk *diff arguments* pada bagian ini. Pengguna dapat melihat *man diff* untuk daftar lengkap argumen *diff*. *SharIF Judge* terdapat dua buah opsi baru yakni *ignore* dan *identical*.

- *ignore* : *SharIF Judge* mengabaikan semua baris baru dan spasi.
- *identical* : *SharIF Judge* tidak mengabaikan apapun namun, keluaran dari *file* yang dikumpulkan harus identik dengan *test case* agar dapat diterima.

### • *Upload Only*

Pengguna dapat menghidupkan *Upload only* namun, *SharIF Judge* tidak akan menilai masalah tersebut. Pengguna dapat memakai *ZIP*, *PDF*, dan *TXT* pada *allowed languages* apabila pengguna menghidupkan bagian ini.

## 2.2.6 *Sample Assignment*

Berikut merupakan contoh dari *assignment* untuk melakukan pengujian *SharIF Judge*. Penambahan *Assignment* dapat dilakukan dengan memencet tombol *Add* pada halaman *Assignment*.

### *Problems*

1. *Problem 1 (Sum)*: Program pengguna dapat membaca *integer n*, membaca *n integers* dan mengeluarkan hasil dari *integer* tersebut.

<i>Sample Input</i>	<i>Sample Output</i>
5 53 78 0 4 9	145

2. *Problem 2 (Max)*: Program pengguna dapat membaca *integer n*, membaca *n integer*, dan mengeluarkan total dari dua buah *integer* terbesar diantara *n integer*.

<i>Sample Input</i>	<i>Sample Output</i>
7 162 173 159 164 181 158 175	356

3. *Problem 3 (Upload)*: Pengguna dapat mengunggah *file c* dan *zip* dan *problem* ini tidak akan dinilai karena hanya berupa *Upload Only*.

### Tests

Pengguna dapat menemukan *file zip* pada direktori *Assignments*. Berikut merupakan susunan pohon dari ketiga *problems* diatas:

```

.
|-- 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

```

*Problem 1* menggunakan metode "*Tester*" untuk mengecek hasil keluaran, sehingga memiliki *file tester.cpp* (*Tester Script*). *Problem 2* menggunakan metode "*Output Comparison*" untuk mengecek hasil keluaran, sehingga memiliki dua buah direktori *in* dan *out* yang berisi *test case*. *Problem 3* merupakan *problem "Upload-Only"*.

### Sample Solutions

Berikut merupakan *sample solutions* untuk *problem 1*:

Contoh solusi untuk bahasa *C*

Kode 2.24: Contoh skrip PHP

```

#include<stdio.h>
int main(){
    int n;
    scanf("%d",&n);

```

```

15  int i;
26  int sum =0 ;
37  int k;
48  for(i=0 ; i<n ; i++){
59      scanf("%d",&k);
60      sum+=k;
71  }
82  printf("%d\n",sum);
93  return 0;
104 }

```

Contoh solusi untuk bahasa C++

```

13
14 1 #include<stdio.h>
15 2 int main(){
16 3     int n;
17 4     scanf("%d",&n);
18 5     int i;
19 6     int sum =0 ;
20 7     int k;
21 8     for(i=0 ; i<n ; i++){
22 9         scanf("%d",&k);
23 0         sum+=k;
24 1     }
25 2     printf("%d\n",sum);
26 3     return 0;
27 4 }

```

Contoh solusi untuk bahasa C

```

30
31 1 import java.util.Scanner;
32 2 class sum
33 3 {
34 4     public static void main(String[] args)
35 5     {
36 6         Scanner sc = new Scanner(System.in);
37 7         int n = sc.nextInt();
38 8         int sum =0;
39 9         for (int i=0 ; i<n ; i++)
40 0         {
41 1             int a = sc.nextInt();
42 2             sum += a;
43 3         }
44 4         System.out.println(sum);
45 5     }
46 6 }

```

Berikut merupakan contoh solusi untuk *problem 2*:

Contoh solusi untuk bahasa C++

```

50
51 1 #include<stdio.h>
52 2 int main(){
53 3     int n , m1=0, m2=0;
54 4     scanf("%d",&n);
55 5     for(;n--;){
56 6         int k;
57 7         scanf("%d",&k);
58 8         if(k>=m1){
59 9             m2=m1;
60 0             m1=k;
61 1         }
62 2         else if(k>m2)
63 3             m2=k;
64 4     }
65 5     printf("%d",m1+m2);
66 6     return 0;
67 7 }

```

contoh solusi untuk bahasa C++

```

70
71 1 #include<iostream>
72 2 using namespace std;
73 3 int main(){
74 4     int n , m1=0, m2=0;
75 5     cin >> n;
76 6     for(;n--;){
77 7         int k;
78 8         cin >> k;

```

```

19     if(k>=m1){
20         m2=m1;
31         m1=k;
42     }
53     else if(k>m2)
64         m2=k;
75 }
86 cout << (m1+m2) << endl ;
97 return 0;
108 }
11

```

### 2.2.7 Test Structure

Penambahan *assignment* harus disertakan dengan *file zip* berisikan *test cases*. *File zip* ini sebaiknya berisikan *folder* untuk setiap masalah dengan nama *p1,p1* dan *p3* selain masalah *Upload-Only*.

### Metode Pemeriksaan

Terdapat dua buah metode untuk melakukan pemeriksaan yakni metode *Input Output* dan metode *Tester*.

#### 1. Metode perbandingan *Input Output*

Pada metode ini, pengguna harus memberi masukan dan keluaran pada *folder problem*. Perangkat lunak akan memberikan *file test input* ke kode pengguna dan melakukan perbandingan dengan hasil keluaran kode pengguna. *File input* harus berada didalam *folder in* dengan nama *input1.txt, input2.txt*, dst. *File output* harus berada di dalam *folder out* dengan nama *output1.txt, output2.txt*.

#### 2. Metode perbandingan *Tester*

Pada metode ini, pengguna harus menyediakan *file input test*, sebuah *file C++*, dan *file output test*(opsional). Perangkat lunak akan memberikan *file input test* ke kode pengguna dan mengambil keluaran pengguna. Selanjutnya, *tester.cpp* akan mengambil masukan pengguna, keluaran tes dan keluaran program pengguna. Jika keluaran pengguna benar maka perangkat lunak akan mengembalikan 1 sedangkan apabila salah maka perangkat lunak akan mengembalikan 0. Berikut adalah templat yang dapat digunakan pengguna untuk menuliskan *file tester.cpp*:

Kode 2.25: Templat kode *tester.cpp*

```

32 1 /*
33 2  * tester.cpp
34 3  */
35 4
36 5 #include <iostream>
37 6 #include <fstream>
38 7 #include <string>
39 8 using namespace std;
40 9 int main(int argc, char const *argv[])
41 10 {
42 11
43 12     ifstream test_in(argv[1]); /* This stream reads from test's input file */
44 13     ifstream test_out(argv[2]); /* This stream reads from test's output file */
45 14     ifstream user_out(argv[3]); /* This stream reads from user's output file */
46 15
47 16     /* Your code here */
48 17     /* If user's output is correct, return 0, otherwise return 1 */
49 18
50 19     ...
51 20
52 21 }
53

```

### Sample File

Pengguna dapat menemukan *file sample test* pada direktori *assignments*. Berikut merupakan contoh dari pohon *file* tersebut:

```

1 .
2 |-- p1
3 |   |-- in
4 |       |-- input1.txt
5 |       |-- input2.txt
6 |       |-- input3.txt
7 |       |-- input4.txt
8 |       |-- input5.txt
9 |       |-- input6.txt
10 |      |-- input7.txt
11 |      |-- input8.txt
12 |      |-- input9.txt
13 |      |-- input10.txt
14 |     |-- out
15 |         --- output1.txt
16 |    |-- tester.cpp
17 |-- p2
18 |   |-- in
19 |       |-- input1.txt
20 |       |-- input2.txt
21 |       |-- input3.txt
22 |       |-- input4.txt
23 |       |-- input5.txt
24 |       |-- input6.txt
25 |       |-- input7.txt
26 |       |-- input8.txt
27 |       |-- input9.txt
28 |       |-- input10.txt
29 |     |-- out
30 |         |-- output1.txt
31 |         |-- output2.txt
32 |         |-- output3.txt
33 |         |-- output4.txt
34 |         |-- output5.txt
35 |         |-- output6.txt
36 |         |-- output7.txt
37 |         |-- output8.txt
38 |         |-- output9.txt
39 |         --- output10.txt

```

*Problem 1* menggunakan metode perbandingan *Tester*, sehingga memiliki *file tester.cpp*. Berikut merupakan *file* untuk *problem 1*:

Kode 2.26: Kode metode perbandingan *tester* dengan bahasa *tester.cpp*

```

1  /*
2   * tester.cpp
3   */
4
5  #include <iostream>
6  #include <fstream>
7  #include <string>
8  using namespace std;
9  int main(int argc, char const *argv[])
10 {
11
12     ifstream test_in(argv[1]);    /* This stream reads from test's input file */
13     ifstream test_out(argv[2]);   /* This stream reads from test's output file */
14     ifstream user_out(argv[3]);   /* This stream reads from user's output file */
15
16     /* Your code here */
17     /* If user's output is correct, return 0, otherwise return 1 */
18     /* e.g.: Here the problem is: read n numbers and print their sum: */
19
20     int sum, user_output;
21     user_out >> user_output;
22
23     if ( test_out.good() ) // if test's output file exists
24     {
25         test_out >> sum;
26     }
27     else
28     {

```

```

1      29      int n, a;
2      30      sum=0;
3      31      test_in >> n;
4      32      for (int i=0 ; i<n ; i++){
5      33          test_in >> a;
6      34          sum += a;
7      35      }
8      36      }
9      37
10     38      if (sum == user_output)
11     39          return 0;
12     40      else
13     41          return 1;
14     42
15     43  }

```

*Problem 2* menggunakan metode perbandingan *Input Output* sehingga memiliki *folder in* dan *folder out* berisikan *test case*. Sedangkan *problem 3* merupakan *Upload Only*, sehingga tidak memiliki *folder* apapun.

## 2.2.8 Deteksi Kecurangan

*SharIF Judge* menggunakan *Moss* untuk mendeteksi kode yang mirip. *Moss* (*Measure of Software Similarity*) merupakan sistem otomatis untuk menentukan kesamaan atau kemiripan dalam program. Penggunaan utama *Moss* adalah untuk melakukan pemeriksaan plagiarisme pada kelas *programming*. Pengguna dapat mengirim hasil kode terakhir (*Final Submission*) ke *server Moss* dengan satu klik.

Penggunaan *Moss* memiliki beberapa hal yang harus diatur oleh pengguna yakni:

1. Pengguna harus mendapatkan *Moss User id* dan melakukan pengaturan pada *SharIF Judge*. Untuk mendapatkan *Moss User id*, pengguna dapat mendaftar pada halaman <http://theory.stanford.edu/~aiken/moss/>. Pengguna selanjutnya akan mendapatkan *email* berupa skrip *perl* berisikan *user id* pengguna. Berikut merupakan contoh dari potongan skrip *perl*:

Kode 2.27: Contoh potongan skrip *perl*

```

30      1
31      2
32      3
33      4
34      5
35      6
36      7
37      8
38      9
39      10
40      11
41      12
42      13
43      14
44      15
45      16
46      17
47      18
48      19
49      20
50      21
51      22
52      23
53      24
54      25
55      26
56      27
57      28
58      29
59      30

```

```

...
$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;
...

```

*User id* pada skrip *perl* diatas dapat digunakan pada *SharIF Judge* untuk mendeteksi kecurangan. Pengguna dapat menyimpan *user id* pada halaman *SharIF Judge Moss* dan *SharIF Judge* akan menggunakan *user id* tersebut.

2. *Server* pengguna harus memiliki instalasi *perl* untuk menggunakan *Moss*.
3. Pengguna dianjurkan untuk mendeteksi kode yang mirip setelah *assignment* selesai karena



*SharIF Judge* akan mengirimkan hasil akhir kepada *Moss* dan pengguna(*students*) dapat mengganti hasil akhir sebelum *assignment* selesai.

### 2.2.9 Keamanan

Berikut merupakan langkah untuk melakukan pengaturan keamanan *SharIF Judge*:

#### 1. Menggunakan *Sandbox*

Pengguna harus memastikan untuk menggunakan *sandbox* untuk bahasa *C/C++* dan menyalakan *Java Security Manager (Java Policy)* untuk bahasa *java*. Untuk menggunakan *sandbox* dapat dilihat pada sub bab 2.2.10.

#### 2. Menggunakan *Shield*

Pengguna harus memastikan untuk menggunakan *shield* untuk bahasa *C*, *C++*, dan *Python*. Penggunaan *shield* dapat dilihat pada subbab 2.2.11.

#### 3. Menjalankan sebagai *Non-Privileged User*

Pengguna diwajibkan menjalankan kode yang telah dikumpulkan sebagai *non-privileged user*. *Non-Privileged User* adalah *user* yang tidak memiliki akses jaringan, tidak dapat menulis *file* apapun, dan tidak dapat membangun banyak proses.

Diasumsikan bahwa PHP dijalankan sebagai pengguna *www-data* pada server. Membangun *user* baru *restricted-user* dan melakukan pengaturan *password*. Selanjutnya, jalankan *sudo visudo* dan tambahkan kode *www-data ALL=(restricted\_user) NOPASSWD: ALL* pada baris terakhir *file sudoers*.

- Pada *file tester\runcode.sh* ubah kode:

Kode 2.28: Kode *runcode.sh* awal

```
1 if $TIMEOUT_EXISTS; then
2     timeout -s9 $((TIMELIMITINT*2)) $CMD <$IN >out 2>err
3 else
4     $CMD <$IN >out 2>err
5 fi
```

menjadi:

Kode 2.29: Kode *runcode.sh* awal

```
1 if $TIMEOUT_EXISTS; then
2     sudo -u restricted_user timeout -s9 $((TIMELIMITINT*2)) $CMD <$IN >out 2>err
3 else
4     sudo -u restricted_user $CMD <$IN >out 2>err
5 fi
```

dan *uncomment* baris berikut:

Kode 2.30: Kode *runcode.sh* awal

```
1 sudo -u restricted_user kill -9 -u restricted_user
```

- Mematikan akses jaringan untuk *restricted\_user*  
Pengguna dapat mematikan akses jaringan untuk *restricted\_user* di *linux* menggunakan *iptables*. Setelah dimatikan lakukan pengujian menggunakan *ping* sebagai *restricted\_user*.
- Menolak izin menulis  
Pastikan tidak ada direktori atau *file* yang dapat ditulis oleh *restricted\_user*.
- Membatasi jumlah proses  
Pengguna dapat membatasi jumlah proses dari *restricted\_user* dengan menambahkan kode dibawah melalui *file /etc/security/limits.conf*.

Kode 2.31: Contoh kode untuk membatasi jumlah proses

```
1 restricted_user    soft    nproc    3
2 restricted_user    hard    nproc    5
```

#### 4. Menggunakan dua *server*

Pengguna dapat memakai dua *server* untuk antar muka web dan menangani permintaan web dan mengguna *server* lainnya untuk menjalankan kode yang sudah dikumpulkan. Penggunaan dua *server* mengurangi risiko menjalankan kode yang sudah dikumpulkan. Pengguna harus mengganti *source code SharIF Judge* untuk memakai hal ini.

#### 2.2.10 *Sandboxing*

*SharIF Judge* menjalankan banyak *arbitrary codes* yang pengguna kumpulkan. *SharIF Judge* harus menjalankan kode pada lingkungan terbatas sehingga memerlukan perkakas untuk *sandbox* kode yang sudah dikumpulkan. Pengguna dapat meningkatkan keamanan dengan menghidupkan *shield* dan *Sandbox*.

#### *C/C++ Sandboxing*

*SharIF Judge* menggunakan *EasySandbox* untuk melakukan *sandboxing* kode *C/C++*. *EasySandbox* berguna untuk membatasi kode yang berjalan menggunakan *seccomp*. *Seccomp* merupakan mekanisme *sandbox* pada *Linux kernel*. Secara asali *EasySandbox* dimatikan pada *SharIF Judge* namun, pengguna dapat menghidupkannya melalui halaman *Settings*. Selain itu, pengguna juga harus "*build EasySandbox*" sebelum menyalakannya. Berikut merupakan cara melakukan *build EasySandbox*:

*File EasySandbox* terdapat pada *tester/easysandbox*. Untuk membangun *EasySandbox* jalankan:

Kode 2.32: Kode *runcode.sh* awal

```
31
32 1 $ cd tester/easysandbox
33 2 $ chmod +x runalltests.sh
34 3 $ chmod +x runtest.sh
35 4 $ make runtests
```

Jika keluaran berupa *message All test passed!* maka, *EasySandbox* berhasil dibangun dan dapat dinyalakan pada perangkat lunak.

## 1 *Java Sandboxing*

2 Secara asali, *Java Sandbox* sudah dinyalakan menggunakan *Java Security Manager*. Pengguna  
3 dapat menghidupkan atau mematikannya pada halaman *Settings*.

### 4 **2.2.11 *Shield***

5 *Shield* merupakan mekanisme sangat simpel untuk melarang jalannya kode yang berpotensi berba-  
6 haya. *Shield* bukan solusi *sandboxing* karena *shield* hanya menyediakan proteksi sebagian terhadap  
7 serangan remeh. Proteksi utama terhadap kode tidak terpercaya adalah dengan menghidupkan  
8 *Sandbox*(dapat dilihat pada subbab 2.2.10).

### 9 ***Shield* untuk C/C++**

10 Dengan menghidupkan *shield* untuk *c/c++*, *SharIF Judge* hanya perlu menambahkan **#define** pada  
11 awal kode yang dikumpulkan sebelum menjalankannya. Sebagai contoh, pengguna dapat melarang  
12 penggunaan **goto** dengan menambahkan kode dibawah pada awal kode yang sudah dikumpulkan.

Kode 2.33: Kode *shield* untuk melarang penggunaan goto

```
13 #define goto YouCannotUseGoto
```

16 Dengan kode diatas, semua kode yang menggunakan **goto** akan mendapatkan *compilation error*.  
17 Apabila pengguna menghidupkan *shield*, semua kode yang mengandung **#undef** akan mendapatkan  
18 *compilation error*.

- 19 • Menghidupkan *shield* untuk C/C++

20 Pengguna dapat menghidupkan atau mematikan *shield* pada halaman *settings*.

- 21 • Menambahkan aturan untuk C/C++ Daftar aturan **#define** untuk bahasa C terdapat pada  
22 halaman *tester/shield/defc.h* dan *tester/shield/defcpp.h* untuk bahasa C++. Pengguna dapat  
23 menambahkan aturan baru pada file tersebut pada halaman *settings*. Berikut merupakan  
24 contoh sintaks pada untuk menambahkan aturan :

Kode 2.34: Sintaks aturan **#define**

```
25 1 /*
26 2
27 3 @file defc.h
28 4 There should be a newline at end of this file.
29 5 Put the message displayed to user after // in each line
30 6
31 7 e.g. If you want to disallow goto, add this line:
32 8 #define goto errorNo13 //Goto is not allowd
33 9
34 10 */
35 11
36 12 #define system errorNo1 // "system" is not allowed
37 13 #define freopen errorNo2 //File operation is not allowed
38 14 #define fopen errorNo3 //File operation is not allowed
39 15 #define fprintf errorNo4 //File operation is not allowed
40 16 #define fscanf errorNo5 //File operation is not allowed
41 17 #define feof errorNo6 //File operation is not allowed
42 18 #define fclose errorNo7 //File operation is not allowed
43 19 #define ifstream errorNo8 //File operation is not allowed
44 20 #define ofstream errorNo9 //File operation is not allowed
45 21 #define fork errorNo10 //Fork is not allowed
46 22 #define clone errorNo11 //Clone is not allowed
47 23 #define sleep errorNo12 //Sleep is not allowed
48
```

Pada akhir baris *file* `defc.h` dan `defcpp.h` harus terdapat baris baru. Terdapat banyak aturan yang tidak dapat digunakan pada *g++*, seperti aturan `#define fopen errorNo3` untuk bahasa *C++*.

### Shield untuk Python

Penggunaan *shield* untuk *python* dapat dihidupkan melalui halaman *settings*. Dengan menghidupkan *shield* untuk *python*, *SharIF Judge* hanya menambahkan beberapa kode pada baris awal kode yang sudah dikumpulkan sebelum dijalankan. Penambahan kode digunakan untuk mencegah pemakaian fungsi berbahaya. Kode-kode tersebut terletak pada *file* `tester/shield/shield_py2.py` dan `tester/shield/shield_py3.py`. Berikut merupakan cara untuk keluar dari *shield* untuk *python* menggunakan fungsi yang telah di daftar hitamkan:

Kode 2.35: Cara keluar dari *shield* untuk *python*

```

11
12.1
13.2 # @file shield_py3.py
14.3
15.4 import sys
16.5 sys.modules['os']=None
17.6
18.7 BLACKLIST = [
19.8     #'__import__', # deny importing modules
20.9     'eval', # eval is evil
21.0     'open',
22.1     'file',
23.2     'exec',
24.3     'execfile',
25.4     'compile',
26.5     'reload',
27.6     #'input'
28.7 ]
29.8 for func in BLACKLIST:
30.9     if func in __builtins__.__dict__:
31.0         del __builtins__.__dict__[func]
32.0

```

## 2.3 CodeIgniter 4[3]

*CodeIgniter 4* merupakan versi terbaru dari *framework CodeIgniter* yang berfungsi untuk membantu pembentukan web. *CodeIgniter 4* dapat dipasang menggunakan *composer* ataupun dipasang secara manual dengan mengunduhnya dari situs web resmi. Setelah dilakukan pemasangan *CodeIgniter 4* memiliki lima buah direktori dengan struktur aplikasi sebagai berikut:

- *app/*

Direktori *app* berisikan semua kode yang dibutuhkan untuk menjalankan aplikasi web yang dibentuk. Direktori ini memiliki beberapa direktori didalamnya yaitu:

- *Config/* berfungsi dalam menyimpan *file* konfigurasi aplikasi web pengguna.
- *Controllers/* berfungsi sebagai penentu alur program yang dibentuk.
- *Database/* berfungsi sebagai penyimpanan *file migrations* dan *seeds*.
- *Filters/* berfungsi dalam menyimpan *file* kelas *filter*.
- *Helpers/* berfungsi dalam menyimpan koleksi fungsi mandiri.
- *Language/* berfungsi sebagai tempat penyimpanan *string* dalam berbagai bahasa.
- *Libraries/* berfungsi dalam menyimpan kelas yang tidak termasuk kategori lain.
- *Models/* berfungsi untuk merepresentasikan data dari *database*.
- *ThirdParty/ library ThridParty* yang dapat digunakan pada aplikasi.

– **Views/** berisikan *file HTML* yang akan ditampilkan kepada pengguna.

- **public/**

Direktori *public* merupakan *web root* dari situs dan berisikan data-data yang dapat diakses oleh pengguna melalui *browser*. Direktori ini berisikan *file .htaccess*, *index.php*, dan *assets* dari aplikasi web yang dibentuk seperti gambar, *CSS*, ataupun *JavaScript*.

- **writable/**

Direktori *writable* berisikan data-data yang mungkin perlu ditulis seperti *file cache, logs*, dan *uploads*. Pengguna dapat menambahkan direktori baru sesuai dengan kebutuhan sehingga menambahkan keamanan pada direktori utama.

- **tests/**

Direktori ini menyimpan *file test* dan tidak perlu dipindahkan ke *server* produksi. Direktori *\_support* berisikan berbagai jenis kelas *mock* dan keperluan yang dapat dipakai pengguna dalam menulis *tests*.

- **vendor/** atau **system/**

Direktori ini berisikan *file* yang diperlukan dalam menjalani *framework* dan tidak boleh dimodifikasi oleh pengguna. Pengguna dapat melakukan *extend* atau membangun kelas baru untuk menambahkan fungsi yang diperlukan.

### 2.3.1 Models- Views-Controllers

*CodeIgniter 4* menggunakan pola *MVC* untuk mengatur *file* agar lebih simpel dalam menemukan *file* yang diperlukan. *MVC* menyimpan data, presentasi, dan alur program dalam *file* yang berbeda.

#### Models

*Models* berfungsi dalam menyimpan dan mengambil data dari tabel spesifik pada *database*. Data tersebut dapat berupa pengguna, pemberitahuan blog, transaksi, dll. *Models* pada umumnya disimpan pada direktori **app/Models** dan memiliki *namespace* sesuai dengan lokasi dari *file* tersebut. Kode 2.36 merupakan contoh dari *models*.

Kode 2.36: Contoh *Models*

```

26
27 1
28 2 <?php
29 3
30 4 namespace App\Models;
31 5
32 6 use CodeIgniter\Model;
33 7
34 8 class UserModel extends Model
35 9 {
36 0     protected $table      = 'users';
37 1     protected $primaryKey = 'id';
38 2
39 3     protected $useAutoIncrement = true;
40 4
41 5     protected $returnType     = 'array';
42 6     protected $useSoftDeletes = true;
43 7
44 8     protected $allowedFields = ['name', 'email'];
45 9
46 0     // Dates
47 1     protected $useTimestamps = false;
48 2     protected $dateFormat     = 'datetime';
49 3     protected $createdField    = 'created_at';
50 4     protected $updatedField    = 'updated_at';
51 5     protected $deletedField    = 'deleted_at';
52 6

```

```

27 // Validation
28 protected $validationRules    = [];
29 protected $validationMessages = [];
40 protected $skipValidation     = false;
51 protected $cleanValidationRules = true;
62
63 }

```

Kode 2.36 merupakan contoh *Models* yang dapat digunakan pada *controllers*. *Models* tersebut terkoneksi dengan tabel **users** dengan *primarykey* **id**. *Model* pada *CodeIgniter 4* juga dapat digunakan untuk mencari, menyimpan, dan menghapus data untuk setiap tabel spesifik. Kode 2.37 merupakan contoh penggunaan *model* untuk mencari data spesifik.

Kode 2.37: Contoh penggunaan *model* untuk mencari data spesifik

```

13
14 1 <?php
15 2
16 3 $users = $userModel->where('active', 1)->findAll();

```

Kode 2.37 menggabungkan *query* dengan metode pencarian *model* untuk mencari seluruh data **active**.

## Views

*Views* merupakan halaman berisikan *HTML* dan sedikit *PHP* yang ditampilkan kepada pengguna ataupun dapat berupa pecahan halaman seperti *header*, *footer*, ataupun *sidebar*. *Views* biasanya terdapat pada **app/Views** dan mendapatkan data berupa variabel dari *controller* untuk ditampilkan.

Kode 2.38: Contoh *Views*

```

24
25 1 <html>
26 2   <head>
27 3     <title>My Blog</title>
28 4   </head>
29 5   <body>
30 6     <h1>Welcome to my Blog!</h1>
31 7   </body>
32 8 </html>

```

Kode 2.39 merupakan contoh *views* pada direktori **app/Views** yang akan menampilkan tulisan *Welcome to my Blog*. *View* ini dapat ditampilkan melalui *controller* seperti berikut:

Kode 2.39: Contoh *Views*

```

36
37 1 <?php
38 2
39 3 namespace App\Controllers;
40 4
41 5 use CodeIgniter\Controller;
42 6
43 7 class Blog extends Controller
44 8 {
45 9     public function index()
46 10     {
47 11         return view('blog_view');
48 12     }
49 13 }

```

Kode 2.39 merupakan contoh memanggil *views* pada *file controllers*. Kode ini mengembalikan halaman **blog\_view** pada *controller* **blog**.

## Controllers

*Contollers* merupakan kelas untuk mengambil atau memberikan data dari *models* menuju *views* untuk ditampilkan. Setiap pembentukan *controllers* dibentuk harus memperpanjang kelas *BaseController*.

1 Kode 2.40 merupakan contoh *controllers* yang dibentuk.

Kode 2.40: Contoh *Controllers* pada *CodeIgniter 4*

```

2
31 <?php
42
53 namespace App\Controllers;
64
75 class HelloWorld extends BaseController
86 {
97     public function index()
108     {
119         return 'Hello World!';
120     }
131
142     public function comment()
153     {
164         return 'I am not flat!';
175     }
186 }

```

20 Kode 2.40 merupakan contoh *controllers* yang melakukan pengembalian *Hello World* pada url:

21 `example.com/index.php/helloworld/`

22 Selain itu, *CodeIgniter 4* menyediakan fungsi bernama *Controller Filters* yang berfungsi untuk  
 23 membiarkan pengguna membangun sebuah kondisi sebelum ataupun sesudah *controller* dijalankan.

24 Kode 2.41 merupakan contoh penggunaan *filters*.

Kode 2.41: Contoh *Controllers Filters* pada *CodeIgniter 4*

```

25
26 1 <?php
27 2
28 3 namespace App\Filters;
29 4
30 5 use CodeIgniter\Filters\FilterInterface;
31 6 use CodeIgniter\HTTP\RequestInterface;
32 7 use CodeIgniter\HTTP\ResponseInterface;
33 8 use Config\Database;
34 9
35 0 class MyFilter implements FilterInterface
36 1 {
37 2     public function before(RequestInterface $request, $arguments = null)
38 3     {
39 4         $session = \Config\Services::session();
40 5         $db = Database::connect();
41 6
42 7         if ( !$db->tableExists('sessions'))
43 8             return redirect()->to('/install');
44 9         if ( !$session->get('logged_in')) // if not logged in
45 0             return redirect()->to('/login');
46 1     }
47 2
48 3     public function after(RequestInterface $request, ResponseInterface $response, $arguments = null)
49 4     {
50 5         // Do something here
51 6     }
52 7 }

```

54 Kode 2.41 merupakan contoh kode yang akan melakukan pengecekan tabel ataupun *session*  
 55 sebelum *controller* dijalankan. Selanjutnya pengguna perlu menambahkan konfigurasi *filter* pada  
 56 *routes* seperti sintaks berikut.

57

58 `$routes->get('/', 'Dashboard::index', ['filter' => 'check-install:dual,noreturn']);`

59 Sintaks diatas akan melakukan pengecekan pada *controller* `Dashboard::index` sebelum dan  
 60 setelah *controller* tersebut dijalankan.

### 2.3.2 *CodeIgniter URLs*

*CodeIgniter 4* menggunakan pendekatan *segment-based* dibandingkan menggunakan *query-string* untuk menghasilkan *URL* sehingga ramah manusia dan mesin pencari. Berikut merupakan contoh *URL* yang dihasilkan *CodeIgniter 4*:

```
https://www.example.com/ci-blog/blog/news/2022/10?page=2
```

*CodeIgniter 4* menghasilkan *URL* seperti diatas dengan membaginya menjadi:

- *Base URL* merupakan *URL* dasar dari aplikasi web yang dibentuk yaitu `https://www.example.com/ci-blog`
- *URI Path* merupakan alamat yang dituju yaitu `/ci-blog/blog/news/2022/10`
- *Route* juga merupakan alamat yang dituju tanpa *URL* dasar yaitu `/blog/news/2022/10`
- *Query* merupakan hasil dari *query* yang ingin ditampilkan yaitu `page=2`

Secara asali *CodeIgniter 4* membangun *URL* dengan `index.php` namun, pengguna dapat menghapus `file index.php` pada *URL* yang dibentuk. Pengguna dapat menghapus `index.php` sesuai dengan *server* yang digunakan. Berikut merupakan contoh dua buah *server* yang biasanya dipakai:

#### *Apache Web Server*

Pengguna dapat *URL* melalui `file .htaccess` dengan menyalakan ekstensi `mod_rewrite`. Kode 2.42 merupakan contoh `file .htaccess` untuk menghapus `index.php` pada *URL* yang dibentuk.

Kode 2.42: Contoh `file .htaccess` pada *Apache Web Server*.

```
18 RewriteEngine On
19 1 RewriteCond %{REQUEST_FILENAME} !-f
20 2 RewriteCond %{REQUEST_FILENAME} !-d
21 3 RewriteRule ^(.*)$ index.php/$1 [L]
```

*File* diatas memperlakukan semua *HTTP Request* selain dari direktori dan *file* yang ada sebagai permintaan.

#### *NGINX*

Pengguna dapat mengubah *URL* menggunakan `try_files` yang akan mencari *URI* dan mengirimkan permintaan pada *URL* yang ingin dihilangkan. Kode 2.43 merupakan contoh penggunaan `try_files` untuk menghapus `index.php` pada *URL*.

Kode 2.43: Contoh penggunaan `try-files`.

```
30 location / {
31 1 try_files $uri $uri/ /index.php$is_args$args;
32 2
33 3 }
```

### 2.3.3 *URI Routing*

*CodeIgniter 4* menyediakan dua buah *routing* yakni:



## Defined Route Routing

Pengguna dapat mendefinisikan *route* secara manual untuk *URL* yang lebih fleksibel. Kode 2.44 merupakan contoh *route* yang didefinisikan secara manual.

Kode 2.44: Contoh *route* yang didefinisikan secara manual

```

1 <?php
2
3 $routes->get('product/(:num)', 'Catalog::productLookup');
```

Kode 2.44 merupakan contoh penggunaan *route* untuk menuju kelas *Catalog* dengan metode *productLookup*. Pengguna juga dapat memakai beberapa *HTTP verb* seperti *GET, POST, PUT, etc.* Selain menulis secara individu, pengguna dapat melakukan *grouping* pada *route* seperti Kode.

Kode 2.45: Contoh *route* yang menggunakan *grouping* manual

```

12 <?php
13 1
14 2
15 3 $routes->group('admin', static function ($routes) {
16 4     $routes->get('users', 'Admin\Users::index');
17 5     $routes->get('blog', 'Admin\Blog::index');
18 6 });
```

Kode 2.45 merupakan contoh penggunaan *grouping* untuk *URI* *admin/users* dan *admin/blog*.

## Auto Routing

Pengguna dapat mendefinisikan *route* secara otomatis melalui fitur *Auto Routing* apabila tidak terdapat *route*. Pengguna dapat menyalakan fitur ini pada *app/Config/Routes.php* dengan cara berikut:

```
$routes->setAutoRoute(true);
```

Pengguna juga perlu mengubah *\$autoRoutesImproved* menjadi *true* pada direktori *app/Config/Feature.php*. Selain menggunakan *auto routing* baru, pengguna dapat menggunakan *Auto Routing (Legacy)* yang terdapat pada *CodeIgniter 3* dengan cara seperti berikut:

### 2.3.4 Databases

*CodeIgniter 4* menyediakan kelas *database* yang dapat menyimpan, memasukan, memperbarui, dan menghapus data pada *database* sesuai dengan konfigurasi. Pengguna dapat melakukan konfigurasi untuk *database* yang ingin dikoneksikan melalui direktori *app/Config/Database.php* atau file *.env*. Kode 2.46 merupakan contoh pada direktori *Database.php*.

Kode 2.46: Contoh konfigurasi *database* pada *CodeIgniter 4*.

```

34 <?php
35 1
36 2
37 3 namespace Config;
38 4
39 5 use CodeIgniter\Database\Config;
40 6
41 7 class Database extends Config
42 8 {
43 9     public $default = [
44 10         'DSN' => '',
45 11         'hostname' => 'localhost',
46 12         'username' => 'root',
47 13         'password' => '',
48 14         'database' => 'database_name',
```

```

115         'DBDriver' => 'MySQLi',
116         'DBPrefix' => '',
117         'pConnect' => true,
118         'DBDebug' => true,
119         'charset' => 'utf8',
120         'DBCollat' => 'utf8_general_ci',
121         'swapPre' => '',
122         'encrypt' => false,
123         'compress' => false,
124         'strictOn' => false,
125         'failover' => [],
126         'port' => 3306,
127     ];
128
129     // ...
130 }

```

Kode 2.46 merupakan contoh konfigurasi untuk database bernama `database_name` dengan `username` root. Selain untuk melakukan koneksi `database`, kelas ini dapat digunakan untuk menambahkan, menghapus, dan memperbaharui data pada `database`. Berikut merupakan contoh penggunaan `query` pada `database`:

Kode 2.47: Contoh konfigurasi `database` pada *CodeIgniter 4*.

```

22
23 1 <?php
24 2
25 3 $builder = $db->table('users');
26 4 $builder->select('title, content, date');
27 5 $builder->from('mytable');
28 6 $query = $builder->get();

```

Kode 2.47 merupakan contoh penggunaan `query` untuk mengambil data `title, content`, dan `date` pada tabel `mytable`. *CodeIgniter 4* juga menyediakan fitur untuk membangun `database` melalui fitur bernama *Database Forge*. Pengguna dapat membangun, mengubah, menghapus tabel dan juga menambahkan `field` pada tabel tersebut. Kode 2.48 merupakan contoh pembentukan `database`.

Kode 2.48: Contoh pembentukan tabel melalui *database forge*.

```

34
35 1 <?php
36 2
37 3 $fields = [
38 4     'id' => [
39 5         'type' => 'INT',
40 6         'constraint' => 5,
41 7         'unsigned' => true,
42 8         'auto_increment' => true,
43 9     ],
44 0     'title' => [
45 1         'type' => 'VARCHAR',
46 2         'constraint' => '100',
47 3         'unique' => true,
48 4     ],
49 5     'author' => [
50 6         'type' => 'VARCHAR',
51 7         'constraint' => 100,
52 8         'default' => 'King of Town',
53 9     ],
54 0     'description' => [
55 1         'type' => 'TEXT',
56 2         'null' => true,
57 3     ],
58 4     'status' => [
59 5         'type' => 'ENUM',
60 6         'constraint' => ['publish', 'pending', 'draft'],
61 7         'default' => 'pending',
62 8     ],
63 9 ];
64 0 $forge->addField($fields);
65 1 $forge->createTable('table_name');

```

Kode merupakan contoh pembentukan `database` dengan tabel bernama `table_name` yang berisikan beberapa `field`.

### 2.3.5 Library

*CodeIgniter 4* menyediakan berbagai *library* untuk membantu pengguna dalam pembentukan aplikasi web. Berikut merupakan contoh *library* yang disediakan oleh *CodeIgniter 4*:

#### Kelas *Email*

*CodeIgniter* menyediakan kelas *email* dengan fitur sebagai berikut:

- Beberapa Protokol: *Mail*, *Sendmail*, dan *SMTP*
- Enkripsi *TLS* dan *SSL* untuk *SMTP*
- Beberapa Penerima
- *CC* dan *BCCs*
- *HTML* atau *email* teks biasa
- Lampiran
- Pembungkus kata
- Prioritas
- Mode *BCC Batch*, memisahkan daftar *email* besar menjadi beberapa *BCC* kecil.
- Alat *Debugging email*

Pengguna dapat melakukan konfigurasi pada file `app/Config/Email.php` untuk melakukan pengiriman *email*. Kode 2.49 merupakan contoh konfigurasi preferensi *email* secara manual.

Kode 2.49: Contoh kode untuk melakukan konfigurasi *email*.

```

18 <?php
19 1
20 2
21 3 $config['protocol'] = 'sendmail';
22 4 $config['mailPath'] = '/usr/sbin/sendmail';
23 5 $config['charset'] = 'iso-8859-1';
24 6 $config['wordWrap'] = true;
25 7
26 8 $email->initialize($config);
27 8

```

Selain itu, pengguna dapat melakukan pengiriman *email* sesuai dengan kebutuhan. Kode 2.50 merupakan contoh penggunaan kelas *email* untuk mengirim *email*.

Kode 2.50: Contoh kode untuk melakukan pengiriman *email*.

```

30 <?php
31 1
32 2
33 3 $email = \Config\Services::email();
34 4
35 5 $email->setFrom('your@example.com', 'Your Name');
36 6 $email->setTo('someone@example.com');
37 7 $email->setCC('another@another-example.com');
38 8 $email->setBCC('them@their-example.com');
39 9
40 0 $email->setSubject('Email Test');
41 1 $email->setMessage('Testing the email class. ');
42 2
43 3 $email->send();
44 3

```

Kode 2.50 merupakan contoh penggunaan kelas *email* untuk mengirim *email* dari `your@example.com` kepada `someone@example.com` dengan subjek `Email Test` dan pesan `Testing the email class.`

#### *Working with Uploaded Files*

Pengunggahan *file* terdapat empat buah proses sebagai berikut:

1. Dibentuk sebuah form untuk pengguna memilih dan mengunggah *file*.

2. Setelah *file* diunggah, *file* akan dipindahkan menuju direktori yang dipilih.
  3. Pada pengiriman dan pemindahan *file* dilakukan validasi sesuai dengan ketentuan yang ada.
  4. Setelah *file* diterima akan dikeluarkan pesan berhasil.
- Perangkat lunak akan menerima *file* dari *views* yang nantinya akan dilakukan validasi pada *controller*.  
 Kode 2.51 merupakan contoh *view* untuk melakukan pengunggahan *file*.

Kode 2.51: Contoh kode untuk melakukan pengunggahan *file*.

```

6 <!DOCTYPE html>
71 <html lang="en">
82 <head>
93 <title>Upload Form</title>
104 </head>
115 <body>
126
137 <?php foreach ($errors as $error): ?>
148 <li><?= esc($error) ?></li>
159 <?php endforeach ?>
171
182 <?= form_open_multipart('upload/upload') ?>
193 <input type="file" name="userfile" size="20">
204 <br><br>
215 <input type="submit" value="upload">
226 </form>
237
248 </body>
259 </html>

```

- Kode 2.51 merupakan contoh *file view* menggunakan *form helper* dan dapat memberitahu apabila terdapat *error*. Setelah dilakukan penerimaan *file*, perangkat lunak akan mengirimkan *file* kepada *controller* untuk dilakukan validasi dan penyimpanan. Kode merupakan contoh *controller* untuk melakukan validasi dan penyimpanan.

Kode 2.52: Contoh kode *controller* untuk melakukan validasi dan penyimpanan.

```

31 <?php
32
33 namespace App\Controllers;
34
35 use CodeIgniter\Files\File;
36
37 class Upload extends BaseController
38 {
39     protected $helpers = ['form'];
40
41     public function index()
42     {
43         return view('upload_form', ['errors' => []]);
44     }
45
46     public function upload()
47     {
48         $validationRule = [
49             'userfile' => [
50                 'label' => 'Image File',
51                 'rules' => [
52                     'uploaded[userfile]',
53                     'is_image[userfile]',
54                     'mime_in[userfile,image/jpeg,image/gif,image/png,image/webp]',
55                     'max_size[userfile,100]',
56                     'max_dims[userfile,1024,768]',
57                 ],
58             ],
59         ];
60         if (! $this->validate($validationRule)) {
61             $data = ['errors' => $this->validator->getErrors()];
62             return view('upload_form', $data);
63         }
64
65         $img = $this->request->getFile('userfile');
66
67         if (! $img->hasMoved()) {
68             $filepath = WRITEPATH . 'uploads/' . $img->store();
69

```

```

110
121         $data = ['uploaded_fileinfo' => new File($filepath)];
132
143         return view('upload_success', $data);
154     }
165
176     $data = ['errors' => 'The file has already been moved.'];
187
198     return view('upload_form', $data);
109 }
110 }

```

13 Kode 2.52 terdapat dua buah fungsi yaitu:

- 14 • `index()` yang mengembalikan *view* bernama `upload_form`
- 15 • `upload()` yang memberikan aturan untuk melakukan validasi dan melakukan penyimpanan
- 16 pada direktori `uploads`.

### 17 2.3.6 *Helpers*

18 *Helpers* merupakan fungsi pada *CodeIgniter 4* yang menyediakan beberapa fungsi untuk pengguna  
 19 dalam membangun aplikasi web. *Helpers* dapat dimuat oleh pengguna seperti berikut:

```

20
21             <?php
                helper('helpers_name');

```

22 Setelah dilakukan pemanggilan, pengguna dapat memakai fungsi-fungsi yang disediakan sesuai  
 23 dengan *helpers* yang digunakan.

## 24 2.4 Koversi CodeIgniter 3 ke CodeIgniter 4[3]

25 Konversi CodeIgniter 3 ke CodeIgniter 4 diperlukan penulisan ulang karena terdapat banyak  
 26 implementasi yang berbeda. Konversi ke CodeIgniter 4 diawali dengan melakukan instalasi proyek  
 27 baru CodeIgniter 4.

### 28 2.4.1 Struktur Aplikasi

29 Struktur direktori pada CodeIgniter 4 memiliki perubahan yang terdiri *app*, *public*, dan *writable*.  
 30 Direktori *app* merupakan perubahan dari direktori *application* dengan isi yang hampir sama dengan  
 31 beberapa perubahan nama dan perpindahan direktori. Pada CodeIgniter 4 terdapat direktori *public*  
 32 yang bertujuan sebagai direktori utama pada aplikasi website. Selanjutnya terdapat direktori  
 33 *writable* yang berisikan *cache data*, *logs*, dan *session data*.

### 34 2.4.2 *Routing*

35 CodeIgniter 4 meletakkan *route* pada file `app\Config\Routes.php`. CodeIgniter 4 memiliki fitur  
 36 *auto routing* seperti pada CodeIgniter 3 namun, pada *default* di matikan. Fitur *auto routing*  
 37 memungkinkan untuk dinyalakan serupa dengan pada CodeIgniter 3 namun tidak direkomendasikan  
 38 karena alasan *security*.

### 2.4.3 Model, View, and Controller

Struktur MVC pada CodeIgniter 4 berbeda dibandingkan CodeIgniter 3 dimana terdapat perbedaan penyimpanan direktori untuk ketiga *file* tersebut. Berikut merupakan penjelasan mengenai struktur MVC:

#### Model

*Model* terdapat pada direktori `app\Models`. Pembentukan *file* untuk *Model* perlu ditambahkan `namespace App\Models;` dan `use CodeIgniter\Model;` pada awal *file* setelah membuka tag PHP. Selanjutnya nama fungsi perlu diubah dari `extends CI_Model` menjadi `extends Model`. *Model* dapat dilakukan pembaharuan melalui cara berikut:

1. Pertama pengguna harus memindahkan seluruh *file model* menuju direktori `app/Models`
2. Selanjutnya pengguna harus menambahkan `namespace App\Models;` setelah pembukaan *tag PHP*.
3. Pengguna juga harus menambahkan `use CodeIgniter\Model;` setelah kode diatas.
4. Pengguna harus mengganti `extends CI_Model` menjadi `extends Model`.
5. Terakhir pemanggilan *model* berubah dari sintaks:

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

menjadi sintaks berikut:

```
$this->x = new X();
```

#### View

*View* pada CodeIgniter 4 terdapat di `app/Views` dengan sintaks yang harus diubah. Sintaks yang harus diubah merupakan sintaks untuk memanggil *view* pada CodeIgniter 3 `$this->load->view('x');` sedangkan pada CodeIgniter 4 dapat menggunakan `return view('x');`. Selanjutnya, sintaks `<?php echo $title?>` pada halaman *view* dapat diubah menjadi `<?= $title ?>`. Berikut merupakan cara melakukan pembaharuan *view*:

1. Pertama pengguna perlu memindahkan seluruh *file views* menuju `app/Views`
2. Selanjutnya pengguna perlu mengubah sintaks:

```
$this->load->view('directory_name/file_name')
```

menjadi sintaks berikut:

```
return view('directory_name/file_name');
```

3. Pengguna juga perlu mengubah sintaks:

```
$content = $this->load->view('file', $data, TRUE);
```

menjadi sintaks berikut:

```
$content = view('file', $data);
```

4. Pada *file views* pengguna dapat mengubah sintaks:

```
<?php echo $title; ?>
```

menjadi sintaks berikut:

```
<?= $title ?>.
```

5. Pengguna juga perlu menghapus apabila terdapat sintaks `defined('BASEPATH') OR exit('No direct script access allowed');`.

## Controller

*Controller* pada CodeIgniter 4 terdapat di `app\Controllers` dan diperlukan beberapa perubahan.

Pertama, perlu ditambahkan namespace `App\Controllers`; pada awal *file* setelah membuka tag

PHP. Selanjutnya, perlu mengubah `extends CI_Controller` menjadi `extends BaseController`.

Selanjutnya, diperlukan pengubahan nama pada pemanggilan *file* menjadi `App\Controllers\User.php`.

Pengguna dapat melakukan pembaharuan *controller* menggunakan cara berikut:

1. Pertama pengguna harus memindahkan seluruh *file controller* menuju `app/Controllers`.
2. Pengguna juga harus menambahkan sintaks namespace `App\Controllers`; setelah pembukaan tag PHP.
3. Selanjutnya pengguna harus mengubah `extends CI_Controller` menjadi `extends BaseController`.
4. Pengguna juga harus menghapus baris `defined('BASEPATH') OR exit('No direct script access allowed');` apabila ada.

### 2.4.4 Libraries

CodeIgniter 4 menyediakan *library* untuk digunakan dan dapat diinstall apabila diperlukan. Pe-

manggilan *library* berubah dari `$this->load->library('x');` menjadi `$this->x = new X();`.

Terdapat beberapa *library* yang harus di perbaharui dengan sedikit perubahan. Berikut merupakan

beberapa *libraries* yang terdapat pembaharuan:

## Configuration

*File configuration* CodeIgniter 4 terdapat pada `app\Config` dengan penulisan sedikit berbeda dengan

versi sebelumnya. Pengguna hanya perlu melakukan pemindahan menuju CodeIgniter 4 dan apabila

menggunakan *file config custom* maka, diperlukan penulisan ulang pada direktori Config dengan

melakukan *extend* pada `CodeIgniter\Config\BaseConfig`.

## Database

Penggunaan *database* pada CodeIgniter 4 hanya berubah sedikit dibandingkan dengan versi se-

belumnya. Data-data penting kredensial diletakan pada `app\Config\Database.php` dan perlu

dilakukan beberapa perubahan sintaks dan *query*. Sintaks untuk memuat database diubah men-

jadi `$db = db_connect();` dan apabila menggunakan beberapa *database* maka sintaks menjadi

`$db = db_connect('group_name');`. Semua *query* harus diubah dari `$this->db` menjadi `$db`

dan beberapa sintaks perlu diubah seperti `$query->result();` menjadi `$query->getResult();`.

Selain itu, terdapat *class* baru yakni *Query Builder Class* yang harus di inisiasi `$builder =`

1 `$db->table('mytable');` dan dapat dipakai untuk menjalankan *query* dengan mengganti `$this->db`  
 2 seperti `$this->db->get();` menjadi `$builder->get();`.

### 3 ***Emails***

4 Perubahan *email* hanya terdapat pada nama dari *method* dan pemanggilan *library email*. Pemanggil-  
 5 an *library* berubah dari `$this->load->library('email');` menjadi `$email = service('email');`  
 6 dan selanjutnya perlu dilakukan perubahan pada semua `$this->email` menjadi `$email`. Selanjut-  
 7 nya beberapa pemanggilan *method* berubah dengan tambahan *set* didepannya seperti *from* menjadi  
 8 *setFrom*.

### 9 ***Working with Uploaded Files***

10 Terdapat banyak perubahan dimana pada CodeIgniter 4 pengguna dapat mengecek apakah *file* telah  
 11 terunggah tanpa *error* dan lebih mudah untuk melakukan penyimpanan *file*. Pada CodeIgniter 4  
 12 melakukan akses pada *uploaded file* dilakukan dengan sintaks berikut:

```
13 $file = $this->request->getFile('userfile')
```

14 selanjutnya dapat dilakukan validasi dengan cara sebagai berikut:

```
15 $file->isValid()
```

16 Penyimpanan *file* yang sudah diunggah dapat dilakukan dengan `$path = $this->request->getFile('userfile')`  
 17 *File* yang sudah diunggah dan di validasi akan tersimpan pada `writable/uploads/head_img/user_name.jpg`.

### 18 ***HTML Tables***

19 Tidak terdapat banyak perubahan pada *framework* versi terbaru hanya perubahan pada nama *method*  
 20 dan pemanggilan *library*. Perubahan pemanggilan *library* dari `$this->load->library('table');`  
 21 menjadi `$table = new \CodeIgniter\View\Table();` dan perlu dilakukan perubahan setiap  
 22 `$this->table` menjadi `$table`. Selain itu, terdapat bebera perubahan pada penamaan *method*  
 23 dari *underscoped* menjadi *camelCase*.

### 24 ***Localization***

25 *CodeIgniter 4* mengembalikan *file* bahasa menjadi *array* sehingga perlu dilakukan beberapa perubah-  
 26 an. Pertama, perlu dilakukan konfigurasi *default language* pada perangkat lunak. Selanjutnya mela-  
 27 kukan pemindahan *file* bahasa pada *CodeIgniter 3* menuju `app\Language\<locale>`. *File-file* baha-  
 28 sa *CodeIgniter 3* perlu dilakukan penghapusan semua kode `$this->lang->load($file, $lang);`  
 29 dan mengubah *method* pemanggilan bahasa dari `$this->lang->line('error_email_missing')`  
 30 menjadi `echo lang('Errors.errorMessageMissing');`

### 31 ***Migrations***

32 Perubahan perlu dilakukan pada nama *file* menjadi nama dengan cap waktu. Selanjutnya dilakuk-  
 33 an penghapusan kode `defined('BASEPATH') OR exit('No direct script access allowed');`  
 34 dan menambahkan dua buah kode setelah membuka tag PHP yaitu:



- namespace App\Database\Migrations;
- use CodeIgniter/Database/Migration;

Setelah itu, `extends CI_Migration` diubah menjadi `extends Migration`. Terakhir, terdapat perubahan pada nama *method Forge* dari `$this->dbforge->add_field` menjadi *camelCase* `$this->forge->addField`.

## Routing

Pengguna dapat melakukan pembaharuan *routing* dengan cara berikut:

1. Pengguna dapat memakai *Auto Routing* 2.3.3 seperti pada *CodeIgniter 3* dengan menyalakan *Auto Routing(Legacy)*.
2. Terdapat perubahan dari `(:any)` menjadi `(:segment)`.
3. Pengguna juga harus mengubah sintaks pada `app/Config/Routes.php` seperti berikut:
  - `$route['journals'] = 'blogs';` menjadi `$routes->add('journals', 'Blogs::index');` untuk memanggil fungsi `index` pada *controller* `Blogs`.
  - `$route['product/(:any)'] = 'catalog/product_lookup'` menjadi `$routes->add('product/(:segment)', 'Catalog::productLookup');`.

## Validations

Pengguna dapat melakukan pembaharuan pada *validations* melalui cara berikut:

1. Pengguna harus mengubah kode pada *view* dari `<?php echo validation_errors(); ?>` menjadi `<?= validation_list_errors() ?>`
2. Pengguna perlu mengubah beberapa kode pada *controller* seperti berikut:
  - `$this->load->helper(array('form', 'url'));` menjadi `helper(['form', 'url']);`
  - Pengguna perlu menghapus kode `$this->load->library('form_validation');`
  - `if ($this->form_validation->run() == FALSE)` menjadi `if (! $this->validate([]))`
  - `$this->load->view('myform');` menjadi seperti berikut:
 

```
return view('myform', ['validation' => $this->validator,]);
```
3. Pengguna juga perlu mengubah kode (dapat dilihat pada kode 2.53) untuk melakukan validasi.

Kode 2.53: Perubahan kode untuk melakukan validasi.

```
<?php
$isValid = $this->validate([
    'name' => 'required|min_length[3]',
    'email' => 'required|valid_email',
    'phone' => 'required|numeric|max_length[10]',
]);
```

### 2.4.5 Helpers

*Helpers* tidak terdapat banyak perubahan namun, beberapa *helpers* pada *CodeIgniter 3* tidak terdapat pada *CodeIgniter 4* sehingga perlu perubahan pada implementasi fungsinya. *Helpers* dapat di dimuat secara otomatis menggunakan `app\Config\Autoload.php`

### 2.4.6 Events

*Events* merupakan pembaharuan dari *Hooks*. Pengguna harus mengubah

```
1          $hook['post_controller_constructor']  
2  menjadi  
3          Events::on('post_controller_constructor', ['MyClass', 'MyFunction']);}  
4  Dan menambahkan namespace CodeIgniter\Events\Events;.
```

#### 5 **2.4.7 Framework**

6 Pengguna tidak membutuhkan direktori *core* dan tidak membutuhkan kelas MY\_X pada direktori  
7 *libraries* untuk memperpanjang atau mengganti potongan *CI4*. Pengguna dapat membangun kelas  
8 dimanapun dan menambahkan metode pada `app/Config/Services.php`.

## BAB 3

## ANALISIS

### 3.1 Analisis konversi menuju *CodeIgniter 4*

Konversi *CodeIgniter 3* menuju *CodeIgniter 4* diperlukan penulisan ulang karena terdapat perubahan struktur aplikasi dan beberapa fungsi yang memiliki pemanggilan berbeda dan harus dilakukan pembaharuan.

#### 3.1.1 Persiapan *CodeIgniter 4*

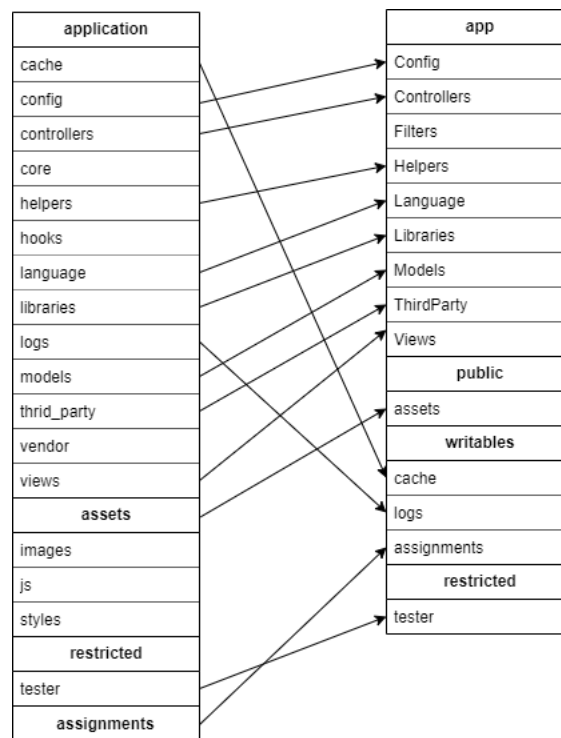
Konversi dimulai dengan mempersiapkan aplikasi *CodeIgniter 4* dengan mengunduh ataupun memasangnya melalui *Composer*. Pengguna juga perlu memasang komponen pendukung seperti *Twig*, *phpoffice*, *radius*, dan *adldap2*.

#### 3.1.2 Pemindahan *file* dari *CodeIgniter 3* ke *CodeIgniter 4*

#### 3.1.3 Pembaharuan *file-file CodeIgniter 3*

#### 3.1.4 Struktur Aplikasi

Struktur aplikasi pada *CodeIgniter 3* dan *CodeIgniter 4* memiliki perubahan sehingga perlu dilakukan pemindahan *file-file* menuju *CodeIgniter 4*. Gambar 3.1 merupakan pemindahan struktur aplikasi *SharIF Judge* pada *CodeIgniter 3* menuju *CodeIgniter 4*.



Gambar 3.1: Pemindahan struktur aplikasi menuju CodeIgniter 4

Berikut merupakan rincian direktori yang akan dipindahkan menuju *CodeIgniter 4*.

## Application

Direktori-direktori **application** pada *CodeIgniter 3* akan dipindahkan dengan penyesuaian menuju direktori **app** terkecuali direktori **vendor**, **cache** dan **core**. Berikut merupakan direktori yang dipindahkan dari direktori *application* menuju direktori *app*.

- **application/config** akan dipindahkan menuju **app/Config**.
- **application/controllers** akan dipindahkan menuju **app/Controllers**.
- **application/helpers** akan dipindahkan menuju **app/Helpers**.
- **application/language** akan dipindahkan menuju **app/Language**.
- **application/libraries** akan dipindahkan menuju **app/Libraries**.
- **application/models** akan dipindahkan menuju **app/Models**.
- **application/views** akan dipindahkan menuju **app/Views**.

## Public

*CodeIgniter 3* tidak menyediakan direktori akar berupa *public* sehingga terdapat perubahan struktur dimana direktori yang sebelumnya ada pada **application** akan dipindah menuju direktori **public**. Berikut merupakan direktori yang dipindahkan menuju direktori **public**.

- **assets** akan dipindahkan menuju **public/assets**.

Selain stuktur aplikasi diatas, *SharIF Judge* memiliki dua buah direktori terpisah diluar direktori utama bernama *assignments* dan *tester*. Direktori *assignments* ini berfungsi untuk menyimpan seluruh *file* yang telah dikumpulkan sedangkan direktori *tester* digunakan untuk melakukan perco-

1 baan untuk keamanan *sandbox*. Kedua direktori ini harus dapat ditulis oleh *PHP* sehingga akan  
 2 dipindahkan menuju direktori *writables*.

### 3 *Writables*

4 Direktori ini merupakan direktori berisikan seluruh direktori yang dapat ditulis oleh *PHP*. Berikut  
 5 merupakan direktori yang dipindahkan menuju *writables*.

- 6 • `application/cache` akan dipindahkan menuju `writables/cache`.
- 7 • `application/log` akan dipindahkan menuju `writables/logs`.
- 8 • `assignments` akan dipindahkan menuju `writables/assignments`.

### 9 **3.1.5 Routing**

10 *Routing* pada aplikasi *SharIF Judge* menggunakan *auto routing* yang telah disediakan oleh *CodeIgniter 3*. *Auto routing* akan membentuk *url* sesuai dengan *controller* dan *method* yang telah dibentuk  
 11 tanpa harus didefinisikan secara manual. Penggunaan *auto routing* seperti pada *CodeIgniter 3*  
 12 memiliki kekurangan pada bagian keamanan dimana *filter* pada *controller* dan proteksi *CSRF* akan  
 13 dilewati. Sehingga, konversi pada aplikasi *SharIF Judge* akan menggunakan *URI Routing* yang  
 14 didefinisikan secara manual untuk alasan keamanan dan *url* yang fleksibel. Berikut merupakan  
 15 contoh *route* yang didefinisikan secara manual.

```
17 $routes->post("login/register",'Login::register');
```

18 *Route* akan didefinisikan secara eksplisit sesuai dengan fungsinya.

### 19 **3.1.6 Model, View, and Controller**

20 *CodeIgniter 4* memiliki perubahan baik dari kegunaan dan cara pemanggilan *Model*, *View*, and  
 21 *Controller*. Berikut merupakan perubahan yang terjadi:

#### 22 *Model*

23 *Model* pada *CodeIgniter 4* memiliki perubahan dimana *model* dapat digunakan untuk mengambil  
 24 data pada satu buah tabel spesifik. Konversi *model* dari *CodeIgniter 3* menuju *CodeIgniter 4* dapat  
 25 dilakukan menggunakan dua buah cara yakni:

- 26 1. Menggunakan *Model* dari *CodeIgniter 3*

27 *Model* pada *CodeIgniter 3* memiliki kekurangan dimana pengguna harus membentuk secara  
 28 manual seluruh fungsi untuk mengambil, memasukan, dan memperbaharui data dari sebuah  
 29 tabel spesifik pada *model* tersebut. Kode 3.1 merupakan contoh fungsi yang digunakan untuk  
 30 mengambil data dari sebuah tabel.

Kode 3.1: Contoh fungsi untuk mengambil data seluruh user

```
31  
32 1 public function get_all_users()  
33 2 {  
34 3     return $this->db->order_by('role', 'asc')->order_by('id')->get('users')->result_array();  
35 4 }
```

37 Kode 3.1 mengambil data dari tabel `users` dengan hasil berupa *array* dan diurutkan sesuai  
 38 dengan *role* dan *idnya*.

## 2. Menggunakan *Model* pada *CodeIgniter 4*

*CodeIgniter 4* menyediakan fungsi *model* yang dapat dibentuk melalui *command line* untuk sebuah tabel spesifik. Pengguna dapat membentuk *model* melalui *command line* menggunakan sintaks sebagai berikut.

```
make:model <name>
```

*Model* pada *CodeIgniter 4* menyediakan fungsi untuk mengambil, memasukan, dan memperbaharui data dari sebuah tabel spesifik tanpa harus membentuk secara manual fungsi-fungsi tersebut. Pengguna dapat memakai fungsi untuk mengambil data menggunakan kode berikut.

```
$userModel = new \App\Models\UserModel();
```

```
$user = $userModel->findAll();
```

Kode diatas akan mengambil seluruh data dari *\$userModel* sesuai dengan konfigurasi yang telah dilakukan pengguna. Pengguna juga dapat melakukan *create*, *update*, dan *delete* melalui kode berikut.

### Kode 3.2: Contoh kode untuk menghapus data pada model

```
$this->Notifications_model->delete('notifications', array('id' => $id));
```

. Konversi aplikasi *SharIF Judge* akan menggunakan kedua buah cara dengan penghapusan beberapa fungsi yang terdapat pada *model* dari *CodeIgniter 4* seperti mengambil, menghapus, dan menambahkan data. Sedangkan untuk fungsi-fungsi lain pada *SharIF Judge* akan dilakukan pembaharuan sesuai dengan dokumentasi yang telah ada.

## View

*View* pada aplikasi *SharIF Judge* menggunakan *template engine* bernama *Twig*. *Twig* merupakan sebuah *template engine* untuk bahasa pemrograman *PHP* yang berguna untuk mempermudah dalam membentuk tampilan sebuah aplikasi. *Twig* tidak terintegrasi pada *CodeIgniter 4* sehingga akan terdapat beberapa pemasangan dan perubahan pada sintaks yang telah dipasang. Selain itu, terdapat beberapa perubahan fungsi pada *CodeIgniter 4* sehingga perlu dilakukan penyesuaian seperti pengubahan *file extension* dari *.twig* menjadi *.php*. Konversi *SharIF Judge* menuju *CodeIgniter 4* akan mengubah *view* yang sebelumnya menggunakan *twig* menjadi menggunakan *PHP* sesuai dengan dokumentasi *CodeIgniter 4*. Kode 3.3 merupakan contoh konversi yang dilakukan.

### Kode 3.3: Contoh *view* menggunakan *twig*

```
{% if registration_enabled %}
  <a href="{{ site_url('register') }}">Register</a> |
{% endif %}
```

menjadi kode berikut:

### Kode 3.4: Contoh *view* menggunakan *php*

```
<?php if($registration_enabled): ?>
  <a href="<? site_url('register') ?>">Register</a> |
<?php endif ?>
```

Seluruh sintaks *twig* akan diubah menjadi sintaks *PHP* dari *CodeIgniter 4* seperti kode 3.4.

## 1 *Controller*

2 *Controller* pada *CodeIgniter 4* memiliki fungsi sama dengan pada *CodeIgniter 3* sehingga hanya  
 3 perlu dilakukan penghapusan dan perubahan pada sintaks yang ada. Namun, terdapat perubahan  
 4 pada *constructor* dimana pada *CodeIgniter 4* terdapat *initController*. *Constructor* pada *PHP*  
 5 tidak diperbolehkan untuk mengembalikan apapun sehingga terdapat beberapa pemindahan fungsi  
 6 seperti *redirect()* menuju *filters*. Selain itu, konversi akan tetap menggunakan *\_\_construct*  
 7 dengan pemindahan beberapa sintaks menuju *initController* seperti pemanggilan *helpers* dan  
 8 variabel yang dapat diakses pada seluruh *controller*. Berikut merupakan contoh penggunaan  
 9 *initController* untuk variabel.

Kode 3.5: Contoh penggunaan *initController* untuk variabel

```
10 protected $config;
11
12
13 public function initController(RequestInterface $request, ResponseInterface $response, LoggerInterface $logger)
14 {
15     // Do Not Edit This Line
16     parent::initController($request, $response, $logger);
17
18     // Preload any models, libraries, etc, here.
19
20     // E.g.: $this->session = \Config\Services::session();
21 }
```

23 Kode 3.5 akan memanggil variabel *\$session* yang dapat digunakan seluruh *controller*.

24 Fungsi lainnya akan dipindahkan sesuai dengan yang ada pada *CodeIgniter 3* dengan pembaha-  
 25 ruan sesuai dengan dokumentasi *CodeIgniter 4*. Selain pembaharuan, akan terdapat pemindahan  
 26 variabel *global* yang sebelumnya telah diinisiasikan menuju *controller*.

### 27 3.1.7 *Libraries*

28 *Libraries* pada *CodeIgniter 3* memiliki perubahan dan penghapusan pada *CodeIgnier 4* sehingga  
 29 perlu dilakukan pembaharuan. Berikut merupakan *libraries* yang dipakai pada *SharIF Judge*:

## 30 *Emails*

31 *Emails* pada *CodeIgnier 4* terdapat perubahan sintaks dan cara pemanggilan sehingga akan dipin-  
 32 dahkan sesuai dengan sintaks yang baru. Sintaks berubah dari yang sebelumnya menggunakan  
 33 *snakecase* menjadi menggunakan *camelcase*. Kode ?? merupakan contoh penggunaan *library email*.

Kode 3.6: Contoh perubahan *library emails*

```
34
35 1 $this->email->setFrom($this->settings_model->get_setting('mail_from'), $this->settings_model->get_setting('mail_from_name'));
36 2     $this->email->setTo($user[1]);
37 3     $this->email->setSubject('SharIF Judge Username and Password');
38 4     $text = $this->settings_model->get_setting('add_user_mail');
39 5     $text = str_replace('{SITE_URL}', base_url(), $text);
40 6     $text = str_replace('{ROLE}', $user[4], $text);
41 7     $text = str_replace('{USERNAME}', $user[0], $text);
42 8     $text = str_replace('{PASSWORD}', htmlspecialchars($user[3]), $text);
43 9     $text = str_replace('{LOGIN_URL}', base_url(), $text);
44 0     $this->email->setMessage($text);
45 1     $this->email->send()
```

47 Kode 3.6 memiliki sintaks dengan nama sama namun terdapat perubahan menjadi *camelcase*.

## 1 *Working with Uploaded Files*

2 *Working with uploaded files* terdapat perubahan pada beberapa sintaks dan validasi terhadap *file*  
 3 yang telah diunggah. Konversi aplikasi *SharIF Judge* akan menggunakan fungsi ini dengan beberapa  
 4 perubahan sintaks sesuai dengan dokumentasinya.

## 5 *Validations*

6 *Validations* terdapat perubahan dan penghapusan beberapa fungsi. Berikut merupakan contoh  
 7 pembentukan aturan untuk mengumpulkan sebuah data pada *form*.

```
8 $validate->setRule('username', 'username', 'requiredmin_length[3]|max_length[20]);|
```

9 Sintaks diatas akan melakukan validasi terhadap *input* yang akan masukan oleh pengguna. Namun,  
 10 *CodeIgniter 4* tidak menyediakan fungsi *form\_error* sehingga akan diubah dengan menggunakan  
 11 fungsi baru bernama *validation\_errors()*. Fungsi tersebut dapat digunakan untuk mengembalik-  
 12 an *error* apabila terdapat data yang tidak sesuai dengan aturan. *Error* tersebut dapat ditampilkan  
 13 pada halaman menggunakan sintaks berikut.

```
14 <?php $error = $validation->getError('username'); ?>
```

15 Selain menggunakan *validation\_errors()* akan digunakan sebuah fungsi *flashMessage* pada  
 16 *session*. Fungsi ini juga dapat menampilkan *error* dari data yang dimasukan pengguna apabila  
 17 tidak sesuai dengan *database*.

18 *Library* yang terdapat pada *CodeIgniter 4* juga dapat di*extend* dan dibentuk sesuai dengan  
 19 kebutuhan. Berikut merupakan *library* yang dibentuk oleh pengguna. Berikut merupakan *library*  
 20 yang dibentuk oleh pengguna.

## 21 *Password\_hash*

22 *Library* ini dibentuk oleh *phpass* untuk melakukan enkripsi *password* dan melakukan verifikasi *pass-*  
 23 *word*. *Phpass* dapat melakukan enkripsi dengan beberapa metode antara lain *CRYPT\_BLOWFISH*  
 24 dan *CRYPT\_EXT\_DES*. Namun, *phpass* hanya mendukung *PHP* versi 5 sampai dengan 7. Sehingga,  
 25 akan dilakukan konversi menggunakan fungsi yang disediakan oleh *PHP* bernama *password\_hash()*.  
 26 Seluruh penggunaan *library* ini akan diubah menggunakan fungsi yang disediakan oleh *PHP* dengan  
 27 metode *hashing* sama yaitu *CRYPT\_BLOWFISH*. Perubahan fungsi *hashing* ini bersifat *backward*  
 28 *compatible* sehingga dapat menggunakan *database* aplikasi terdahulu tanpa perlu membentuk data  
 29 baru. Berikut merupakan contoh pengubahan kode dari *phpass* menjadi *password\_hash*.

```
30 'password' => $this->password_hash->HashPassword($password)
```

31 menjadi

```
32 'password' => password_hash($password,PASSWORD_BCRYPT)
```

33 Sintaks diatas menggunakan *PASSWORD\_BCRYPT* karena algoritma tersebut memakai *CRYPT\_BLOWFISH*  
 34 untuk melakukan enkripsi.



### 3.1.8 *Configuration*

*Configuration* terdapat perubahan nama dari yang sebelumnya `application/config/config.php` menjadi `app/Config/App.php` dan penambahan *file* dengan nama `app/Config/Secrets.php`. Berhubung dengan perubahan nama tersebut, terdapat beberapa perpindahan beberapa sintaks menuju direktori baru tersebut. Berikut merupakan sintaks yang dipindahkan menuju `app/Config/Security.php`:

```

• $config['csrf_protection'] = TRUE;
• $config['csrf_token_name'] = 'shj_csrf_token';
• $config['csrf_cookie_name'] = 'shjcsrftoken';
• $config['csrf_expire'] = 7200;
• $config['csrf_regenerate'] = FALSE;
```

*Configurations* yang telah dipindahkan akan diubah dari yang sebelumnya menggunakan *array* menjadi menggunakan *variable*. Seluruh *configurations* pada *CodeIgniter 3* akan dipindahkan menuju *CodeIgniter 4* sesuai dengan direktorinya dan fungsinya.

### 3.1.9 *Database*

*Database* pada *CodeIgniter 4* fungsi yang sama pada *CodeIgniter 3* sehingga akan dilakukan pemindahan konfigurasi sesuai dengan yang ada pada *CodeIgniter 3*. Namun, terjadi beberapa perubahan pada sintaks untuk melakukan koneksi ke *database* dan beberapa sintaks untuk melakukan *query*. Sintaks koneksi *database* akan berubah dari `$this->load->database()`; menjadi `db = db_connect()`. Selain itu, pemanggilan fungsi *query builder* berubah menggunakan *camelcase* dari yang sebelumnya menggunakan *snakecase*. Seperti dari yang sebelumnya menggunakan sintaks `get_where` menjadi `getWhere`.

*Database* pada aplikasi *SharIF Judge* menggunakan *autoload* yang dapat memuat beberapa fungsi secara otomatis. Berikut merupakan contoh penggunaan *autoload* pada *CodeIgniter 3*.

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

Sintaks diatas memuat *library database* dan akan ditambahkan pada *file* `autoload.php`. Namun, pada konversi ini tidak akan menggunakan *autoload* karena *CodeIgniter 3* tidak mengikuti standar *PSR 4* sehingga pada *CodeIgniter 4* akan dimuat menggunakan `db = db_connect()` pada `initController`.

### 3.1.10 *Helpers*

*Helpers* tidak terdapat banyak perubahan namun, terdapat perubahan pemanggilan *helpers* dan beberapa penghapusan *helpers*. Berikut merupakan *helpers* yang dihapus dan diubah cara pemanggilannya.

- *Download Helper* *Helper* ini sudah tidak tersedia pada *CodeIgniter 4* sehingga perlu dilakukan pengubahan dengan menggunakan fungsi *HTTP Response*. *HTTP Response* menyediakan fungsi bernama *Force File Download* yang berguna untuk mengunduh sebuah *file* menuju perangkat pengguna. Fungsi ini dapat dipanggil menggunakan sintaks berikut.

```
return $this->response->download($name, $data);
```

Sintaks diatas mengembalikan *file* yang ingin diunduh pengguna dengan dua buah parameter. Parameter pertama berupa nama *file* yang ingin diunduh sedangkan parameter kedua merupakan data dalam *file* tersebut.

- **redirect()**

Fungsi ini memiliki perubahan pada *CodeIgniter 4* dimana **redirect()** tidak langsung mengarahkan kepada *url* yang dibentuk. Pengguna harus mengembalikan fungsi *redirect()* menggunakan **return** dengan sintaks sebagai berikut.

```
return redirect()->to('login/form')
```

Sintaks diatas akan mengembalikan pengguna menuju *url login/form* yang sudah dibentuk pada *routes*.

Konversi *helpers* akan dipindahkan dari yang sebelumnya menggunakan *autoload* menuju *initController* dengan menambahkan pada variabel *helpers*.

## 3.2 Analisis Sistem Kini

Seperti dibahas pada bab 2.2, *SharIF Judge* merupakan sebuah *online judge* dan di kustomisasi sesuai dengan kebutuhan Informatika UNPAR. *SharIF Judge* dibentuk menggunakan *framework CodeIgniter 3* yang menerapkan arsitektur *Model-View-Controller* atau MVC. Arsitektur ini memisahkan pemrosesan data pada *Model*, memisahkan logika pada *Controller*, dan memisahkan tampilan pada *View*.

### 3.2.1 Model

*Model* terdapat pada direktori *application/models*. Direktori ini berisikan kelas *model* dengan fungsi-fungsi untuk mengolah data pada aplikasi. Berikut merupakan *model* pada *SharIF Judge* beserta fungsi-fungsinya.

#### Assignment\_model.php

*Model Assignment* terdapat beberapa fungsi untuk memproses data pada tabel *assignment*. Berikut merupakan fungsi-fungsi dari *model* tersebut:

- **add\_assignment**

Fungsi ini berguna untuk menambahkan atau memperbaharui *assignment* pada *database*.

- **delete\_assignment**

Fungsi ini berguna untuk menghapus *assignment* pada *database*.

- **all\_assignments**

Fungsi ini berguna untuk mengembalikan seluruh *assignment* beserta informasi *assignment* tersebut.

- **new\_assignment\_id**

Fungsi ini berguna untuk mencari *id* terkecil yang dapat digunakan untuk menambahkan *assignment* baru.

- **all\_problems**

Fungsi ini berguna untuk mengembalikan seluruh *problems* dari *assignment* yang ada.

- **problem\_info**

Fungsi ini berguna untuk mengembalikan baris tabel untuk *problem* tertentu.

- **assignment\_info**

Fungsi ini berguna untuk mengembalikan baris tabel untuk **assignment** tertentu.

- **is\_participant**

Fungsi ini berguna untuk mengecek apakah *username* merupakan peserta atau tidak.

- **increase\_total\_submits**

Fungsi ini berguna untuk menambahkan satu buah total *submit*.

- **set\_moss\_time**

Fungsi ini berguna untuk memperbaharui "*Moss Update Time*" untuk *assignment* tertentu.

- **get\_moss\_time**

Fungsi ini berguna untuk mengembalikan "*Moss Update Time*" untuk *assignment* tertentu.

- **save\_problem\_description**

Fungsi ini berguna untuk menyimpan atau memperbaharui deskripsi *problem*.

- **\_update\_coefficients**

Fungsi ini dipanggil pada fungsi **add\_assignment** yang berguna untuk memperbaharui koefisien dari *assignment* tertentu.

#### **Hof\_model.php**

Berikut merupakan fungsi-fungsi dari **Hof\_model.php** yang berguna untuk mengambil data untuk ditampilkan pada halaman *Hall of Fame*.

- **get\_all\_final\_submission**

Fungsi ini berguna untuk mengambil data untuk *Hall of Fame* berdasarkan *username*.

- **get\_all\_user\_assignments**

Fungsi ini berguna untuk mengambil detail *assignment* dan *problem* berdasarkan pengguna.

#### **Logs\_model.php**

Berikut merupakan fungsi-fungsi dari **Logs\_model.php** yang berguna untuk mencatat *log* pada beberapa tabel.

- **insert\_to\_logs**

Fungsi ini berguna untuk mencatat *log* pada tabel *login*.

- **get\_all\_logs**

Fungsi ini berguna untuk mengembalikan seluruh *log* dalam bentuk *array*.

#### **Notifications\_model.php**

*Notifications Assignment* terdapat beberapa fungsi untuk memproses data pada tabel *notifications*.

Berikut merupakan fungsi-fungsi dari *model* tersebut:

- **get\_all\_notifications**

Fungsi ini berguna untuk mengembalikan seluruh *notifications* dalam bentuk *array*.

- **get\_latest\_notifications**

Fungsi ini berguna untuk mengembalikan sepuluh *notification* terakhir.

- 1 • `add_notification`  
2 Fungsi ini berguna untuk menambahkan *notification* baru.
- 3 • `update_notification`  
4 Fungsi ini berguna untuk memperbaharui *notification* tertentu.
- 5 • `delete_notification`  
6 Fungsi ini berguna untuk menghapus *notification* tertentu.
- 7 • `get_notification`  
8 Fungsi ini berguna untuk mengembalikan *notification* dalam bentuk *array*.
- 9 • `have_new_notification`  
10 Fungsi ini berguna untuk mengecek apakah terdapat *notification* setelah waktu tertentu.

#### 11 `Queue_model.php`

12 Berikut merupakan fungsi-fungsi dari `Queue_model.php` yang berguna untuk memproses data pada  
13 halaman *queue*.

- 14 • `in_queue`  
15 Fungsi ini berguna untuk mengecek apakah *submission* pengguna tertentu sudah dalam  
16 antrean.
- 17 • `get_queue`  
18 Fungsi ini berguna untuk mengembalikan data seluruh antrian.
- 19 • `empty_queue`  
20 Fungsi ini berguna untuk menghapus seluruh tabel *queue*.
- 21 • `add_to_queue`  
22 Fungsi ini berguna untuk memasukan *submission* kedalam tabel *queue*.
- 23 • `rejudge`  
24 Fungsi ini berguna untuk menambahkan *submission* kedalam antrean untuk dilakukan *rejudge*.
- 25 • `rejudge_single`  
26 Fungsi ini berguna untuk menambahkan satu buah *submission* kedalam antrean untuk dila-  
27 kukan *rejudge*.
- 28 • `get_first_item`  
29 Fungsi ini berguna untuk mengambil data pertama dalam antrean.
- 30 • `remove_item`  
31 Fungsi ini berguna untuk menghapus data tertentu dalam antrean.
- 32 • `save_judge_result_in_db`  
33 Fungsi ini berguna untuk menyimpan hasil dari *judge* ke dalam *database*.
- 34 • `add_to_queue_exec`  
35 Fungsi ini berguna untuk menambahkan data *dummy* pada antrean.

#### 36 `Scoreboard_model.php`

37 Berikut merupakan fungsi-fungsi dari `Scoreboard_model.php` yang berguna untuk memproses data  
38 untuk ditampilkan pada halaman *Score Board*.

- 39 • `_generate_scoreboard`  
40 Fungsi ini dipanggil pada fungsi `update_scoreboard` dan berfungsi untuk menghasilkan

*scoreboard* dari *final submission*.

- **update\_scoreboards**

Fungsi ini berguna untuk memperbaharui *cache scoreboard* dari seluruh *assignment*. Fungsi ini dipanggil setiap terdapat penghapusan pengguna atau seluruh *assignment* pengguna.

- **update\_scoreboard**

Fungsi ini berguna untuk memperbaharui *cache scoreboard* dari seluruh *assignment*. Fungsi ini dipanggil setelah *judge* atau *rejudge*.

- **get\_scoreboard**

Fungsi ini berguna untuk mengambil seluruh *cache scoreboard* dari *assignment* tertentu.

#### **Settings\_model.php**

Berikut merupakan fungsi-fungsi dari **Settings\_model.php** yang berguna untuk memproses data untuk ditampilkan pada tabel *settings*.

- **get\_setting**

Fungsi ini berguna untuk mengembalikan data *setting* tertentu.

- **set\_setting**

Fungsi ini berguna untuk memperbaharui *setting*.

- **get\_all\_settings**

Fungsi ini berguna untuk mengembalikan seluruh data *setting*.

- **set\_settings**

Fungsi memperbaharui beberapa *setting*.

#### **Submit\_model.php**

Berikut merupakan fungsi-fungsi dari **Submit\_model.php** yang berguna untuk memproses data yang berkaitan dengan *submission*.

- **get\_submission**

Fungsi ini berguna untuk mengembalikan baris data *submission* tertentu.

- **get\_final\_submissions**

Fungsi ini berguna untuk mengembalikan seluruh *final submission*.

- **get\_all\_submissions**

Fungsi ini berguna untuk mengembalikan seluruh *submission*.

- **count\_final\_submissions**

Fungsi ini berguna untuk menghitung seluruh *final submission*.

- **count\_all\_submissions**

Fungsi ini berguna untuk menghitung seluruh *submission*.

- **set\_final\_submission**

Fungsi ini berguna untuk memperbaharui *submission* tertentu menjadi *final submission*.

- **add\_upload\_only**

Fungsi ini berguna untuk menambahkan hasil dari *upload only* ke dalam *database*.

### 1 *User\_model.php*

2 Berikut merupakan fungsi-fungsi dari *User\_model.php* yang berguna untuk memproses data pada  
3 tabel *users*.

4 • **have\_user**

5 Fungsi ini berguna untuk mengecek apakah terdapat pengguna dengan *username* tertentu.

6 • **user\_id\_to\_username**

7 Fungsi ini berguna untuk mengembalikan *user id* menjadi *username*.

8 • **username\_to\_user\_id**

9 Fungsi ini berguna untuk mengembalikan *username* menjadi *user id*.

10 • **have\_email**

11 Fungsi ini berguna untuk mengecek apakah terdapat *username* dengan *email* tertentu.

12 • **add\_user**

13 Fungsi ini berguna untuk menambahkan sebuah pengguna.

14 • **add\_users**

15 Fungsi ini berguna untuk menambahkan banyak pengguna.

16 • **delete\_user**

17 Fungsi ini berguna untuk menghapus pengguna tertentu.

18 • **delete\_submissions**

19 Fungsi ini berguna untuk menghapus seluruh *submission* pada pengguna tertentu.

20 • **validate\_user**

21 Fungsi ini berguna untuk mengecek *username* dan *password* apakah sesuai dengan data.

22 • **selected\_assignment**

23 Fungsi ini berguna untuk mengembalikan *assignment* untuk pengguna tertentu.

24 • **get\_names**

25 Fungsi ini berguna untuk mengembalikan nama dari pengguna tertentu.

26 • **update\_profile**

27 Fungsi ini berguna untuk memperbaharui profil dari pengguna seperti nama, *email*, *password*,  
28 dan *role*.

29 • **send\_password\_reset\_mail**

30 Fungsi ini berguna untuk menghasilkan *password reset key* dan mengirim *email* untuk mela-  
31 kukan *reset password*.

32 • **passchange\_is\_valid**

33 Fungsi ini berguna untuk mengecek apakah *password key* yang diberikan sesuai atau tidak.

34 • **reset\_password**

35 Fungsi ini berguna untuk mengatur ulang *password* sesuai dengan *password key* tertentu.

36 • **get\_all\_users**

37 Fungsi ini berguna untuk mengembalikan seluruh pengguna untuk halaman *users*.

38 • **get\_user**

39 Fungsi ini berguna untuk mengembalikan baris data untuk pengguna tertentu.

40 • **update\_login\_time**

41 Fungsi ini berguna untuk memperbaharui *login time* dan *last login time* untuk pengguna  
42 tertentu.

### 1 *User.php*

2 Berikut merupakan fungsi-fungsi dari *User.php* yang berguna untuk memproses data pada tabel  
3 *users*.

- 4 • *select\_assignment*

5 Fungsi ini berguna untuk mengatur *assignment* yang dipilih oleh pengguna.

- 6 • *save\_widget\_positions*

7 Fungsi ini berguna untuk memperbaharui posisi dari *dashboard widget* pada *database*.

- 8 • *get\_widget\_positions*

9 Fungsi ini berguna untuk mengembalikan data *dashboard widget*.

### 10 3.2.2 *View*

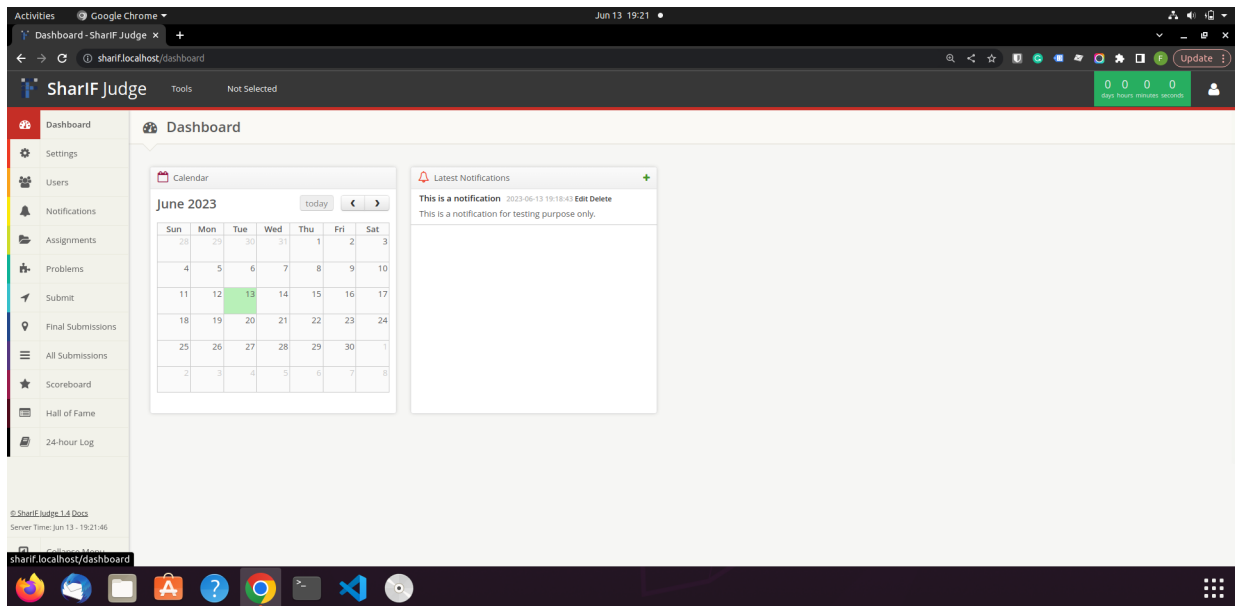
### 11 3.2.3 Halaman pada *SharIF Judge*

12 Halaman pada *SharIF Judge* dibagi berdasarkan *role* penggunaanya. Setiap *role* memiliki tampilan  
13 halaman berbeda dengan fitur-fitur berbeda setiap *role* penggunaanya. Berikut merupakan halaman  
14 dan fitur pada *SharIF Judge*:

### 15 3.2.4 *View*

16 *View* terdapat pada direktori *application/views*. Direktori ini berisikan seluruh *file* untuk tam-  
17 pilan halaman *SharIF Judge*. *File* tersebut dipisahkan oleh direktori sesuai dengan fungsinya.  
18 Direktori tersebut dibagi menjadi tiga buah direktori utama yakni *error*, *pages*, dan *templates*.  
19 Direktori *error* berisikan tampilan halaman *error* yang akan dilihat oleh pengguna. Direktori *pages*  
20 merupakan tampilan utama *SharIF Judge* yang terbagi lagi menjadi dua buah direktori yakni *admin*  
21 dan *authentication*. Direktori *admin* berisikan tampilan halaman untuk *role admin*. Direktori  
22 *authentication* berisikan tampilan halaman untuk akses pengguna seperti *LOgin*, *Register*, dan *Reset*  
23 *Password*. Direktori *templates* terdiri dari tampilan yang digunakan oleh seluruh tampilan halaman  
24 seperti *header* dan *side bar*. Berikut merupakan tampilan halaman pada *SharIF Judge*:

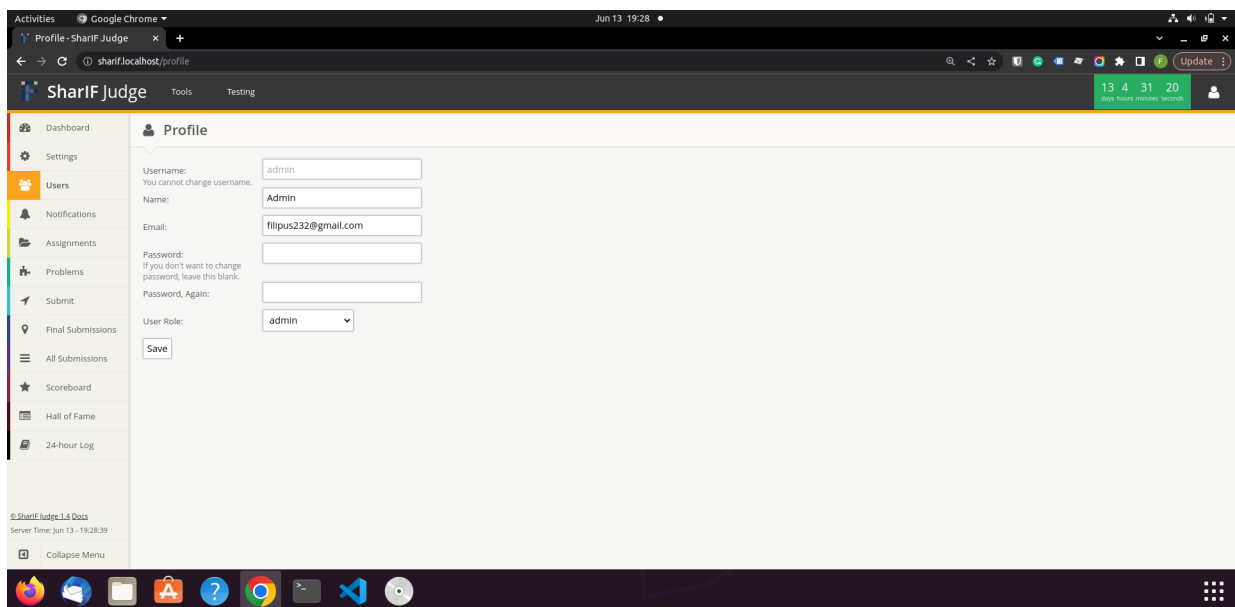
## 1 *Dashboard*



Gambar 3.2: Tampilan Halaman Dashboard

2 Gambar 3.2 merupakan tampilan halaman *dashboard* yang terdapat pada semua *role* pengguna.

## 3 *Profile*

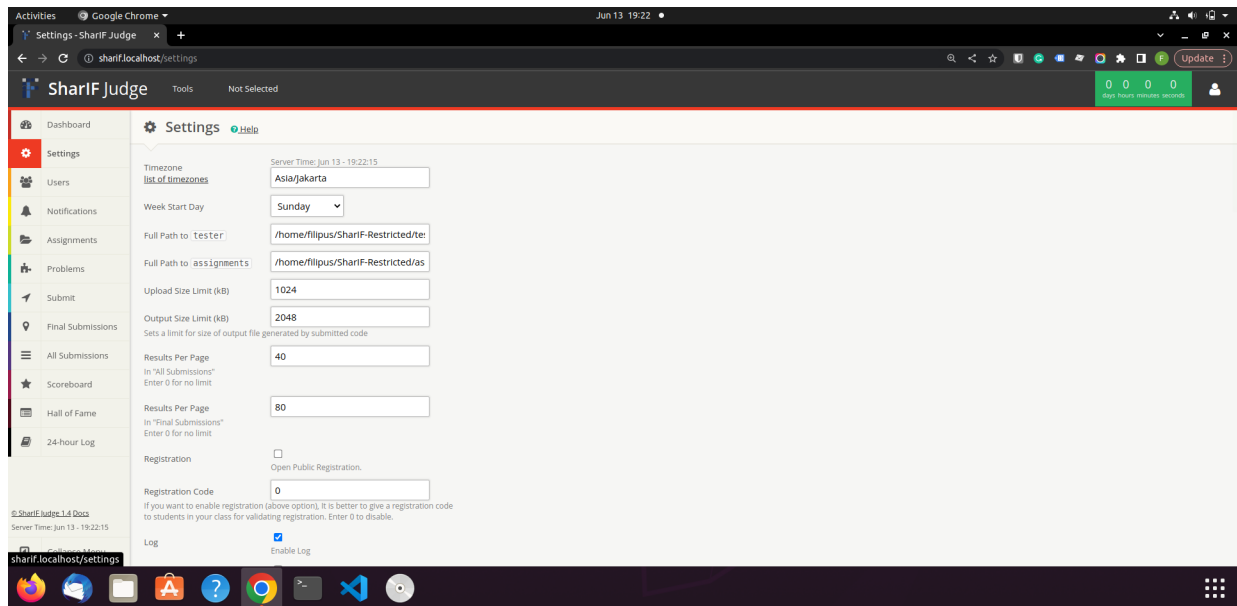


Gambar 3.3: Tampilan Halaman Profile

4 Gambar 3.3 merupakan tampilan halaman *profile* yang terdapat pada semua *role* pengguna. Namun,  
5 terdapat fitur yang tidak dapat digunakan oleh *siswa* dan *instructor* yakni mengganti role.



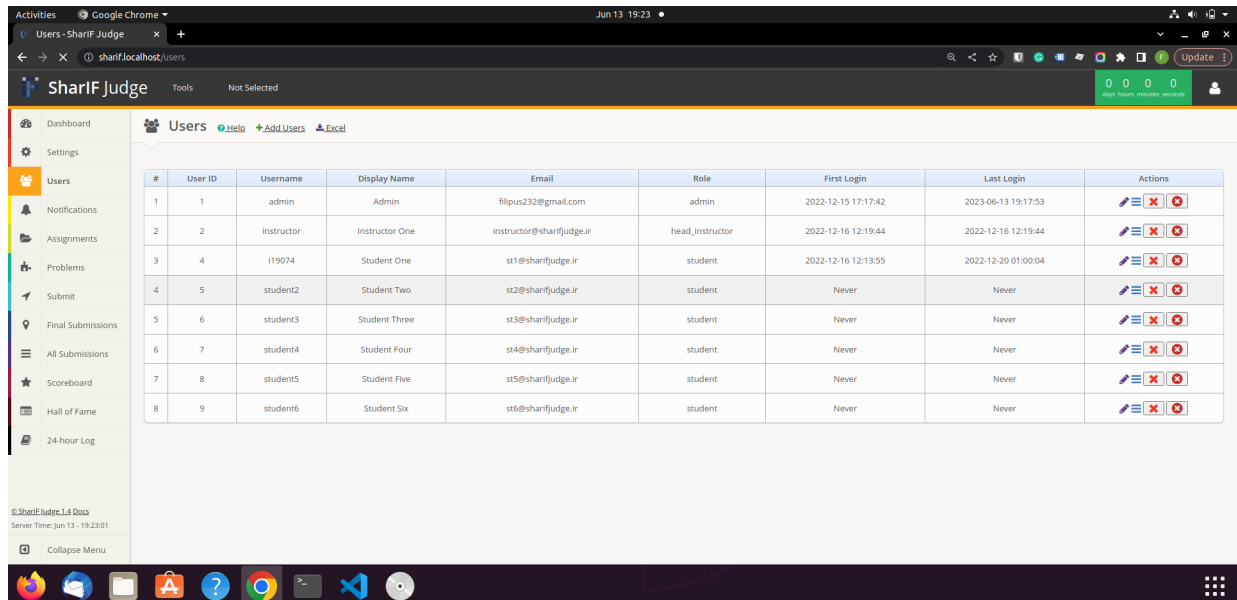
## 1 Settings



Gambar 3.4: Tampilan Halaman Settings

2 Gambar 3.4 merupakan tampilan halaman *settings* yang terdapat hanya pada *role* admin dan *head*  
3 *instructor*.

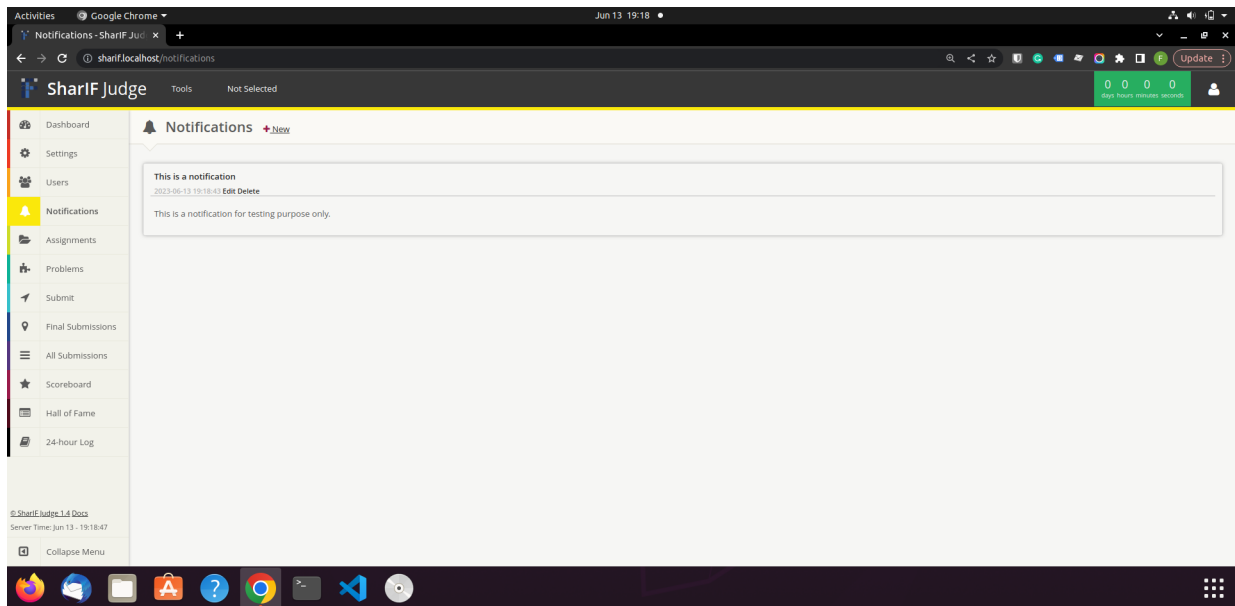
## 4 Users



Gambar 3.5: Tampilan Halaman Users

5 Gambar 3.5 merupakan tampilan halaman *users* yang terdapat hanya pada *role* admin dan *head*  
6 *instructor*.

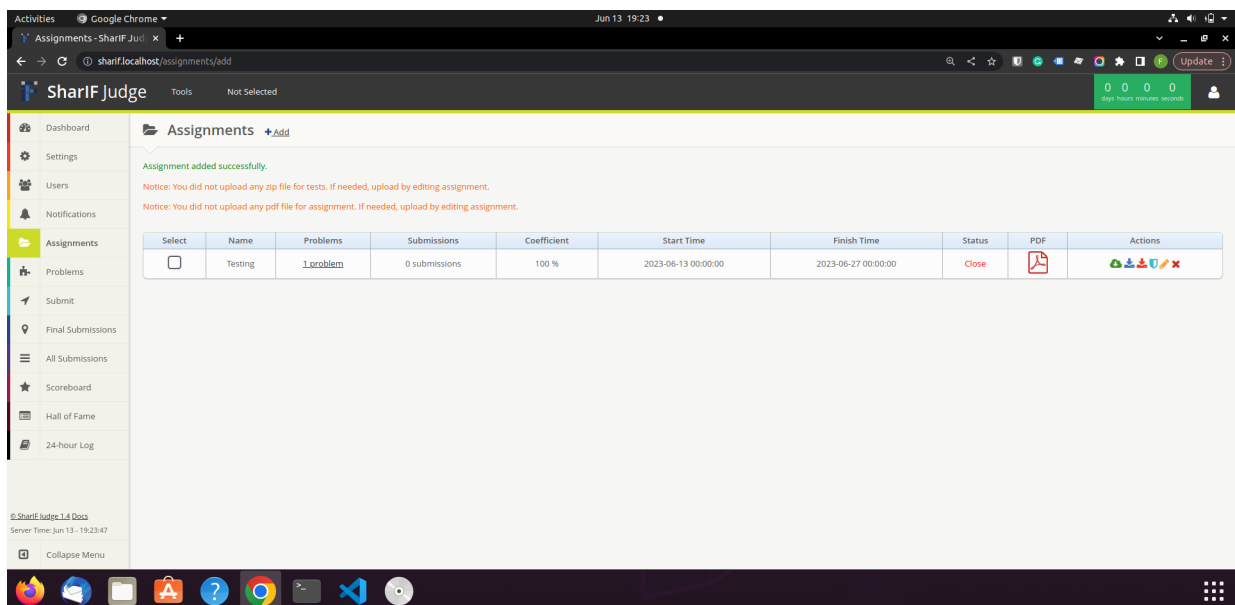
## 1 *Notifications*



Gambar 3.6: Tampilan Halaman Notifications

2 Gambar 3.6 merupakan tampilan halaman *notifications* yang terdapat pada semua *role* pengguna.

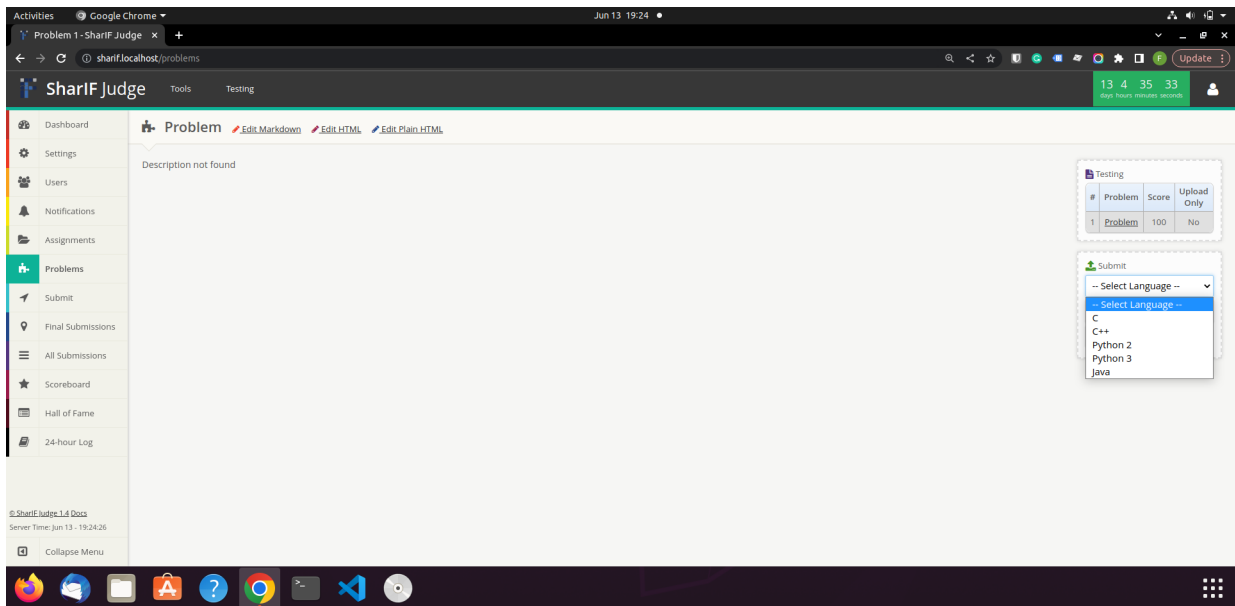
## 3 *Assignments*



Gambar 3.7: Tampilan Halaman Assignments

4 Gambar 3.7 merupakan tampilan halaman *assignments* yang terdapat pada semua *role* pengguna.  
 5 Namun, terdapat bagian yang tidak dapat diakses oleh *role* siswa dan *instructor* yakni bagian  
 6 *actions*.

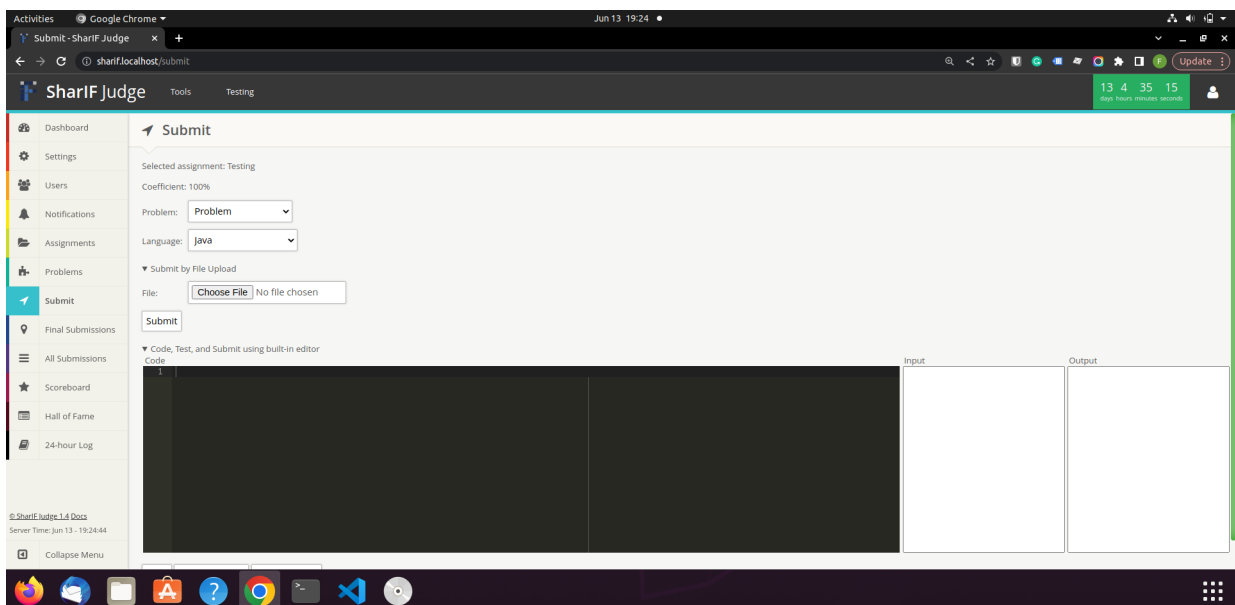
## 1 *Problems*



Gambar 3.8: Tampilan Halaman Problems

2 Gambar 3.8 merupakan tampilan halaman *problems* yang terdapat pada semua *role* pengguna.

## 3 *Submit*



Gambar 3.9: Tampilan Halaman Submit

4 Gambar 3.9 merupakan tampilan halaman *submit* yang terdapat pada semua *role* pengguna.

## 1 *Final Submissions*

Final Submissions of Testing

#	ID	Username	Name	Problem	Submit Time	Score			Language	Status	Code	Log	Actions
						Score	Delay %	Final Score					
1	1	admin	Admin	1	2023-06-13 19:26:24	0	No Delay 100%	0	java	Compilation Error	Code	Log	

Gambar 3.10: Tampilan Halaman Final Submission

2 Gambar 3.10 merupakan tampilan halaman *submit* yang terdapat pada semua *role* pengguna.

## 3 *All Submissions*

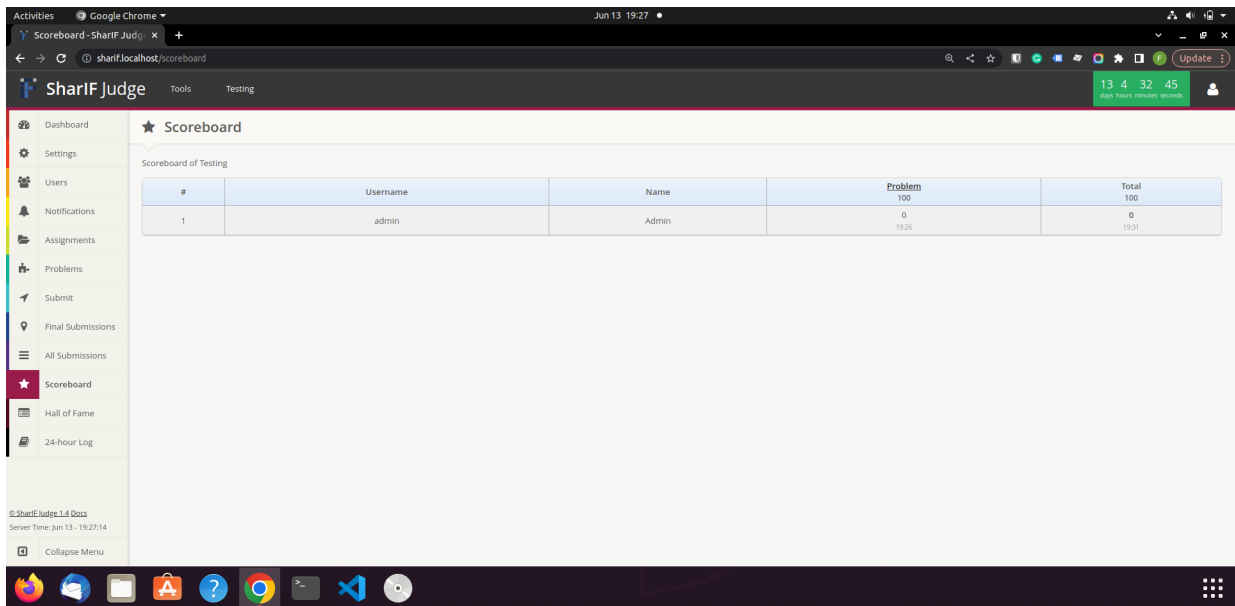
All Submissions of Testing

⚠ You cannot change your final submissions after assignment finishes.

Final	ID	Username	Name	Problem	Submit Time	Score			Language	Status	Code	Log	Actions
						Score	Delay %	Final Score					
✓	2	admin	Admin	1	2023-06-13 19:28:03	0	No Delay 100%	0	Py 3	Syntax Error	Code	Log	
○	1	admin	Admin	1	2023-06-13 19:26:24	0	No Delay 100%	0	java	Compilation Error	Code	Log	

Gambar 3.11: Tampilan Halaman All Submission

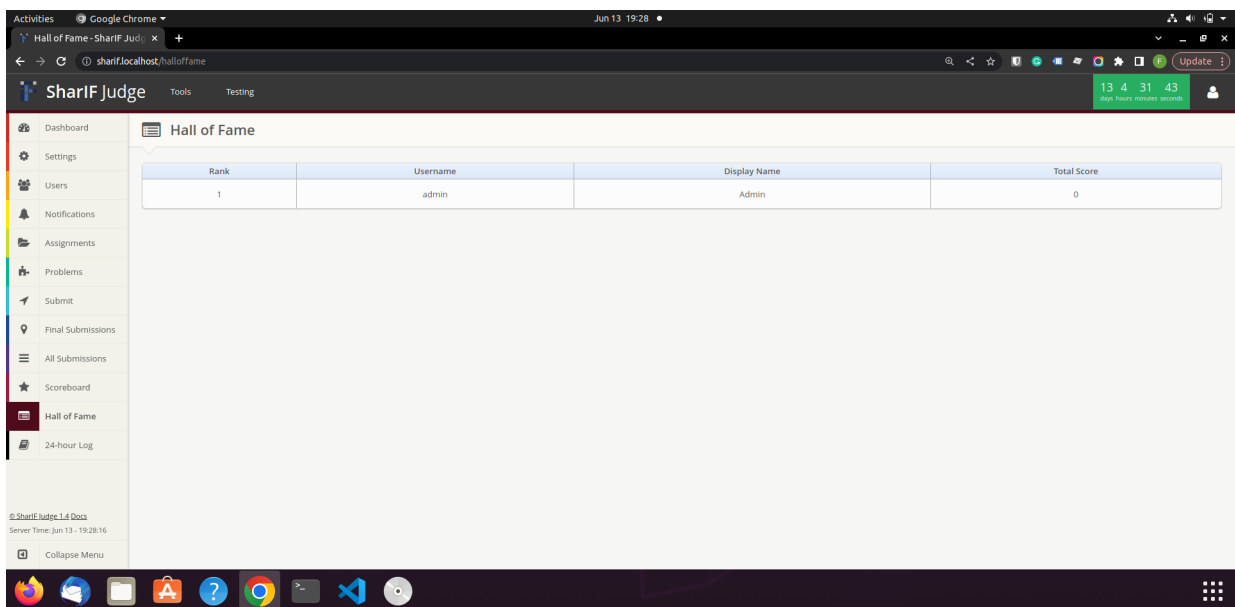
4 Gambar 3.11 merupakan tampilan halaman *All Submission* yang terdapat pada semua *role* pengguna.

1 *Scoreboard*

#	Username	Name	Problem	Total
1	admin	Admin	0	0

Gambar 3.12: Tampilan Halaman Scoreboard

2 Gambar 3.11 merupakan tampilan halaman *All Submission* yang terdapat pada semua *role* pengguna.

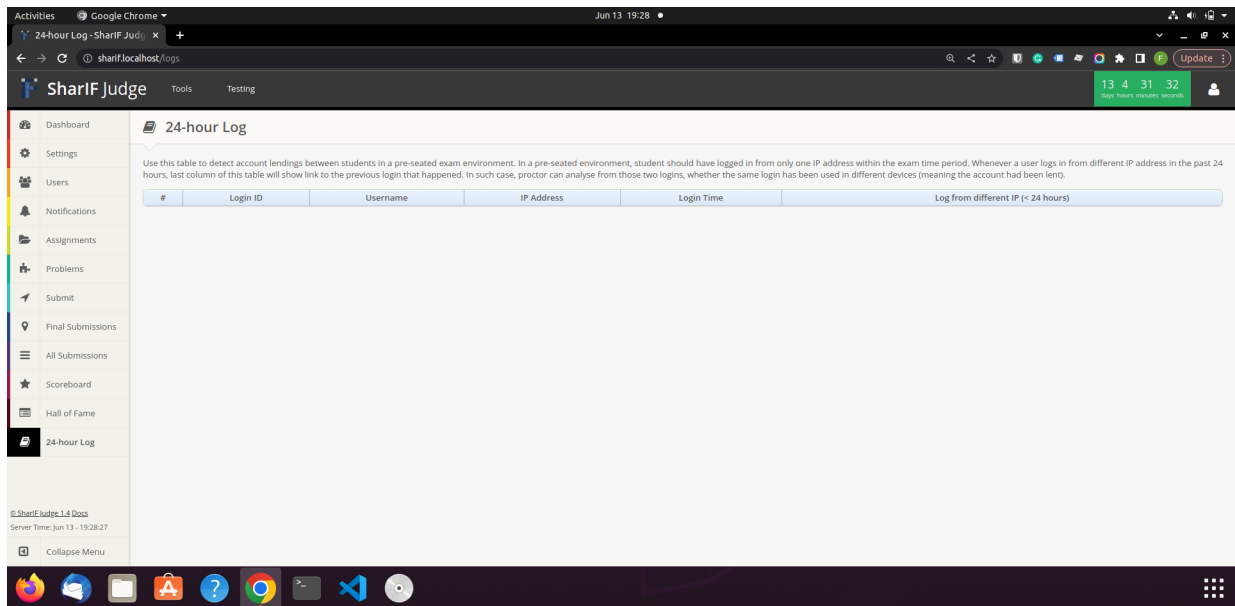
3 *Hall of Fame*

Rank	Username	Display Name	Total Score
1	admin	Admin	0

Gambar 3.13: Tampilan Halaman Hall of Fame

4 Gambar 3.13 merupakan tampilan halaman *Hall of Fame* yang terdapat pada semua *role* pengguna.

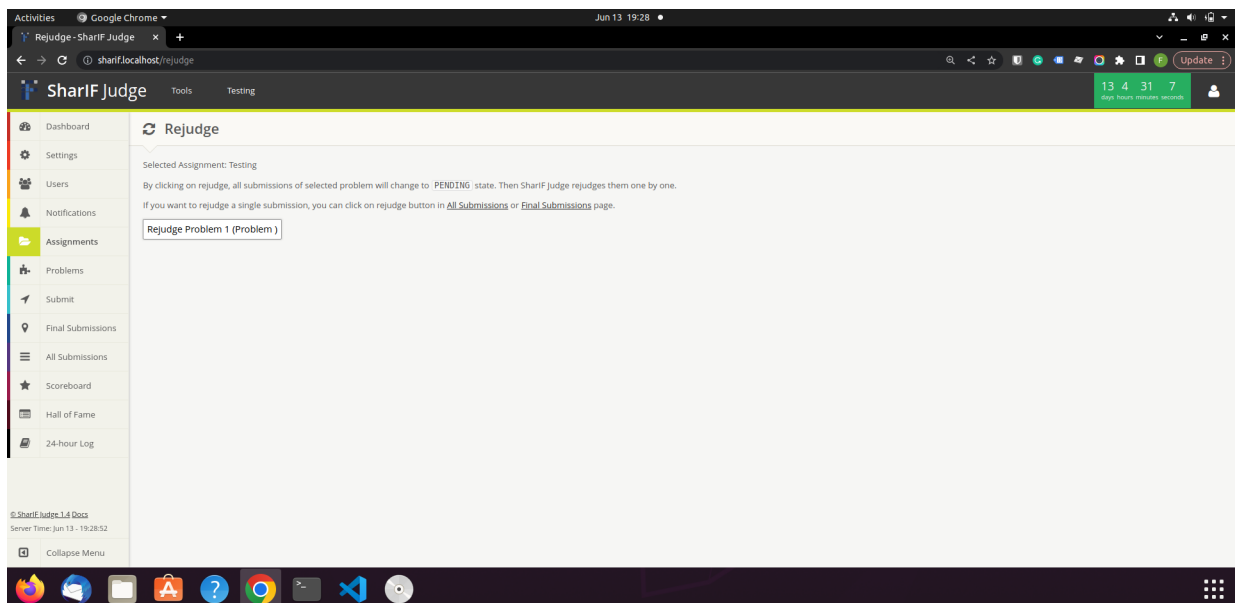
## 1 *24-hour Log*



Gambar 3.14: Tampilan Halaman 24-hour Log

2 Gambar 3.14 merupakan tampilan halaman *24-hour Log* yang terdapat hanya pada *role admin* dan  
 3 *head instructor*.

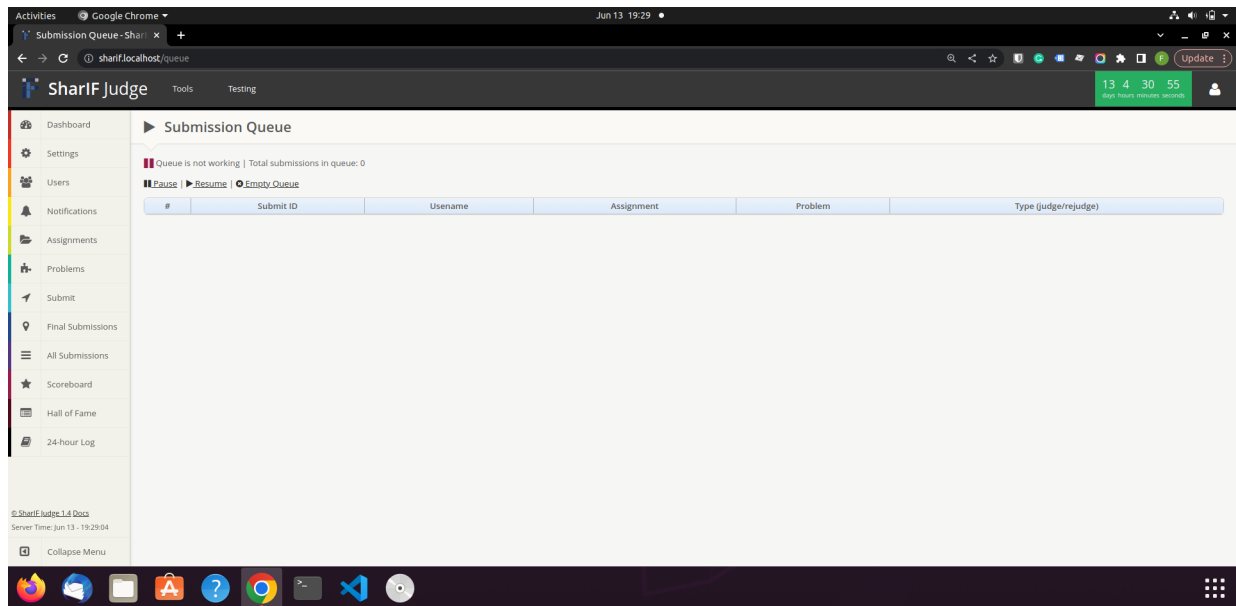
## 4 *Rejudge*



Gambar 3.15: Tampilan Halaman ReJudge

5 Gambar 3.15 merupakan tampilan halaman *ReJudge* yang terdapat hanya pada *role admin* dan  
 6 *head instructor*.

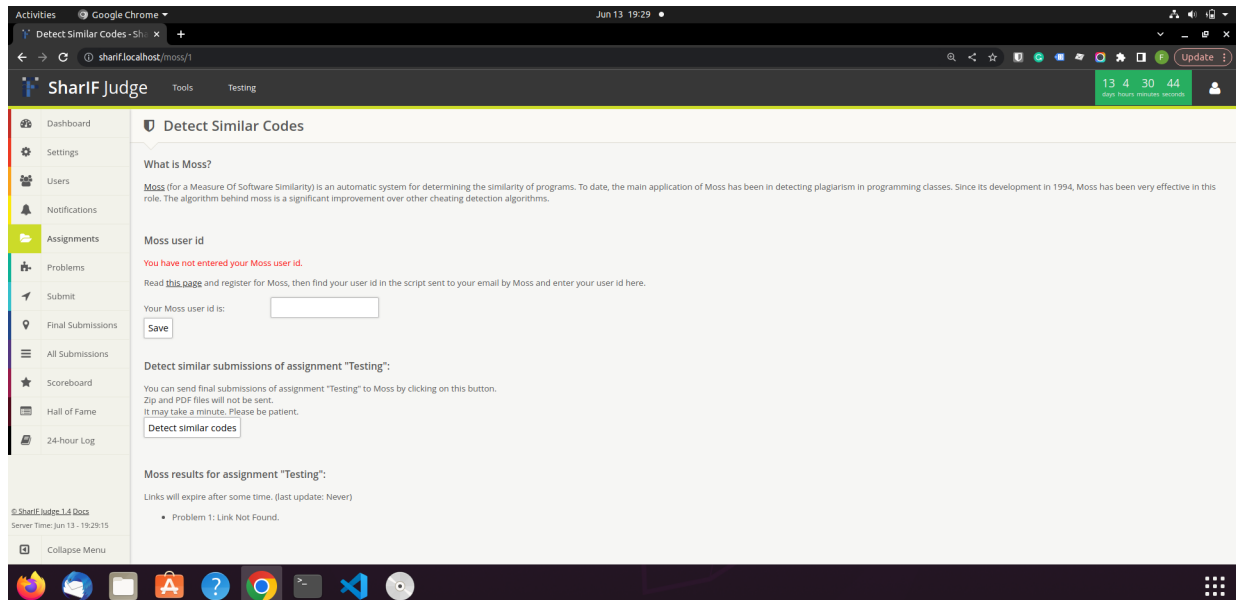
### 1 *Submission Queue*



Gambar 3.16: Tampilan Halaman Submission Queue

- 2 Gambar 3.16 merupakan tampilan halaman *Submission Queue* yang terdapat hanya pada *role admin*  
 3 dan *head instructor*.

### 4 *Cheat Detection*



Gambar 3.17: Tampilan Halaman Cheat Detection

- 5 Gambar 3.17 merupakan tampilan halaman *Cheat Detection* yang terdapat hanya pada *role admin*  
 6 dan *head instructor*.

### 3.2.5 *Controller*

*Controller* pada direktori `application/controller`. Direktori ini berisikan kelas *controller* dengan fungsi-fungsi dalam mengambil atau memberikan data *models* untuk dialihkan menuju *views* untuk ditampilkan. Berikut merupakan *controller* pada *SharIF Judge* beserta fungsi-fungsinya.

#### `Assignments.php`

Berikut merupakan fungsi-fungsi pada *controller* `Assignments.php`.

- `index`

Fungsi ini berguna untuk mengambil dan memberikan data menuju halaman `assignments.twig` menggunakan `Assignment_model`.

- `select`

Fungsi ini berguna untuk memilih *assignment* menggunakan *ajax*.

- `pdf`

Fungsi ini berguna untuk mengunduh *assignment* atau *problem* dalam bentuk pdf.

- `downloadtestsdesc`

Fungsi ini berguna untuk mengunduh dan mengompres data *test* dan deskripsi sebuah *assignment*.

- `download_submissions`

Fungsi ini berguna untuk mengunduh dan mengompres kode terakhir sebuah *assignment* pengguna.

- `delete`

Fungsi ini berguna untuk menghapus *assignment*.

- `add`

Fungsi ini berguna untuk menambah atau mengubah *assignment* berdasarkan masukan pengguna.

- `_add`

Fungsi ini berguna untuk menambah atau mengubah *assignment*.

- `edit`

Fungsi ini berguna untuk mengecek *role* pengguna dapat mengubah *assignment*. Selanjutnya akan dikembalikan pada fungsi `add`.

- `pdfCheck` Fungsi ini berguna untuk mengecek *file* pdf dari sebuah *assignment*.

#### `Dashboard.php`

Berikut merupakan fungsi-fungsi pada *controller* `Dashboard.php`.

- `index`

Fungsi ini berguna untuk mengambil dan memberikan data menuju halaman *dashboard* menggunakan tiga buah *model*. *Model* tersebut terdiri dari `Assignment_model`, `Settings_model`, dan `Notifications_model`.

- `widget_positions`

Fungsi ini berguna untuk menyimpan data *widget* pengguna.



### 1 **Install.php**

2 *Controller Install.php* hanya memiliki satu buah fungsi bernama **index**. Fungsi ini berguna untuk  
3 membentuk tabel yang dibutuhkan oleh *SharIF Judge* pada *database*. Selain itu, fungsi ini juga  
4 berguna untuk memasukan data pengguna *admin* yang pertama kali memasang *SharIF Judge* pada  
5 perangkat.

### 6 **Login.php**

7 Berikut merupakan fungsi-fungsi pada *controller Login.php*.

- 8 • **\_registration\_code**

9 Fungsi ini berguna untuk memeriksa kode registrasi.

- 10 • **index**

11 Fungsi ini berguna untuk melakukan validasi *username* dan *password* pengguna. Selain itu,  
12 fungsi ini juga memperbaharui *log* pada tabel *login*.

- 13 • **register**

14 Fungsi ini berguna untuk melakukan validasi dalam pembuatan akun.

- 15 • **logout**

16 Fungsi ini berguna untuk menghancurkan *session* dari pengguna dan memindahkan pengguna  
17 ke halaman *login*.

- 18 • **lost**

19 Fungsi ini berguna untuk mengirim *email* lupa password.

- 20 • **rest**

21 Fungsi ini berguna untuk melakukan *reset password* pengguna.

### 22 **Logs.php**

23 *Controller Logs.php* hanya memiliki satu buah fungsi bernama **index**. Fungsi ini berguna untuk  
24 mengambil dan memberikan data pada halaman *logs* menggunakan *Logs\_model*.

### 25 **Moss.php**

26 Berikut merupakan fungsi-fungsi pada *controller Moss.php*.

- 27 • **index**

28 Fungsi ini berguna untuk mengambil dan memberikan data pada halaman *moss*.

- 29 • **update**

30 Fungsi ini berguna untuk memperbaharui *moss\_userid* yang dimasukan oleh pengguna.

- 31 • **\_detec**

32 Fungsi ini berguna untuk melakukan pengecekan terhadap *submission*. !!TODO

### 33 **Notification.php**

34 Berikut merupakan fungsi-fungsi pada *controller Notification.php*.

- 35 • **index**

36 Fungsi ini berguna untuk mengambil dan memberikam data pada halaman *notifications*  
37 menggunakan *assignment\_model* dan *notifications\_model*.

- **add**

Fungsi ini berguna untuk menambahkan data *notifications*.

- **edit**

Fungsi ini berguna untuk memperbaharui data *notifications*.

- **delete**

Fungsi ini berguna untuk menghapus data *notifications*.

- **check**

Fungsi ini berguna memeriksa *notifications* baru.

### **Problems.php**

Berikut merupakan fungsi-fungsi pada *controller Problems.php*.

- **index**

Fungsi ini berguna untuk mengambil dan memberikan data *problems* sesuai dengan *assignment* tertentu pada halaman *problems*.

- **edit**

Fungsi ini berguna untuk memperbaharui deskripsi *problems* pada *assignment* tertentu.

### **Profile.php**

Berikut merupakan fungsi-fungsi pada *controller Profile.php*.

- **index**

Fungsi ini berguna untuk mengambil dan memberikan data pada halaman *profile*. Selain itu, fungsi ini berguna untuk melakukan pembaharuan data *profile*.

- **\_password\_check**

Fungsi ini berguna untuk melakukan validasi terhadap *password* yang akan dimasukkan pengguna sesuai dengan aturan.

- **\_password\_again\_check**

Fungsi ini berguna untuk melakukan validasi terhadap pengulangan *password* yang dimasukkan pengguna.

- **\_email\_check**

Fungsi ini berguna untuk melakukan validasi terhadap *email* yang dimasukkan pengguna

- **\_role\_check**

Fungsi ini berguna untuk melakukan validasi *role* pengguna.

### **Queue.php**

- **index**

Fungsi ini berguna untuk mengambil dan memberikan data pada halaman *queue* menggunakan tiga buah *model*. *Model* tersebut adalah *Assignment\_model*, *queue\_model*, dan *settings\_model*.

- **pause**

Fungsi ini berguna untuk memperbaharui data pada tabel *settings*.

- **resume**

Fungsi ini berguna untuk melanjutkan proses *queue*.

- `empty_queue`

Fungsi ini berguna untuk menghapus data tabel *queue*.

### **Queueprocess.php**

*Controller Queueprocess.php* hanya memiliki satu buah fungsi bernama `index`. Fungsi ini berguna untuk menjalankan proses *judge* berdasarkan *queue* satu demi satu sesuai antrean. Fungsi ini menggunakan beberapa *model* yaitu `Queue_model`, `Submit_model`, `Assignments_model`, dan `Settings_model`.

### **Rejudge.php**

Berikut merupakan fungsi-fungsi pada *controller Rejudge.php*.

- `index`

Fungsi ini berguna untuk mengambil dan memberikan data pada halaman *rejudge* menggunakan `Assignment_model`.

- `rejudge_single`

Fungsi ini berguna untuk melakukan *rejudge* pada satu buah masalah tertentu.

### **Scoreboard.php**

Berikut merupakan fungsi-fungsi pada *controller Scoreboard.php*.

- `index`

Fungsi ini berguna untuk memberikan data pada halaman *scoreboard* menggunakan dua buah *model*. *Model* tersebut adalah `Assignment_model` dan `Scoreboard_model`.

### **Server\_time.php**

*Controller Server\_time.php* hanya memiliki satu buah fungsi bernama `index`. Fungsi ini berguna untuk mengeluarkan *server\_time*.

### **Settings.php**

Berikut merupakan fungsi-fungsi pada *controller Settings.php*.

- `index`

Fungsi ini berguna untuk mengambil dan memberikan data pada halaman *settings* menggunakan `Settings_model` dan `Assignment_model`.

- `update`

Fungsi ini berguna untuk mengambil masukan dan memperbaharui data pada halaman *settings* berdasarkan masukan tersebut. Data tersebut nantinya akan disimpan pada *database* menggunakan fungsi `Settings_model`.

### **Submission.php**

Berikut merupakan fungsi-fungsi pada *controller Submission.php*.

- `_download_excel`

Fungsi ini berguna untuk mengubah data-data dari *submission* yang dipilih menjadi format *excel*.

- `final_excel`

Fungsi ini berguna untuk mengunduh data *final submissions*.

- `all_excel`

Fungsi ini berguna untuk mengunduh data seluruh *submissions*.

- `the_final`

Fungsi ini berguna untuk memberikan data pada halaman *Final Submissions* menggunakan beberapa *model*. *Model* tersebut terdiri dari `Submit_model`, `Settings_model`, dan `User_model`.

- `all`

Fungsi ini berguna untuk memberikan data pada halaman *All Submissions* menggunakan beberapa *model*. *Model* tersebut terdiri dari `Submit_model`, `Settings_model`, dan `User_model`.

- `select`

Fungsi ini berguna untuk memilih *submission* yang akan dijadikan *submission final* oleh pengguna.

- `_check_type`

Fungsi ini berguna untuk melakukan pengecekan tipe *submission* yang telah dikumpulkan oleh pengguna.

- `view_code`

Fungsi ini berguna untuk memperlihatkan *submission* yang telah dikumpulkan oleh pengguna sesuai dengan tipenya.

- `download_file`

Fungsi ini berguna untuk mengunduh hasil dari *submission* yang telah dikumpulkan oleh pengguna.

## **Submit.php**

Berikut merupakan fungsi-fungsi pada *controller* `Submit.php`.

- `_language_to_type`

Fungsi ini berguna untuk mengubah bahasa pemrograman menjadi tipe sesuai dengan pilihan pengguna.

- `_language_to_ext`

Fungsi ini berguna untuk mengubah bahasa pemrograman menjadi ekstensi sesuai dengan pilihan pengguna.

- `_match`

Fungsi ini berguna untuk mencocokkan tipe dengan ekstensi dari bahasa pemrogramannya.

- `_check_language`

Fungsi ini berguna untuk melakukan pengecekan terhadap bahasa pemrograman yang digunakan.

- `index`

Fungsi ini berguna untuk mengambil dan memberikan data pada halaman *submit* menggunakan `Assignment_model`.

- `_upload`

Fungsi ini berguna untuk menyimpan jawaban dan memasukannya ke *queue* untuk dinilai.

- `load`

Fungsi ini berguna untuk memuat kode dari *editor file*.

- `save`

Fungsi ini berguna untuk menyimpan kode menuju *editor file* dan mengirim ataupun menjalankannya.

- `_submit`

Fungsi ini berguna untuk menambahkan kode pada *queue* untuk dilakukan *judge*.

- `_execute`

Fungsi ini berguna untuk menambahkan kode untuk dijalankan atau di *queue*.

- `get_output` Fungsi ini berguna untuk memuat *file* menjadi hasil eksekusi.

### **User.php**

Berikut merupakan fungsi-fungsi pada *controller User.php*.

- `index`

Fungsi ini berguna untuk mengambil dan memberikan data pada halaman *users*.

- `add`

Fungsi ini berguna untuk menambahkan pengguna baru sesuai dengan masukan.

- `delete`

Fungsi ini berguna untuk menghapus pengguna yang dipilih.

- `delete_submissions`

Fungsi ini berguna untuk menghapus *submission* dari sebuah pengguna.

- `list_excel`

Fungsi ini berguna untuk menghasilkan dan mengunduh data pengguna pada format *excel*.

## **3.3 Analisis Sistem Usulan**

### **3.3.1 Model**

### **3.3.2 View**

### **3.3.3 Controller**



## BAB 4

### PERANCANGAN

#### 4.1 Perancangan Konversi Menuju *CodeIgniter 4*

Bagian ini akan membahas mengenai implementasi konversi menuju *CodeIgniter 4* seperti yang telah dibahas pada Bab 2.

##### 4.1.1 Implementasi Persiapan *CodeIgniter 4*

Pertama perlu dilakukan pemasangan aplikasi *CodeIgniter 4*. Pemasangan ini dapat dilakukan dengan pengunduhan manual melalui situs resmi *CodeIgniter 4* ataupun melalui *composer*. Pemasangan melalui *composer* dapat dilakukan menggunakan *command line* dengan kode sebagai berikut:

```
composer create-project codeigniter4/appstarter SharIF-JudgeV3
```

Setelah dilakukan pemasangan *CodeIgniter 4*, perlu dilakukan pemasangan komponen pendukung melalui *composer* seperti *Twig* sebagai *template engine*, *PHPOffice*, *RADIUS*, dan *ADLDAP*. Pemasangan komponen pendukung *Twig* dapat dilakukan menggunakan *command line* dengan kode sebagai berikut:

```
composer require "twig/twig:^2.0"
```

Pemasangan komponen pendukung *PHPOffice* dapat dilakukan sebagai berikut:

```
composer require phpoffice/phpspreadsheet
```

Pemasangan komponen pendukung *RADIUS* dapat dilakukan dengan cara berikut:

```
composer require dapphp/radius
```

Pemasangan komponen pendukung *ADLDAP* dapat dilakukan dengan cara berikut:

### 4.1.2 Pemindahan *file* dari *CodeIgniter 3* ke *CodeIgniter 4*

*CodeIgniter 4* memiliki struktur berbeda dengan *CodeIgniter 3* sehingga kita memerlukan pemindahan *file-file* sesuai dengan struktur yang ada pada *CodeIgniter 4*. Data-data konfigurasi yang terdapat pada `application/config` akan dipindahkan menuju `app/config`. Data-data pada `model` akan dipindahkan menuju `app/Models`. Data-data *controllers* akan dipindahkan menuju `app/Controllers`. Data-data *views* akan dipindahkan menuju `app/Views`. Sedangkan data-data seperti aset, *javascript*, dan logo akan dipindahkan menuju `public`.

### 4.1.3 Pembaharuan *file-file CodeIgniter 3*

*File-file* yang telah dipindahkan akan dilakukan pembaharuan agar sesuai dengan standar pada *CodeIgniter 4*.

#### *Model*

Seluruh *file model* yang telah dipindahkan akan dilakukan perubahan dengan menghapuskan `defined('BASEPATH')` OR `exit('No direct script access allowed')`; dan menambahkan kedua kode berikut:

- `namespace App\Models;`
- `use CodeIgniter\Model;`

Selanjutnya dilakukan perubahan seluruh pemanggilan *model* dari `$this->load->model('x');` menjadi `$this->x = new X();`.

#### *View*

#### *Controller*

*File-file* yang telah dipindahkan akan dilakukan perubahan dengan menghapuskan `defined('BASEPATH')` OR `exit('No direct script access allowed');`. Selanjutnya dilakukan perubahan dengan mengganti kode berikut:

```
extends CI_Controller
```

Menjadi kode berikut: `extends BaseController`.

Setelah itu dilakukan penambahan kode `namespace App\Controllers;` setelah pembukaan tag *PHP*.

#### *Library*

## 4.2 Implementasi Analisis Konversi *CodeIgniter 3* ke *CodeIgniter 4*



## DAFTAR REFERENSI

- [1] Version 3.1.13 (2022) *CodeIgniter User Guide*. CodeIgniter Foundation. Richmond,Canada.
- [2] Version 1.4 (2023) *SharIF Judge Documentation*. Informatika UNPAR. Jl. Ciumbuleuit No. 94, Bandung.
- [3] Version 4.3 (2023) *CodeIgniter User Guide*. CodeIgniter Foundation. Richmond,Canada.
- [4] Prihatini, F. N. dan Indudewi, D. (2016) Kesadaran dan Perilaku Plagiarisme dikalangan Mahasiswa(Studi pada Mahasiswa Fakultas Ekonomi Jurusan Akuntansi Universitas Semarang). *Dinamika Sosial Budaya*, **18**, 68–75.
- [5] Kurnia, A., Lim, A., dan Cheang, B. (2001) Online judge. *Computers & Education*, **18**, 299–315.



# LAMPIRAN A

## KODE PROGRAM

### Kode 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
```

### Kode 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