

**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 . . . . .	3
1.3 Tujuan . . . . .	3
1.4 Batasan Masalah . . . . .	3
1.5 Metodologi . . . . .	3
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> . . . . .	13
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> . . . . .	17
2.2.4 Users . . . . .	17
2.2.5 Menambah <i>Assignment</i> . . . . .	18
2.2.6 <i>Sample Assignment</i> . . . . .	22
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> . . . . .	29
2.2.11 <i>Shield</i> . . . . .	29
2.3 CodeIgniter 4[3] . . . . .	31
2.3.1 <i>Models-Views-Controllers</i> . . . . .	31
2.3.2 <i>CodeIgniter URLs</i> . . . . .	34
2.3.3 <i>URI Routing</i> . . . . .	35
2.3.4 <i>Database</i> . . . . .	36
2.3.5 <i>Library</i> . . . . .	37
2.3.6 <i>Helpers</i> . . . . .	41
2.4 Konversi CodeIgniter 3 ke CodeIgniter 4[3] . . . . .	41
2.4.1 Struktur Aplikasi . . . . .	42
2.4.2 <i>Routing</i> . . . . .	42

2.4.3	<i>Model, View, and Controller</i>	42
2.4.4	<i>Configuration</i>	43
2.4.5	<i>Database</i>	44
2.4.6	<i>Migrations</i>	44
2.4.7	<i>Routing</i>	44
2.4.8	<i>Libraries</i>	44
2.4.9	<i>Helpers</i>	46
2.4.10	<i>Events</i>	46
2.4.11	<i>Framework</i>	46
<b>3</b>	<b>ANALISIS</b>	<b>47</b>
3.1	Analisis Sistem Kini	47
3.1.1	<i>Model</i>	47
3.1.2	<i>View</i>	52
3.1.3	<i>Controller</i>	60
3.2	<i>Library</i>	66
3.3	Analisis Sistem Usulan	69
3.3.1	Persiapan <i>CodeIgniter 4</i>	69
3.3.2	Struktur Aplikasi	69
3.3.3	<i>Routing</i>	71
3.3.4	<i>Model, View, and Controller</i>	71
3.3.5	<i>Libraries</i>	73
3.3.6	<i>Configuration</i>	76
3.3.7	<i>Database</i>	76
3.3.8	<i>Helpers</i>	77
<b>4</b>	<b>PERANCANGAN</b>	<b>79</b>
4.1	Instalasi <i>CodeIgniter 4</i>	79
4.2	Perubahan Struktur Aplikasi	79
4.2.1	app/Config	79
4.2.2	<i>Controllers</i>	80
4.2.3	<i>Filters</i>	81
4.2.4	<i>Libraries</i>	82
4.2.5	<i>Model</i>	90
4.2.6	<i>View</i>	90
4.2.7	public	91
	<b>DAFTAR REFERENSI</b>	<b>93</b>
	<b>A KODE PROGRAM</b>	<b>95</b>
	<b>B HASIL EKSPERIMEN</b>	<b>97</b>

## DAFTAR GAMBAR

1.1	Tampilan halaman <i>SharIF Judge</i> . . . . .	1
1.2	<i>Pemindahan struktur aplikasi menuju CodeIgniter 4</i> . . . . .	2
2.1	<i>Flow Chart Aplikasi CodeIgniter 3</i> . . . . .	5
2.2	Tampilan halaman <i>SharIF Judge</i> untuk menambahkan <i>assignment</i> . . . . .	19
3.1	Tampilan Halaman <i>Dashboard</i> . . . . .	52
3.2	Tampilan Halaman <i>Profile</i> . . . . .	53
3.3	Tampilan Halaman <i>Settings</i> . . . . .	53
3.4	Tampilan Halaman <i>Users</i> . . . . .	54
3.5	Tampilan Halaman <i>Notifications</i> . . . . .	54
3.6	Tampilan Halaman <i>Assignments</i> . . . . .	55
3.7	Tampilan Halaman <i>Problems</i> . . . . .	55
3.8	Tampilan Halaman <i>Submit</i> . . . . .	56
3.9	Tampilan Halaman <i>Final Submission</i> . . . . .	56
3.10	Tampilan Halaman <i>All Submission</i> . . . . .	57
3.11	Tampilan Halaman <i>Scoreboard</i> . . . . .	57
3.12	Tampilan Halaman <i>Hall of Fame</i> . . . . .	58
3.13	Tampilan Halaman <i>24-hour Log</i> . . . . .	58
3.14	Tampilan Halaman <i>ReJudge</i> . . . . .	59
3.15	Tampilan Halaman <i>Submission Queue</i> . . . . .	59
3.16	Tampilan Halaman <i>Cheat Detection</i> . . . . .	60
3.17	<i>Pemindahan struktur aplikasi menuju CodeIgniter 4</i> . . . . .	70
B.1	Hasil 1 . . . . .	97
B.2	Hasil 2 . . . . .	97
B.3	Hasil 3 . . . . .	97
B.4	Hasil 4 . . . . .	97

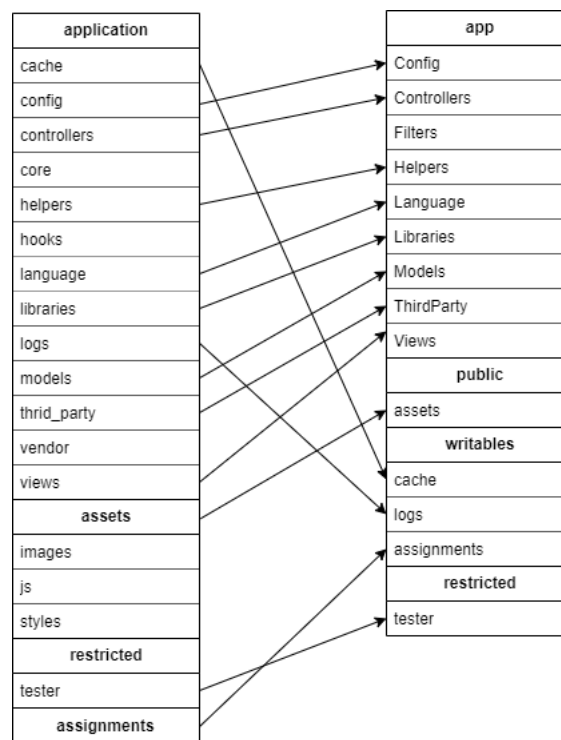




Sharif Judge pada awalnya dibentuk oleh Mohammad Javad menggunakan *framework* CodeIgniter 3 yang merupakan *framework* berbasis PHP (*Hypertext Preprocessor*). Sharif Judge kemudian di *fork* dan dimodifikasi menjadi SharIF Judge dengan penambahan fungsi sesuai kebutuhan Informatika Unpar untuk mengumpulkan tugas dan ujian mahasiswa[2].

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

CodeIgniter 4 merupakan versi terbaru dari *framework* CodeIgniter yang memiliki banyak perubahan fitur dari versi sebelumnya. CodeIgniter 4 dapat dijalankan menggunakan versi PHP 7.4 atau lebih baru sedangkan CodeIgniter 3 dapat dijalankan menggunakan versi PHP 5.6 atau lebih baru. CodeIgniter 4 juga membagi file menggunakan struktur MVC namun, memiliki struktur direktori berbeda dengan versi sebelumnya[3]. Perubahan direktori dapat dilihat pada gambar 1.2.



Gambar 1.2: Pemindahan struktur aplikasi menuju CodeIgniter 4

Gambar 1.2 merupakan perubahan struktur yang terdapat pada CodeIgniter 4. Rincian perubahan dapat dilihat pada bab 3

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

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



## 1.2 Rumusan Masalah

- Bagaimana cara melakukan konversi SharIF Judge pada CodeIgniter 3 menjadi CodeIgniter 4?
- Bagaimana mengevaluasi kode SharIF Judge pada CodeIgniter 3 dan mengubahnya agar dapat berjalan pada CodeIgniter 4?

## 1.3 Tujuan

Tujuan dari skripsi ini adalah sebagai berikut:

- Melakukan konversi dengan mengubah kode sesuai dengan standar CodeIgniter 4.
- Melakukan evaluasi kode SharIF Judge dan mengubahnya agar dapat berjalan di CodeIgniter 4.

## 1.4 Batasan Masalah

Batasan masalah pada pembentukan skripsi ini adalah sebagai berikut:

- EasySandbox tidak dijalankan karena tidak mendukung perangkat berbasis sistem operasi *MACOS*

## 1.5 Metodologi

Metodologi yang dilakukan dalam melakukan penelitian ini adalah sebagian berikut:

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

1    4. **Bab 4:** Perancangan

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

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

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

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

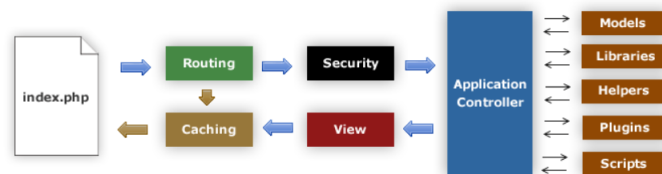
7        Bab ini berisikan kesimpulan dari hasil konversi yang telah dilakukan dan saran-saran terhadap  
8        perangkat lunak.

## BAB 2

### LANDASAN TEORI

#### 2.1 CodeIgniter 3<sup>[1]</sup>

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*

```

8
91 class Blog_model extends CI_Model {
102
113     public $title;
124     public $content;
135     public $date;
146
157     public function get_last_ten_entries()
168     {
179         $query = $this->db->get('entries', 10);
180         return $query->result();
191     }
202
213     public function insert_entry()
224     {
235         $this->title   = $_POST['title']; // please read the below note
246         $this->content = $_POST['content'];
257         $this->date    = time();
268
279         $this->db->insert('entries', $this);
280     }
281
292     public function update_entry()
303     {
314         $this->title   = $_POST['title'];
325         $this->content = $_POST['content'];
336         $this->date    = time();
347
358         $this->db->update('entries', $this, array('id' => $_POST['id']));
369     }
370 }
371

```

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 kode sebagai berikut:

```
$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*

```

1  <?php
2  <html>
3  <head>
4      <title>My Blog</title>
5  </head>
6  <body>
7      <h1>Welcome to my Blog!</h1>
8  </body>
9  </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');
```

## 16 *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*

```

21 <?php
22 class Blog extends CI_Controller {
23
24     public function index()
25     {
26         echo 'Hello World!';
27     }
28
29     public function comments()
30     {
31         echo 'Look at this!';
32     }
33 }

```

Kode 2.3 berfungsi dalam mengembalikan *string* sesuai dengan fungsi *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 dapat diambil dari *URL*.

### 41 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
```

*URL* diatas dibentuk berdasarkan *Controller* 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*

```
12 RewriteEngine On
13 1 RewriteCond %{REQUEST_FILENAME} !-f
14 2 RewriteCond %{REQUEST_FILENAME} !-d
15 3 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 berikut:

Kode 2.5: File *application/config.php*

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

Pengguna dapat mengubah *enable\_query\_strings* menjadi *TRUE*. Pengubahan fitur tersebut dapat memperbolehkan pengguna untuk menambahkan *query* pada *URL* yang dibentuk. Berikut merupakan contoh pengaksesan *URL* melalui *query strings*:

```
index.php?c=controller&m=method
```

*URL* dapat mengakses fungsi ataupun metode dari *controller* menggunakan *query* tanpa harus menggunakan *URL* biasa.

### 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* yang ingin digunakan. 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*

```

10 1 <?php
11 2 defined('BASEPATH') OR exit('No direct script access allowed');
12 3
13 4 class Someclass {
14 5
15 6     public function some_method()
16 7     {
17 8     }
18 9 }
```

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*:

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

Kode diatas memanggil *library someclass* dan 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:

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

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

```
<?php echo $library_src;?>
<?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:

```
$this->javascript
```

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

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

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

```
$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
- *ModeBCC 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*

```
21 $this->load->library('email');
22
23
24 $this->email->from('your@example.com', 'Your Name');
25 $this->email->to('someone@example.com');
26 $this->email->cc('another@another-example.com');
27 $this->email->bcc('them@their-example.com');
28
29 $this->email->subject('Email Test');
30 $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@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

```
41 $config['protocol'] = 'sendmail';
42 $config['mailpath'] = '/usr/sbin/sendmail';
43 $config['charset'] = 'iso-8859-1';
44 $config['wordwrap'] = TRUE;
45
46 $this->email->initialize($config);
```



Kode 2.8 merupakan contoh konfigurasi pengiriman *email* dengan protokol `sendmail`, tujuan `usr/sbin/sendmail`, *character set* berjenis `iso-8859-1`, dan menyalakan fitur `wordwrap`. Selanjutnya konfigurasi dapat diinisialisasikan menggunakan `initialize`.

#### 4 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

```

12  <?php
13  1
14  2
15  3 class Upload extends CI_Controller {
16  4
17  5     public function __construct()
18  6     {
19  7         parent::__construct();
20  8         $this->load->helper(array('form', 'url'));
21  9     }
22  0
23  1     public function index()
24  2     {
25  3         $this->load->view('upload_form', array('error' => ' ' ));
26  4     }
27  5
28  6     public function do_upload()
29  7     {
30  8         $config['upload_path']          = './uploads/';
31  9         $config['allowed_types']        = 'gif|jpg|png';
32  0         $config['max_size']             = 100;
33  1         $config['max_width']           = 1024;
34  2         $config['max_height']          = 768;
35  3
36  4         $this->load->library('upload', $config);
37  5
38  6         if ( ! $this->upload->do_upload('userfile'))
39  7         {
40  8             $error = array('error' => $this->upload->display_errors());
41  9
42  0             $this->load->view('upload_form', $error);
43  1         }
44  2         else
45  3         {
46  4             $data = array('upload_data' => $this->upload->data());
47  5
48  6             $this->load->view('upload_success', $data);
49  7         }
50  8     }
51  9 }
52  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* pada direktori `./uploads/`.

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.

## 1 Kelas *Zip Encoding*

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

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

7 Setelah diinisiasi, pengguna dapat memanggil *library* tersebut menggunakan kode sebagai  
 8 berikut:

```
9          $this->zip
```

10 Kode 2.10 merupakan contoh penggunaan *library Zip Encoding* untuk menyimpan dan menunduh  
 11 data.

Kode 2.10: Contoh penggunaan *library Zip Encoding*

```
12
131 $name = 'mydata1.txt';
142 $data = 'A Data String!';
153
164 $this->zip->add_data($name, $data);
175
186 // Write the zip file to a folder on your server. Name it "my_backup.zip"
197 $this->zip->archive('/path/to/directory/my_backup.zip');
208
219 // Download the file to your desktop. Name it "my_backup.zip"
220 $this->zip->download('my_backup.zip');
```

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

Kode 2.11: Contoh *library* yang dibentuk

```
28
291 class Example_library {
302
313     protected $CI;
324
335     // We'll use a constructor, as you can't directly call a function
346     // from a property definition.
357     public function __construct()
368     {
379         // Assign the CodeIgniter super-object
380         $this->CI =& get_instance();
391     }
402
413     public function foo()
424     {
435         $this->CI->load->helper('url');
446         redirect();
457     }
468 }
47
```

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

### 2.1.5 Database

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

Kode 2.12: Contoh konfigurasi *database*

```

3
41 $db['default'] = array(
52     'dsn' => '',
63     'hostname' => 'localhost',
74     'username' => 'root',
85     'password' => '',
96     'database' => 'database_name',
107    'dbdriver' => 'mysqli',
118    'dbprefix' => '',
129    'pconnect' => TRUE,
130    'db_debug' => TRUE,
141    'cache_on' => FALSE,
152    'cachedir' => '',
163    'char_set' => 'utf8',
174    'dbcollat' => 'utf8_general_ci',
185    'swap_pre' => '',
196    'encrypt' => FALSE,
207    'compress' => FALSE,
218    'stricton' => FALSE,
229    'failover' => array()
230 );

```

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*

```

30
311 $this->db->select('*');
322 $this->db->from('blogs');
333 $this->db->join('comments', 'comments.id = blogs.id');
344 $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 mebuat, menghapus, atau mengubah *database* harus dilakukan inisiasi sebagai berikut:

```
$this->load->dbforge()
```

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

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

```

44
451 $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*

```

49
501 $fields = array(
512     'blog_id' => array(
523         'type' => 'INT',
534         'constraint' => 5,
545         'unsigned' => TRUE,

```

```

16         'auto_increment' => TRUE
17     ),
18     'blog_title' => array(
19         'type' => 'VARCHAR',
20         'constraint' => '100',
21         'unique' => TRUE,
22     ),
23     'blog_author' => array(
24         'type' => 'VARCHAR',
25         'constraint' => '100',
26         'default' => 'King of Town',
27     ),
28     'blog_description' => array(
29         'type' => 'TEXT',
30         'null' => TRUE,
31     ),
32 );
33 $this->dbforge->add_field($fields);
34 $this->dbforge->create_table('table_name');

```

Kode 2.15 merupakan contoh penggunaan *database* untuk menambahkan kolom sesuai dengan tipe, batas dari data yang disimpan, penambahan otomatis, dan lainnya pada tabel `table_name`.

### 2.1.6 URI Routing

*URL string* yang dihasilkan *CodeIgniter 3* 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 metode dengan penambahan *segment* yang diinginkan.

Kode 2.16: Contoh *url* yang sudah dimetakan

```

29 example.com/product/1/
30 example.com/product/2/
31 example.com/product/3/
32 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*.

## 1 Ekspresi Reguler

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

```
4 $route['products/([a-z]+)/(\d+)'] = '$1/id_$2';
```

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

```
7 $route['login/(.+)'] = 'auth/login/$1';
```

### 8 2.1.7 Auto-loading

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

## 13 2.2 SharIF Judge[2]

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

### 19 2.2.1 Struktur Aplikasi

```
20 .
21 |
22 | -- application
23 | | -- cache
24 | | -- config
25 | | -- controllers
26 | | -- core
27 | | -- helpers
28 | | -- hooks
29 | | -- language
30 | | -- libraries
31 | | -- logs
32 | | -- models
33 | | -- third_party
34 | | -- vendor
35 | | -- views
36 | -- assets
37 | | -- images
38 | | -- js
39 | | -- styles
40 | -- restricted
41 | | -- tester
42 | -- system
43 | -- assignments
```

### 45 2.2.2 Instalasi

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

## 1 Persyaratan

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

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

Kode 2.17: Kode untuk melakukan pengujian fungsi

```
8
10 1 echo shell_exec("php_v");
```

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

## 15 Instalasi

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

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

```
20
21 1 $system_path = '/home/mohammad/secret/system';
22 2 application_folder = '/home/mohammad/secret/application';
```

- 24 3. Membangun *database MySQL* atau *PostgreSQL* untuk *SharIF Judge*. Jangan melakukan instalasi paket koneksi *database* apapun untuk *C*, *C++*, *Java*, atau *Python*.
- 26 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*

```
29
30 1 /* Enter database connection settings here: */
31 2 'dbdriver' => 'postgre', // database driver (mysqli, postgre)
32 3 'hostname' => 'localhost', // database host
33 4 'username' => '', // database username
34 5 'password' => '', // database password
35 6 'database' => '', // database name
36 7 'dbprefix' => 'shj_', // table prefix
```

- 38 5. Mengatur *RADIUS server* dan *mail server* pada file *application/config/secrets.example.php*
- 39 dan menyimpannya dengan nama *secrets.php*.
- 40 6. Mengatur *application/cache/Twig* agar dapat ditulis oleh PHP.
- 41 7. Membuka halaman utama SharIF Judge pada *web browser* dan mengikuti proses instalasi.
- 42 8. Melakukan *Log in* dengan akun admin.
- 43 9. Memindahkan direktori *tester* dan *assignments* diluar direktori publik dan mengatur kedua
- 44 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	Admin	Head Instructor	Instructor	Student
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>	✓	✓	✓	✓

## 1 Menambahkan Pengguna

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

Kode 2.20: Contoh sintaks untuk menambahkan pengguna

```

5  USERNAME,EMAIL,DISPLAY-NAME,PASSWORD,ROLE
6 1
7 2
8 3 * Usernames may contain lowercase letters or numbers and must be between 3 and 20 characters in length.
9 4 * Passwords must be between 6 and 30 characters in length.
10 5 * You can use RANDOM[n] for password to generate random n-digit password.
11 6 * ROLE must be one of these: 'admin', 'head_instructor', 'instructor', 'student'

```

- 13 Dengan contoh sebagai berikut:

Kode 2.21: Contoh kode untuk menambahkan pengguna

```

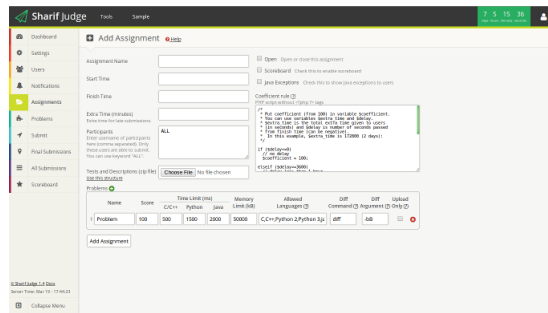
14 # This is a comment!
15 1 # This is another comment!
16 2
17 3 instructor,instructor@sharifjudge.ir,Instructor One,123456,head_instructor
18 4 instructor2,instructor2@sharifjudge.ir,Instructor Two,random[7],instructor
19 5 student1,st1@sharifjudge.ir,Student One,random[6],student
20 6 student2,st2@sharifjudge.ir,Student Two,random[6],student
21 7 student3,st3@sharifjudge.ir,Student Three,random[6],student
22 8 student4,st4@sharifjudge.ir,Student Four,random[6],student
23 9 student5,st5@sharifjudge.ir,Student Five,random[6],student
24 0 student6,st6@sharifjudge.ir,Student Six,random[6],student
25 1 student7,st7@sharifjudge.ir,Student Seven,random[6],student
26

```

## 27 2.2.5 Menambah *Assignment*

- 28 Pengguna dapat menambahkan *assignment* baru melalui bagian *Add* pada halaman *Assignmen-*  
 29 *ts*(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 memasukan *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 menunjukkan *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;

```

```

1  12
2  13 elseif (($delay-86400)<=3600)
3  14 // delay less than 1 hour in second day
4  15 $coefficient = ceil(70-((20*($delay-86400))/3600));
5  16
6  17 elseif (($delay-86400)<=86400)
7  18 // delay more than 1 hour in second day
8  19 $coefficient = 50;
9  20
10 21 elseif ($delay > $extra_time)
11 22 // too late
12 23 $coefficient = 0;
13

```

#### • *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.

Sample Input	Sample Output
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*.

Sample Input	Sample Output
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

```

```

19 | | |-- in
20 | | |-- input1.txt
21 | | |-- input2.txt
22 | | |-- input3.txt
23 | | |-- input4.txt
24 | | |-- input5.txt
25 | | |-- input6.txt
26 | | |-- input7.txt
27 | | |-- input8.txt
28 | | |-- input9.txt
29 | | --- input10.txt
30 | |-- out
31 | | |-- output1.txt
32 | | |-- output2.txt
33 | | |-- output3.txt
34 | | |-- output4.txt
35 | | |-- output5.txt
36 | | |-- output6.txt
37 | | |-- output7.txt
38 | | |-- output8.txt
39 | | |-- output9.txt
40 | | --- output10.txt
41 | |-- desc.md
42 | --- Problem2.pdf
43 | |-- p3
44 | --- desc.md
45 | --- SampleAssignment.pdf

```

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

### 33 *Sample Solutions*

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

35 Contoh solusi untuk bahasa *C*

Kode 2.24: Contoh skrip PHP

```

36
37 1 #include<stdio.h>
38 2 int main(){
39 3     int n;
40 4     scanf("%d",&n);
41 5     int i;
42 6     int sum =0 ;
43 7     int k;
44 8     for(i=0 ; i<n ; i++){
45 9         scanf("%d",&k);
46 0         sum+=k;
47 1     }
48 2     printf("%d\n",sum);
49 3     return 0;
50 4 }

```

52 Contoh solusi untuk bahasa *C++*

```

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

```

69 Contoh solusi untuk bahasa *C*

```

1
21 import java.util.Scanner;
32 class sum
43 {
54     public static void main(String[] args)
65     {
76         Scanner sc = new Scanner(System.in);
87         int n = sc.nextInt();
98         int sum =0;
109         for (int i=0 ; i<n ; i++)
110         {
121             int a = sc.nextInt();
132             sum += a;
143         }
154         System.out.println(sum);
165     }
176 }

```

Berikut merupakan contoh solusi untuk *problem 2*:

Contoh solusi untuk bahasa *C++*

```

21
221 #include<stdio.h>
232 int main(){
243     int n , m1=0, m2=0;
254     scanf("%d",&n);
265     for(;n--;){
276         int k;
287         scanf("%d",&k);
298         if(k>=m1){
309             m2=m1;
310             m1=k;
321         }
332         else if(k>m2)
343             m2=k;
354     }
365     printf("%d",m1+m2);
376     return 0;
387 }

```

contoh solusi untuk bahasa *C++*

```

41
421 #include<iostream>
432 using namespace std;
443 int main(){
454     int n , m1=0, m2=0;
465     cin >> n;
476     for(;n--;){
487         int k;
498         cin >> k;
509         if(k>=m1){
510             m2=m1;
521             m1=k;
532         }
543         else if(k>m2)
554             m2=k;
565     }
576     cout << (m1+m2) << endl ;
587     return 0;
598 }

```

### 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*

```
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
19     ...
20
21 }
```

## 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
```

```

1  24 | | |-- input6.txt
2  25 | | |-- input7.txt
3  26 | | |-- input8.txt
4  27 | | |-- input9.txt
5  28 | | --- input10.txt
6  29 | |-- out
7  30 | | |-- output1.txt
8  31 | | |-- output2.txt
9  32 | | |-- output3.txt
10 33 | | |-- output4.txt
11 34 | | |-- output5.txt
12 35 | | |-- output6.txt
13 36 | | |-- output7.txt
14 37 | | |-- output8.txt
15 38 | | |-- output9.txt
16 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*

```

20 1  /*
21 2  * tester.cpp
22 3  */
23 4
24 5 #include <iostream>
25 6 #include <fstream>
26 7 #include <string>
27 8 using namespace std;
28 9 int main(int argc, char const *argv[])
29 10 {
30 11
31 12     ifstream test_in(argv[1]); /* This stream reads from test's input file */
32 13     ifstream test_out(argv[2]); /* This stream reads from test's output file */
33 14     ifstream user_out(argv[3]); /* This stream reads from user's output file */
34 15
35 16     /* Your code here */
36 17     /* If user's output is correct, return 0, otherwise return 1 */
37 18     /* e.g.: Here the problem is: read n numbers and print their sum: */
38 19
39 20     int sum, user_output;
40 21     user_out >> user_output;
41 22
42 23     if ( test_out.good() ) // if test's output file exists
43 24     {
44 25         test_out >> sum;
45 26     }
46 27     else
47 28     {
48 29         int n, a;
49 30         sum=0;
50 31         test_in >> n;
51 32         for (int i=0 ; i<n ; i++){
52 33             test_in >> a;
53 34             sum += a;
54 35         }
55 36     }
56 37
57 38     if (sum == user_output)
58 39         return 0;
59 40     else
60 41         return 1;
61 42
62 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*

```

1  ...
2  ...
3  ...
4  $server = 'moss.stanford.edu';
5  $port = '7690';
6  $noreq = "Request_not_sent.";
7  $usage = "usage: _moss_ [-x] [-l _language_] [-d] [-b _basefile1_] ... [-b _basefileN_] [-m _#_] [-c \"string\"] [_file1_ _file2_ _file3_ ...]";
8  ...
9  #
10 # The userid is used to authenticate your queries to the server; don't change_it!
11 #
12 $userid=YOUR_MOSS_USER_ID;
13 ...
14 #
15 # Process the command line options. This is done in a non-standard
16 # way to allow multiple -b's.
17 #
18 $opt_l = "c"; # default language is c
19 $opt_m = 10;
20 $opt_d = 0;
21 ...
22 ...

```

*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

```
13
14 1 $ cd tester/easysandbox
15 2 $ chmod +x runalltests.sh
16 3 $ chmod +x runtest.sh
17 4 $ make runtests
```

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

#### Java Sandboxing

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

### 2.2.11 Shield

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

#### Shield untuk C/C++

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

Kode 2.33: Kode *shield* untuk melarang penggunaan `goto`

```
33
34 1 #define goto YouCannotUseGoto
```

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

- Menghidupkan *shield* untuk *C/C++*  
Pengguna dapat menghidupkan atau mematikan *shield* pada halaman *settings*.
- Menambahkan aturan untuk *C/C++* Daftar aturan `#define` untuk bahasa *C* terdapat pada halaman *tester/shield/defc.h* dan *tester/shield/defcpp.h* untuk bahasa *C++*. Pengguna dapat menambahkan aturan baru pada *file* tersebut pada halaman *settings*. Berikut merupakan contoh sintaks pada untuk menambahkan aturan :

Kode 2.34: Sintaks aturan `#define`

```

1  /*
2
3  @file defc.h
4  There should be a newline at end of this file.
5  Put the message displayed to user after // in each line
6
7  e.g. If you want to disallow goto, add this line:
8  #define goto errorNo13    //Goto is not allowd
9
10 */
11
12 #define system errorNo1    //"system" is not allowed
13 #define freopen errorNo2   //File operation is not allowed
14 #define fopen errorNo3     //File operation is not allowed
15 #define fprintf errorNo4   //File operation is not allowed
16 #define fscanf errorNo5    //File operation is not allowed
17 #define feof errorNo6      //File operation is not allowed
18 #define fclose errorNo7    //File operation is not allowed
19 #define ifstream errorNo8   //File operation is not allowed
20 #define ofstream errorNo9  //File operation is not allowed
21 #define fork errorNo10     //Fork is not allowed
22 #define clone errorNo11    //Clone is not allowed
23 #define sleep errorNo12    //Sleep is not allowed

```

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*

```

42
43
44 # @file shield_py3.py
45
46 import sys
47 sys.modules['os']=None
48
49 BLACKLIST = [
50     #'__import__', # deny importing modules
51     'eval', # eval is evil
52     'open',
53     'file',
54     'exec',
55     'execfile',
56     'compile',
57     'reload',
58     #'input'
59 ]
60 for func in BLACKLIST:

```

```

19 if func in __builtins__.__dict__:
20     del __builtins__.__dict__[func]

```

## 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 penyimpan *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*

```

6
7 1
8 2 <?php
9 3
10 4 namespace App\Models;
11 5
12 6 use CodeIgniter\Model;
13 7
14 8 class UserModel extends Model
15 9 {
16 10     protected $table      = 'users';
17 11     protected $primaryKey = 'id';
18 12
19 13     protected $useAutoIncrement = true;
20 14
21 15     protected $returnType     = 'array';
22 16     protected $useSoftDeletes = true;
23 17
24 18     protected $allowedFields = ['name', 'email'];
25 19
26 20     // Dates
27 21     protected $useTimestamps = false;
28 22     protected $dateFormat    = 'datetime';
29 23     protected $createdField   = 'created_at';
30 24     protected $updatedField   = 'updated_at';
31 25     protected $deletedField   = 'deleted_at';
32 26
33 27     // Validation
34 28     protected $validationRules      = [];
35 29     protected $validationMessages   = [];
36 30     protected $skipValidation        = false;
37 31     protected $cleanValidationRules = true;
38 32
39 33 }

```

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

```

45
46 1 <?php
47 2
48 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*

```

56
57 1 <html>
58 2     <head>

```

```

13 <title>My Blog</title>
24 </head>
35 <body>
46 <h1>Welcome to my Blog!</h1>
57 </body>
68 </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*

```

10 <?php
11
12 namespace App\Controllers;
13
14 use CodeIgniter\Controller;
15
16 class Blog extends Controller
17 {
18     public function index()
19     {
20         return view('blog-view');
21     }
22 }
23

```

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

## 27 **Controllers**

*Contollers* merupakan kelas untuk mengambil atau memberikan data dari *models* menuju *views* untuk ditampilkan. Setiap pembentukan *controllers* dibentuk harus memperpanjang kelas *BaseController*. Kode 2.40 merupakan contoh *controllers* yang dibentuk.

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

```

31 <?php
32
33 namespace App\Controllers;
34
35 class Helloworld extends BaseController
36 {
37     public function index()
38     {
39         return 'Hello World!';
40     }
41
42     public function comment()
43     {
44         return 'I am not flat!';
45     }
46 }
47

```

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

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

Selain itu, *CodeIgniter 4* menyediakan fungsi bernama *Controller Filters* yang berfungsi untuk membiarkan pengguna membangun sebuah kondisi sebelum ataupun sesudah *controller* dijalankan. Kode 2.41 merupakan contoh penggunaan *filters*.

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

```

54 <?php
55
56 namespace App\Filters;
57
58

```

```

15 use CodeIgniter\Filters\FilterInterface;
26 use CodeIgniter\HTTP\RequestInterface;
37 use CodeIgniter\HTTP\ResponseInterface;
48 use Config\Database;
59
60 class MyFilter implements FilterInterface
61 {
62     public function before(RequestInterface $request, $arguments = null)
63     {
64         $session = \Config\Services::session();
65         $db = Database::connect();
66
67         if ( !$db->tableExists('sessions'))
68             return redirect()->to('/install');
69         if ( !$session->get('logged_in')) // if not logged in
70             return redirect()->to('/login');
71     }
72
73     public function after(RequestInterface $request, ResponseInterface $response, $arguments = null)
74     {
75         // Do something here
76     }
77 }

```

Kode 2.41 merupakan contoh kode yang akan melakukan pengecekan tabel ataupun *session* sebelum *controller* dijalankan. Kode ini akan disimpan pada direktori `app/Filters`. Selanjutnya pengguna perlu menambahkan konfigurasi *filter* pada *routes* seperti sintaks berikut.

```

28
29 $routes->get('/', 'Dashboard::index', ['filter' => 'check-install:dual,noreturn']);

```

Sintaks diatas akan melakukan pengecekan pada *controller* `Dashboard::index` sebelum dan 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*:

```

36 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

```

39 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:



## 1 *Apache Web Server*

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

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

```
4 RewriteEngine On
5 RewriteCond %{REQUEST_FILENAME} !-f
6 RewriteCond %{REQUEST_FILENAME} !-d
7 RewriteRule ^(.*)$ index.php/$1 [L]
```

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

## 12 *NGINX*

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

Kode 2.43: Contoh penggunaan `try-files`.

```
16 location / {
17     try_files $uri $uri/ /index.php$is_args$args;
18 }
19
```

## 21 **2.3.3 *URI Routing***

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

### 23 *Defined Route Routing*

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

Kode 2.44: Contoh *route* yang didefinisikan secara manual

```
26 <?php
27
28 $routes->get('product/(:num)', 'Catalog::productLookup');
```

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

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

```
34 <?php
35
36 $routes->group('admin', static function ($routes) {
37     $routes->get('users', 'Admin\Users::index');
38     $routes->get('blog', 'Admin\Blog::index');
39 });
```

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

## 1 *Auto Routing*

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

```
5 $routes->setAutoRoute(true);
```

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

## 9 **2.3.4 Database**

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

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

```
14 <?php
15 1 namespace Config;
16 2
17 3 use CodeIgniter\Database\Config;
18 4
19 5 class Database extends Config
20 6 {
21 7     public $default = [
22 8         'DSN' => '',
23 9         'hostname' => 'localhost',
24 0         'username' => 'root',
25 1         'password' => '',
26 2         'database' => 'database_name',
27 3         'DBDriver' => 'MySQLi',
28 4         'DBPrefix' => '',
29 5         'pConnect' => true,
30 6         'DBDebug' => true,
31 7         'charset' => 'utf8',
32 8         'DBCollat' => 'utf8_general_ci',
33 9         'swapPre' => '',
34 0         'encrypt' => false,
35 1         'compress' => false,
36 2         'strictOn' => false,
37 3         'failover' => [],
38 4         'port' => 3306,
39 5     ];
40 6     // ...
41 7 }
42 8
43 9
44 0
```

46 Kode 2.46 merupakan contoh konfigurasi untuk database bernama `database_name` dengan  
47 *username* `root`. Selain itu, konfigurasi juga dapat dilakukan pada file `.env` untuk mempermudah  
48 dalam pengubahan pada saat melakukan *deploy*. Kode 2.47 merupakan contoh konfigurasi pada file  
49 `.env`:

Kode 2.47: Contoh konfigurasi *database* pada file `.env`.

```
50 database.default.username = 'root';
51 1 database.default.password = '';
52 2 database.default.database = 'ci4';
53 3
```

Kode 2.47 akan menyimpan konfigurasi pada grup *default* dengan *username* berupa root, tanpa menggunakan *password*, dan juga dengan nama *database* ci4. 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 2.48: Contoh penggunaan *query* menggunakan konfigurasi pada *CodeIgniter 4*.

```

5
61 <?php
72
83 $builder = $db->table('users');
94 $builder->select('title, content, date');
105 $query = $builder->get();

```

Kode 2.48 merupakan contoh penggunaan *query* untuk mengambil data *title*, *content*, dan *date* pada tabel *users*. *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 2.49 merupakan contoh pembentukan *database*.

Kode 2.49: Contoh pembentukan tabel melalui *database forge*.

```

16
171 <?php
182
193 $fields = [
204     'id' => [
215         'type'          => 'INT',
226         'constraint'    => 5,
237         'unsigned'      => true,
248         'auto_increment' => true,
259     ],
260     'title' => [
271         'type'          => 'VARCHAR',
282         'constraint'    => '100',
293         'unique'        => true,
304     ],
315     'author' => [
326         'type'          => 'VARCHAR',
337         'constraint'    => 100,
348         'default'       => 'King of Town',
359     ],
360     'description' => [
371         'type'          => 'TEXT',
382         'null'         => true,
393     ],
404     'status' => [
415         'type'          => 'ENUM',
426         'constraint'    => ['publish', 'pending', 'draft'],
437         'default'       => 'pending',
448     ],
459 ];
460 $forge->addField($fields);
461 $forge->createTable('table_name');

```

Kode 2.49 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 2.50 merupakan contoh konfigurasi preferensi *email* secara manual.

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

```

11 <?php
12
13
14 $config['protocol'] = 'sendmail';
15 $config['mailPath'] = '/usr/sbin/sendmail';
16 $config['charset'] = 'iso-8859-1';
17 $config['wordWrap'] = true;
18
19 $email->initialize($config);

```

Selain itu, pengguna dapat melakukan pengiriman *email* sesuai dengan kebutuhan. Kode 2.51 merupakan contoh penggunaan kelas *email* untuk mengirim *email*.

Kode 2.51: Contoh kode untuk melakukan pengiriman *email*.

```

23 <?php
24
25
26 $email = \Config\Services::email();
27
28 $email->setFrom('your@example.com', 'Your Name');
29 $email->setTo('someone@example.com');
30 $email->setCC('another@another-example.com');
31 $email->setBCC('them@their-example.com');
32
33 $email->setSubject('Email Test');
34 $email->setMessage('Testing the email class.');
```

Kode 2.51 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 2.52 merupakan contoh *view* untuk melakukan pengunggahan *file*.

Kode 2.52: Contoh kode untuk melakukan pengunggahan *file*.

```

48 <!DOCTYPE html>
49 <html lang="en">
50 <head>
51 <title>Upload Form</title>

```

```

15 </head>
16 <body>
17
18 <?php foreach ($errors as $error): ?>
19     <li><?= esc($error) ?></li>
20 <?php endforeach ?>
21
22 <?= form_open_multipart('upload/upload') ?>
23     <input type="file" name="userfile" size="20">
24     <br><br>
25     <input type="submit" value="upload">
26 </form>
27
28 </body>
29 </html>

```

Kode 2.52 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 2.53: Contoh kode *controller* untuk melakukan validasi dan penyimpanan.

```

21 <?php
22
23 namespace App\Controllers;
24
25 use CodeIgniter\Files\File;
26
27 class Upload extends BaseController
28 {
29     protected $helpers = ['form'];
30
31     public function index()
32     {
33         return view('upload_form', ['errors' => []]);
34     }
35
36     public function upload()
37     {
38         $validationRule = [
39             'userfile' => [
40                 'label' => 'Image File',
41                 'rules' => [
42                     'uploaded[userfile]',
43                     'is_image[userfile]',
44                     'mime_in[userfile,image/jpg,image/jpeg,image/gif,image/png,image/webp]',
45                     'max_size[userfile,100]',
46                     'max_dims[userfile,1024,768]',
47                 ],
48             ],
49         ];
50
51         if (!$this->validate($validationRule)) {
52             $data = ['errors' => $this->validator->getErrors()];
53
54             return view('upload_form', $data);
55         }
56
57         $img = $this->request->getFile('userfile');
58
59         if (!$img->hasMoved()) {
60             $filepath = WRITEPATH . 'uploads/' . $img->store();
61
62             $data = ['uploaded_fileinfo' => new File($filepath)];
63
64             return view('upload_success', $data);
65         }
66
67         $data = ['errors' => 'The file has already been moved.'];
68
69         return view('upload_form', $data);
70     }
71 }

```

Kode 2.53 terdapat dua buah fungsi yaitu:

- `index()` yang mengembalikan *view* bernama `upload_form`

- `upload()` yang memberikan aturan untuk melakukan validasi dan melakukan penyimpanan pada direktori `uploads`.

### Validation

*CodeIgniter 4* menyediakan *library* untuk melakukan validasi terhadap data yang dikirimkan oleh pengguna. Data yang divalidasi dapat diberikan aturan-aturan sesuai dengan konfigurasi pengguna. Kode 2.54 merupakan contoh penggunaan *validation*.

Kode 2.54: Contoh kode untuk melakukan pengumpulan data.

```

7  <html>
8  <head>
9  <title>My Form</title>
10 </head>
11 <body>
12
13 <?= validation_list_errors() ?>
14
15 <?= form_open('form') ?>
16
17     <h5>Username</h5>
18     <input type="text" name="username" value="<?= set_value('username') ?>" size="50">
19
20     <h5>Password</h5>
21     <input type="text" name="password" value="<?= set_value('password') ?>" size="50">
22
23     <h5>Password Confirm</h5>
24     <input type="text" name="passconf" value="<?= set_value('passconf') ?>" size="50">
25
26     <h5>Email Address</h5>
27     <input type="text" name="email" value="<?= set_value('email') ?>" size="50">
28
29     <div><input type="submit" value="Submit"></div>
30
31 <?= form_close() ?>
32
33 </body>
34 </html>

```

Kode 2.54 merupakan contoh penggunaan *validation* pada halaman *view*. Pengambilan *error* dapat menggunakan sintaks `validation_list_errors()`. Selanjutnya akan digunakan fungsi `form_open` untuk membuka *tag form* sesuai dengan *url* yang sudah dibentuk. Setiap input akan diberikan *name* sesuai dengan aturan yang ingin diberikan. Setelah *tag form* selesai maka akan ditutup dengan sintaks `form_close`. Setelah itu data-data yang sudah dimasukan akan dikirimkan menuju *controller* seperti pada kode 2.55.

Kode 2.55: Contoh kode untuk melakukan validasi data yang sudah dikumpulkan.

```

43 <?php
44
45 namespace App\Controllers;
46
47 class Form extends BaseController
48 {
49     protected $helpers = ['form'];
50
51     public function index()
52     {
53         if (! $this->request->is('post')) {
54             return view('signup');
55         }
56
57         $rules = [];
58
59         if (! $this->validate($rules)) {
60             return view('signup');
61         }
62
63         // If you want to get the validated data.

```

```

122         $validData = $this->validator->getValidated();
123
124         return view('success');
125     }
126 }

```

7 Data-data yang sudah diberikan oleh pengguna akan dijalankan menggunakan *controller* diatas.  
 8 Sintaks `if (! $this->request->is('post'))` akan melakukan pengecekan apakah *request* yang  
 9 diberikan berupa *post* atau tidak. Selanjutnya dapat ditentukan aturan pada variabel **rules** yang  
 10 nantinya akan dilakukan validasi menggunakan fungsi **validate**. Fungsi **validate** akan mengecek  
 11 data yang diberikan dan menentukan apakah sudah sesuai dengan aturan yang ada atau belum.  
 12 Kode 2.56 merupakan contoh pembentukan aturan sesuai dengan nama *form* yang ada.

Kode 2.56: Contoh kode untuk menetapkan aturan untuk validasi data yang sudah dikumpulkan.

```

13
14 1 $rules = [
15 2     'username' => 'required|max_length[30]',
16 3     'password' => 'required|max_length[255]|min_length[10]',
17 4     'passconf' => 'required|max_length[255]|matches[password]',
18 5     'email'    => 'required|max_length[254]|valid_email',
19 6 ];

```

21 Kode 2.56 merupakan contoh aturan yang ditetapkan untuk setiap *form* yang ada. Aturan-aturan  
 22 tersebut dapat diganti sesuai dengan kebutuhan dari pengguna. Selain menggunakan aturan yang  
 23 disediakan *CodeIgniter 4*, pengguna dapat membentuk aturannya sendiri pada file *Validation.php*.  
 24 Kode 2.57 merupakan contoh aturan yang dibentuk secara manual.

Kode 2.57: Contoh kode pembentukan aturan secara manual pada file *Validation.php*.

```

25
26 1 <?php
27 2
28 3 class MyRules
29 4 {
30 5     public function even($value): bool
31 6     {
32 7         return (int) $value % 2 === 0;
33 8     }
34 9 }

```

36 Kode 2.57 merupakan contoh pembentukan aturan secara manual. Penggunaan aturan yang  
 37 dibentuk secara manual sama seperti penggunaan aturan lainnya.

### 38 2.3.6 *Helpers*

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

```

41
42         <?php
43         helper('helpers_name');

```

43 Setelah dilakukan pemanggilan, pengguna dapat memakai fungsi-fungsi yang disediakan sesuai  
 44 dengan *helpers* yang digunakan. Fungsi-fungsi itu antara lain adalah *form*, *array*, dan lainnya.

## 45 2.4 Konversi CodeIgniter 3 ke CodeIgniter 4[3]

46 Konversi CodeIgniter 3 ke CodeIgniter 4 diperlukan penulisan ulang karena terdapat banyak  
 47 implementasi yang berbeda. Konversi ke CodeIgniter 4 diawali dengan melakukan instalasi proyek  
 48 baru CodeIgniter 4.

### 2.4.1 Struktur Aplikasi

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

### 2.4.2 Routing

CodeIgniter 4 meletakkan *route* pada *file* `app\Config\Routes.php`. CodeIgniter 4 memiliki fitur *auto routing* seperti pada CodeIgniter 3 namun, pada *default* di matikan. Fitur *auto routing* memungkinkan untuk dinyalakan serupa dengan pada CodeIgniter 3 namun tidak direkomendasikan 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 1. Pertama pengguna perlu memindahkan seluruh *file views* menuju `app/Views`
- 2 2. Selanjutnya pengguna perlu mengubah sintaks:

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

4 menjadi sintaks berikut:

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

- 6 3. Pengguna juga perlu mengubah sintaks:

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

8 menjadi sintaks berikut:

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

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

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

12 menjadi sintaks berikut:

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

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

## 16 ***Controller***

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

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

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

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

21 Pengguna dapat melakukan pembaharuan *controller* menggunakan cara berikut:

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

## 28 **2.4.4 *Configuration***

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

30 versi sebelumnya. Penulisan berudah dari yang sebelumnya menggunakan *array* akan berubah

31 menjadi menggunakan variabel. Pengguna hanya perlu melakukan pemindahan menuju CodeIgniter

32 4 dan apabila menggunakan *file config custom* maka, diperlukan penulisan ulang pada direktori

33 Config dengan melakukan *extend* pada `CodeIgniter\Config\BaseConfig`. Beberapa konfigurasi

34 juga akan dipindahkan menuju file `.env`.

### 2.4.5 Database

Penggunaan *database* pada CodeIgniter 4 hanya berubah sedikit dibandingkan dengan versi sebelumnya. Data-data penting kredensial diletakan pada `app\Config\Database.php` dan perlu dilakukan beberapa perubahan sintaks dan *query*. Sintaks untuk memuat database diubah menjadi `$db = db_connect();` dan apabila menggunakan beberapa *database* maka sintaks menjadi `$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 = $db->table('mytable');` dan dapat dipakai untuk menjalankan *query* dengan mengganti `$this->db` seperti `$this->db->get();` menjadi `$builder->get();`.

### 2.4.6 Migrations

Perubahan perlu dilakukan pada nama *file* menjadi nama dengan cap waktu. Selanjutnya dilakukan penghapusan kode `defined('BASEPATH') OR exit('No direct script access allowed');` 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 metode *Forge* dari yang sebelumnya bernama `$this->dbforge->add_field` menjadi menggunakan *camelCase* `$this->forge->addField`.

### 2.4.7 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');
```

Sintak diatas berguna untuk memanggil fungsi `index` pada *controller* `Blogs`.

- `$route['product/(:any)'] = 'catalog/product_lookup'`  
menjadi

```
$routes->add('product/(:segment)', 'Catalog::productLookup');
```

### 2.4.8 Libraries

CodeIgniter 4 menyediakan *library* untuk digunakan dan dapat diinstall apabila diperlukan. Pemanggilan *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:

## 1 *Emails*

2 Perubahan *email* hanya terdapat pada nama dari *method* dan pemanggilan *library email*. Pemanggilan *library* berubah dari `$this->load->library('email');` menjadi `$email = service('email');`; dan selanjutnya perlu dilakukan perubahan pada semua `$this->email` menjadi `$email`. Selanjutnya beberapa pemanggilan *method* berubah dengan tambahan *set* didepannya seperti *from* menjadi *setFrom*.

## 7 *Working with Uploaded Files*

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

```
11 $file = $this->request->getFile('userfile')
```

12 selanjutnya dapat dilakukan validasi dengan cara sebagai berikut:

```
13 $file->isValid()
```

14 *File* yang sudah diunggah dapat disimpan dengan sintaks berikut:

```
15 $path = $this->request->getFile('userfile')->store('head_img/', 'user_name.jpg');
```

16 Sintaks diatas akan mengambil file dengan atribut nama *userfile* dan menyimpannya pada direktori *head\_img* dengan nama file *user\_name.jpg*.

## 18 *HTML Tables*

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

## 24 *Localization*

25 *CodeIgniter 4* mengembalikan *file* bahasa menjadi *array* sehingga perlu dilakukan beberapa perubahan. Pertama, perlu dilakukan konfigurasi *default language* pada perangkat lunak. Selanjutnya melakukan pemindahan *file* bahasa pada *CodeIgniter 3* menuju `app\Language\<locale>`. *File-file* bahasa *CodeIgniter 3* perlu dilakukan penghapusan semua kode `$this->lang->load($file, $lang);` dan mengubah *method* pemanggilan bahasa dari `$this->lang->line('error_email_missing')` menjadi `echo lang('Errors.errorEmailMissing');`

## 31 *Validations*

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

- 33 1. Pengguna harus mengubah kode pada *view* dari `<?php echo validation_errors(); ?>`  
34 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 2.58) untuk melakukan validasi.

Kode 2.58: Perubahan kode untuk melakukan validasi.

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

### 2.4.9 *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`

### 2.4.10 *Events*

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

```
$hook['post_controller_constructor']
```

menjadi

```
Events::on('post_controller_constructor', ['MyClass', 'MyFunction']);}
```

Dan menambahkan *namespace* `CodeIgniter\Events\Events`;

### 2.4.11 *Framework*

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

## BAB 3

## ANALISIS

### 3.1 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*.

#### 3.1.1 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.

- `add_notification`

Fungsi ini berguna untuk menambahkan *notification* baru.

- `update_notification`

Fungsi ini berguna untuk memperbaharui *notification* tertentu.

- `delete_notification`

Fungsi ini berguna untuk menghapus *notification* tertentu.

- `get_notification`

Fungsi ini berguna untuk mengembalikan *notification* dalam bentuk *array*.

- `have_new_notification`

Fungsi ini berguna untuk mengecek apakah terdapat *notification* setelah waktu tertentu.

#### 7 `Queue_model.php`

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

- `in_queue`

Fungsi ini berguna untuk mengecek apakah *submission* pengguna tertentu sudah dalam antrian.

- `get_queue`

Fungsi ini berguna untuk mengembalikan data seluruh antrian.

- `empty_queue`

Fungsi ini berguna untuk menghapus seluruh tabel *queue*.

- `add_to_queue`

Fungsi ini berguna untuk memasukkan *submission* kedalam tabel *queue*.

- `rejudge`

Fungsi ini berguna untuk menambahkan *submission* kedalam antrian untuk dilakukan *rejudge*.

- `rejudge_single`

Fungsi ini berguna untuk menambahkan satu buah *submission* kedalam antrian untuk dilakukan *rejudge*.

- `get_first_item`

Fungsi ini berguna untuk mengambil data pertama dalam antrian.

- `remove_item`

Fungsi ini berguna untuk menghapus data tertentu dalam antrian.

- `save_judge_result_in_db`

Fungsi ini berguna untuk menyimpan hasil dari *judge* ke dalam *database*.

- `add_to_queue_exec`

Fungsi ini berguna untuk menambahkan data *dummy* pada antrian.

#### 32 `Scoreboard_model.php`

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

- `_generate_scoreboard`

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*.

#### **User\_model.php**

Berikut merupakan fungsi-fungsi dari `User_model.php` yang berguna untuk memproses data pada tabel *users*.

- `have_user`

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



- `user_id_to_username`

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

- `username_to_user_id`

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

- `have_email`

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

- `add_user`

Fungsi ini berguna untuk menambahkan sebuah pengguna.

- `add_users`

Fungsi ini berguna untuk menambahkan banyak pengguna.

- `delete_user`

Fungsi ini berguna untuk menghapus pengguna tertentu.

- `delete_submissions`

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

- `validate_user`

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

- `selected_assignment`

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

- `get_names`

Fungsi ini berguna untuk mengembalikan nama dari pengguna tertentu.

- `update_profile`

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

- `send_password_reset_mail`

Fungsi ini berguna untuk menghasilkan *password reset key* dan mengirim *email* untuk melakukan *reset password*.

- `passchange_is_valid`

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

- `reset_password`

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

- `get_all_users`

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

- `get_user`

Fungsi ini berguna untuk mengembalikan baris data untuk pengguna tertentu.

- `update_login_time`

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

### *User.php*

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

- `select_assignment`

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

- `save_widget_positions`

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

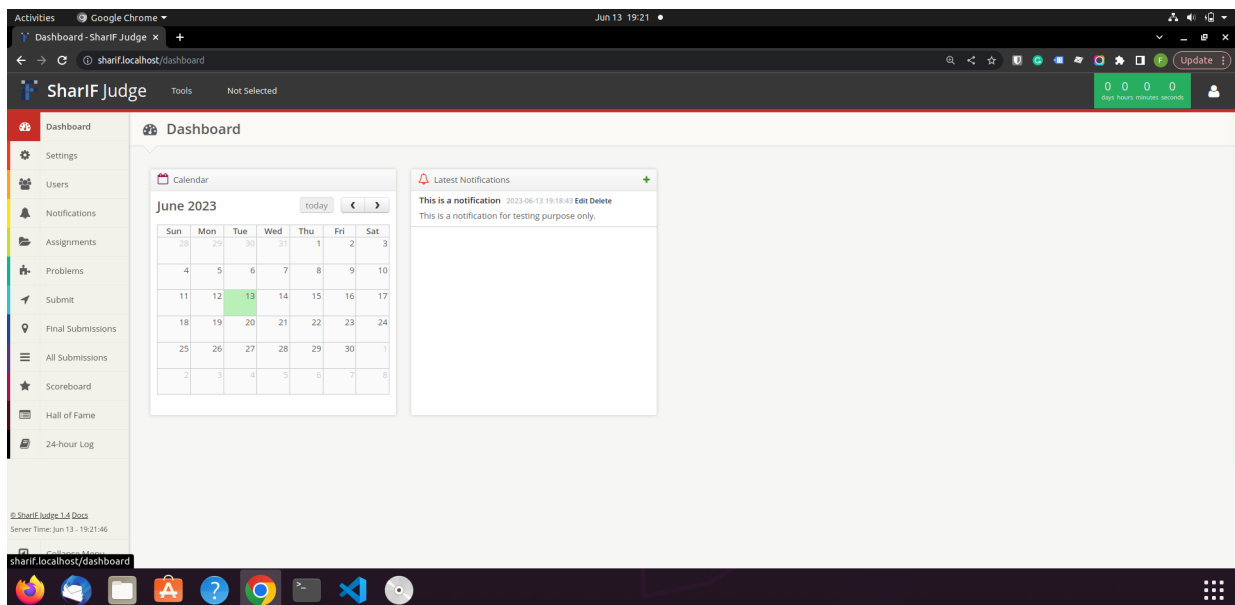
- `get_widget_positions`

Fungsi ini berguna untuk mengembalikan data *dashboard widget*.

### 3.1.2 View

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

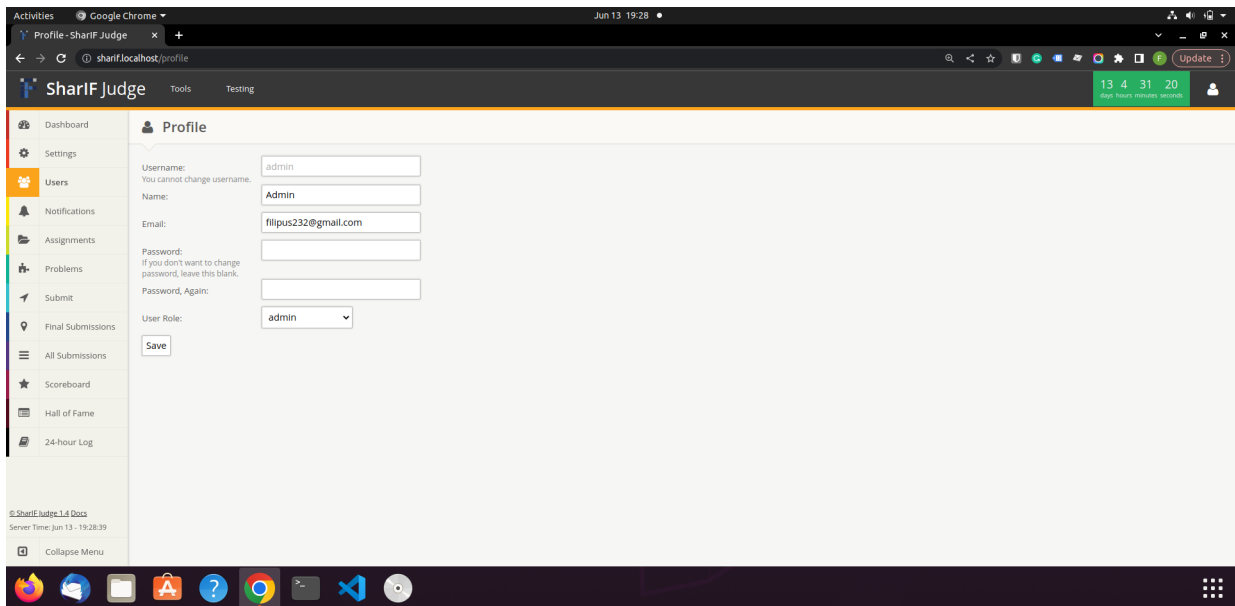
### Dashboard



Gambar 3.1: Tampilan Halaman Dashboard

Gambar 3.1 merupakan tampilan halaman *dashboard* yang terdapat pada semua *role* pengguna.

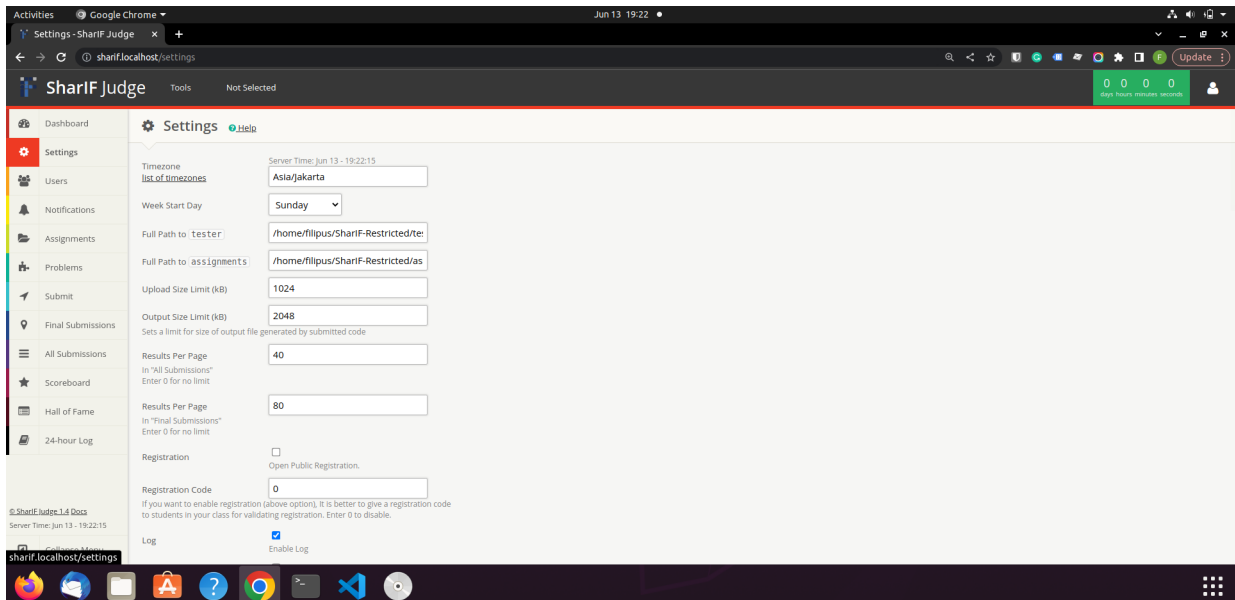
## 1 Profile



Gambar 3.2: Tampilan Halaman Profile

- 2 Gambar 3.2 merupakan tampilan halaman *profile* yang terdapat pada semua *role* pengguna. Namun,  
3 terdapat fitur yang tidak dapat digunakan oleh *siswa* dan *instructor* yakni mengganti role.

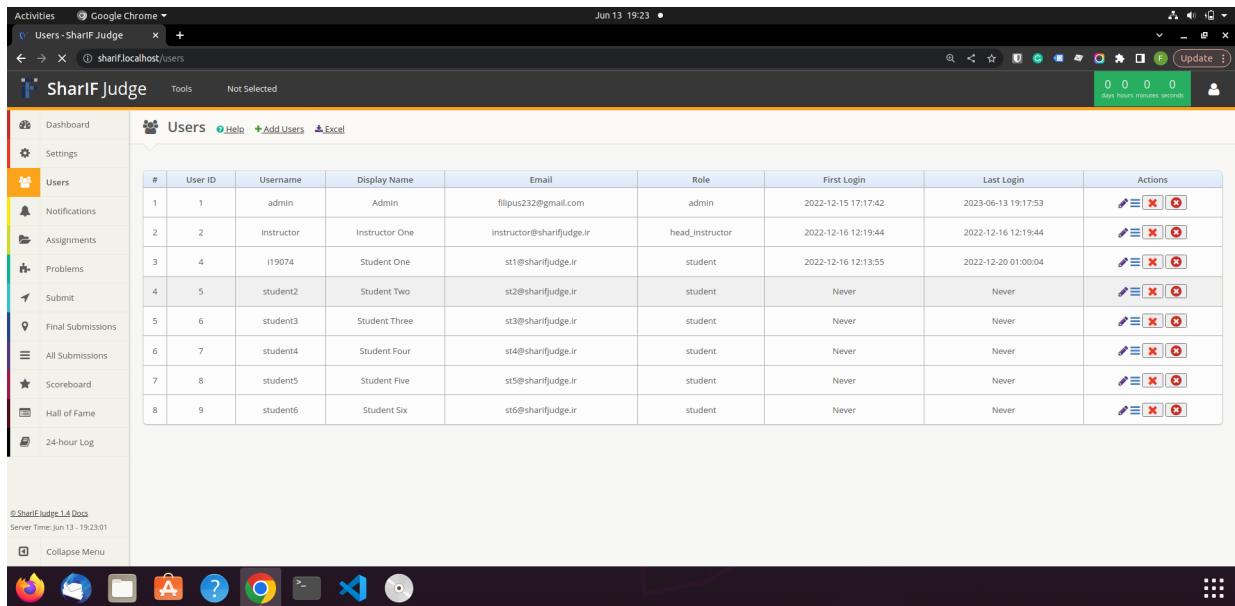
## 4 Settings



Gambar 3.3: Tampilan Halaman Settings

- 5 Gambar 3.3 merupakan tampilan halaman *settings* yang terdapat hanya pada *role* admin dan *head*  
6 *instructor*.

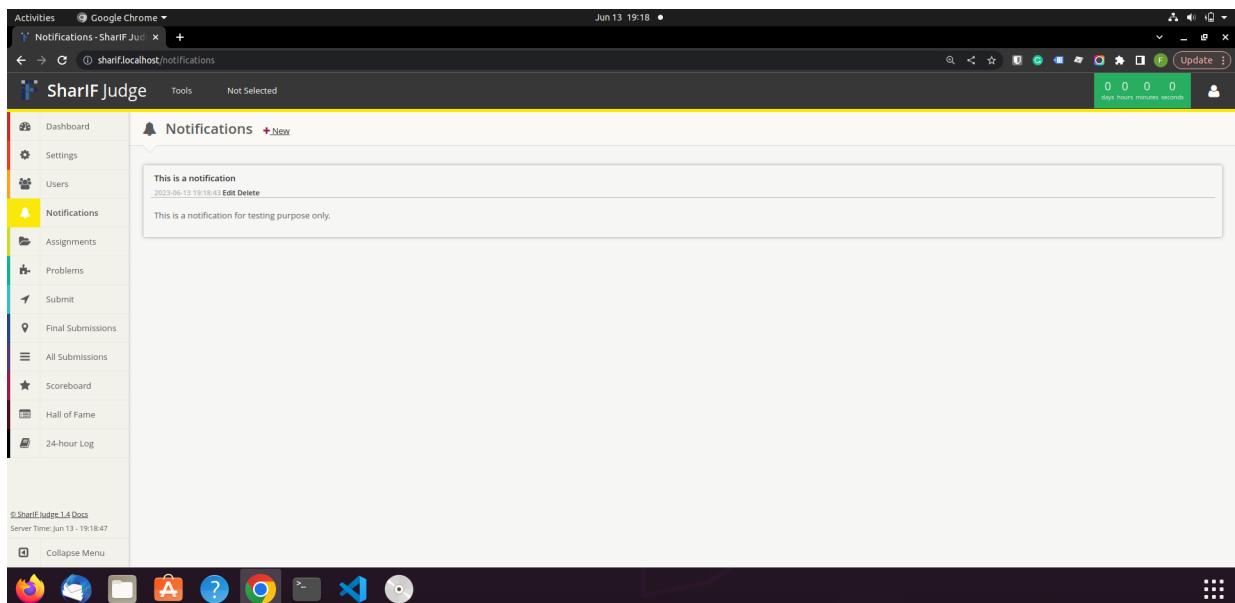
## 1 Users



Gambar 3.4: Tampilan Halaman Users

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

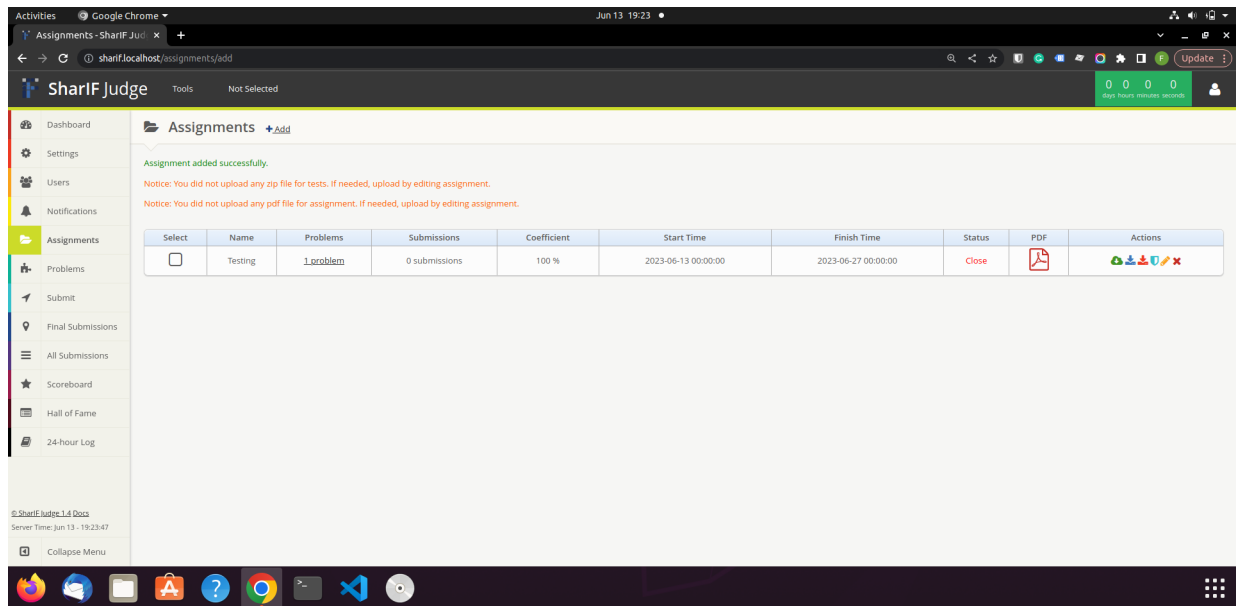
## 4 Notifications



Gambar 3.5: Tampilan Halaman Notifications

- 5 Gambar 3.5 merupakan tampilan halaman *notifications* yang terdapat pada semua *role* pengguna.

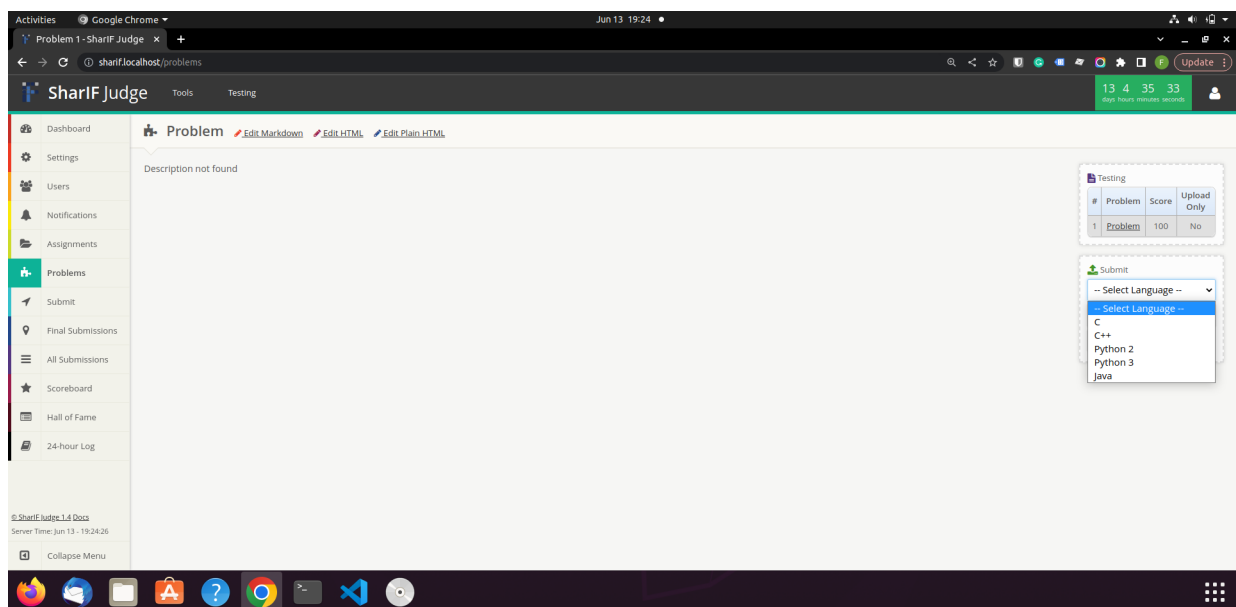
## 1 *Assignments*



Gambar 3.6: Tampilan Halaman Assignments

- 2 Gambar 3.6 merupakan tampilan halaman *assignments* yang terdapat pada semua *role* pengguna.
- 3 Namun, terdapat bagian yang tidak dapat diakses oleh *role* siswa dan *instructor* yakni bagian
- 4 *actions*.

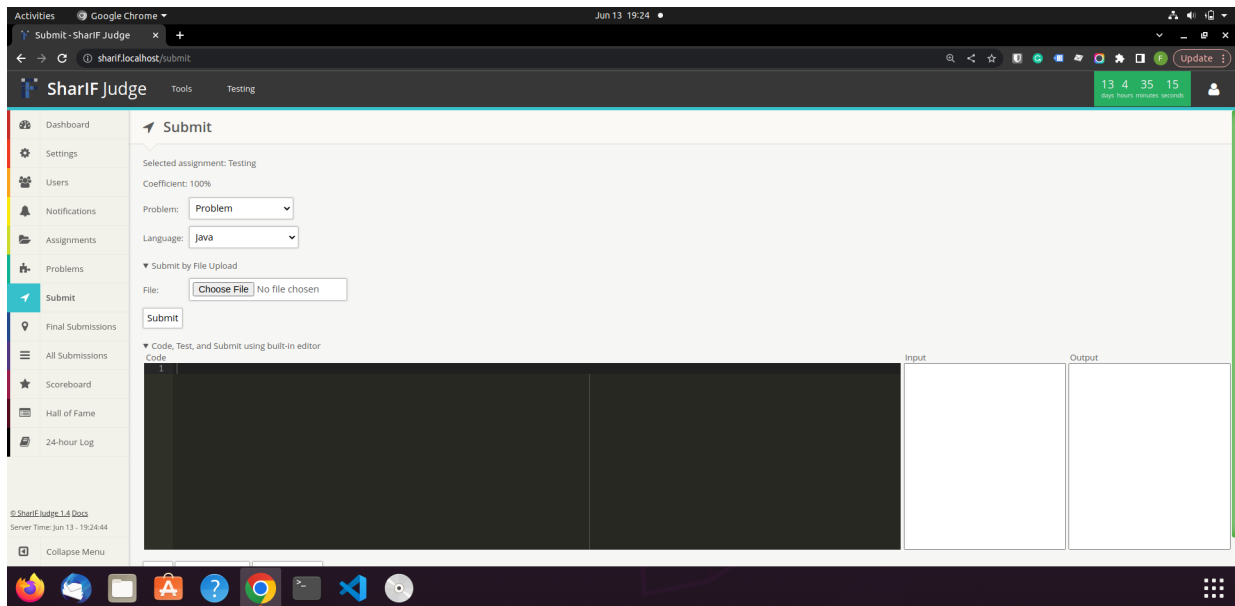
## 5 *Problems*



Gambar 3.7: Tampilan Halaman Problems

- 6 Gambar 3.7 merupakan tampilan halaman *problems* yang terdapat pada semua *role* pengguna.

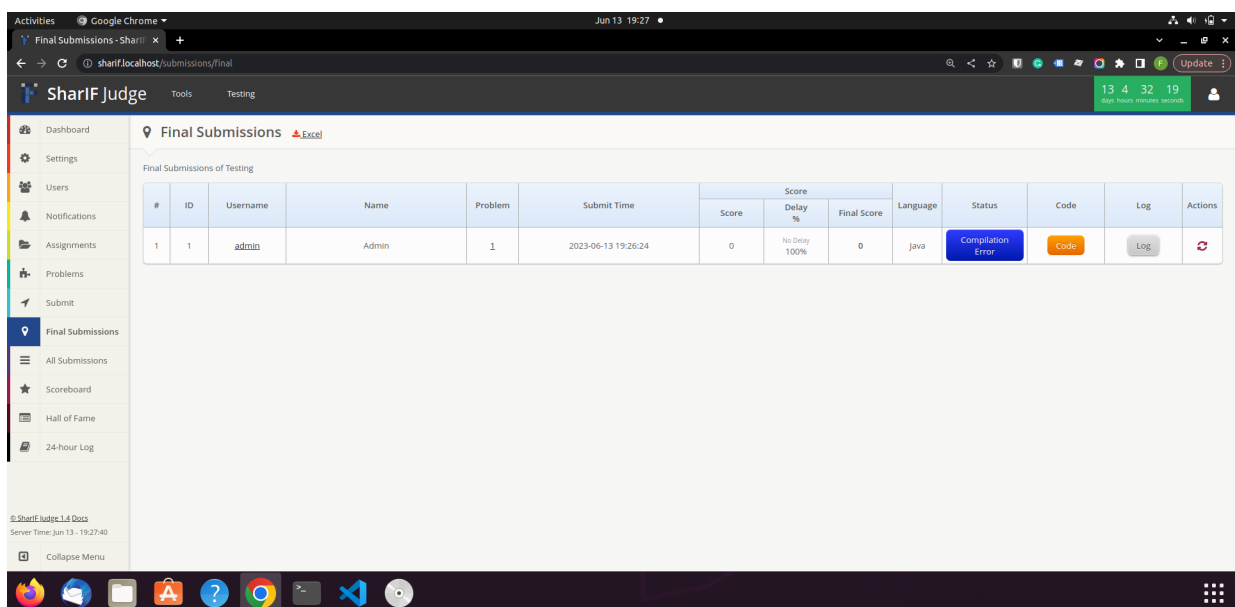
### 1 *Submit*



Gambar 3.8: Tampilan Halaman Submit

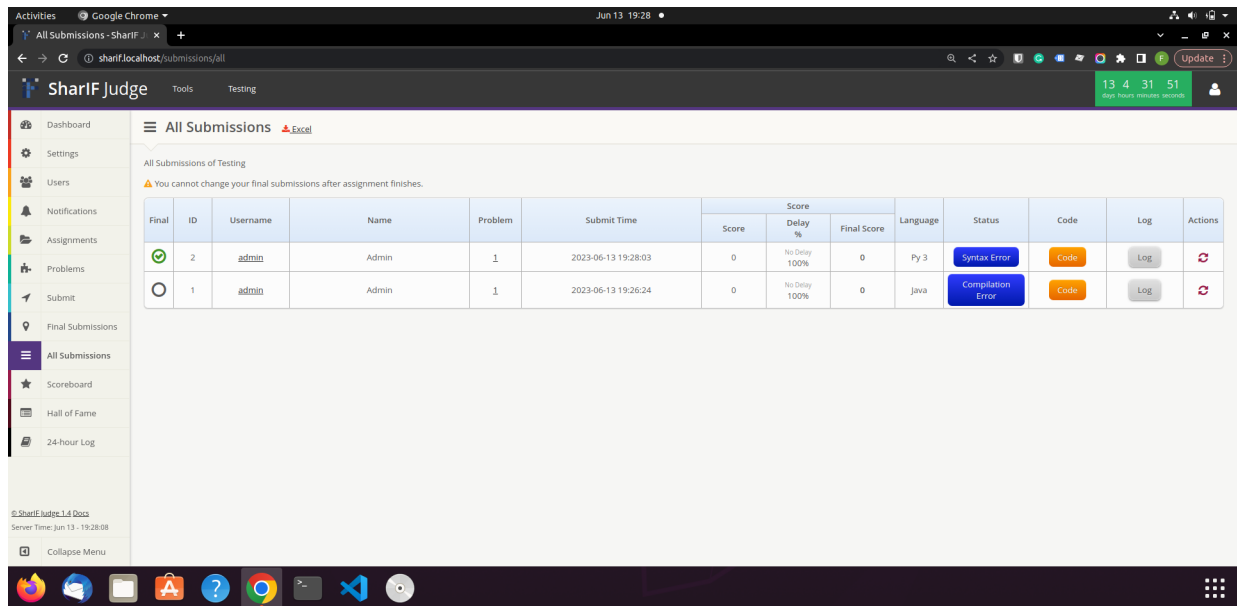
2 Gambar 3.8 merupakan tampilan halaman *submit* yang terdapat pada semua *role* pengguna.

### 3 *Final Submissions*



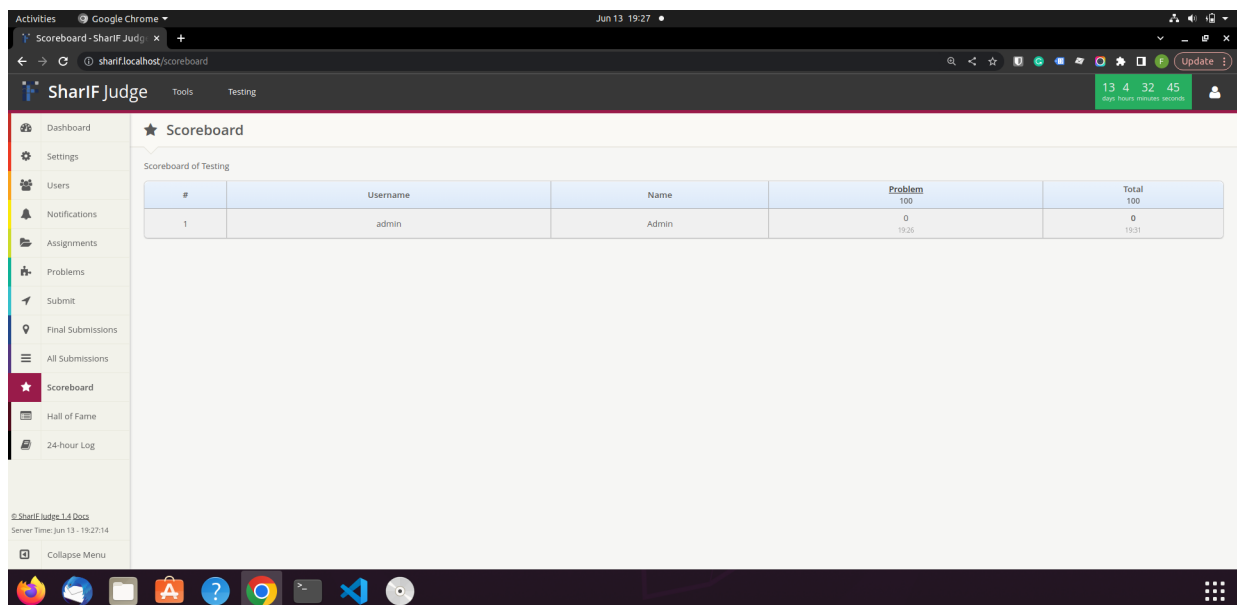
Gambar 3.9: Tampilan Halaman Final Submission

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

1 *All Submissions*

Gambar 3.10: Tampilan Halaman All Submission

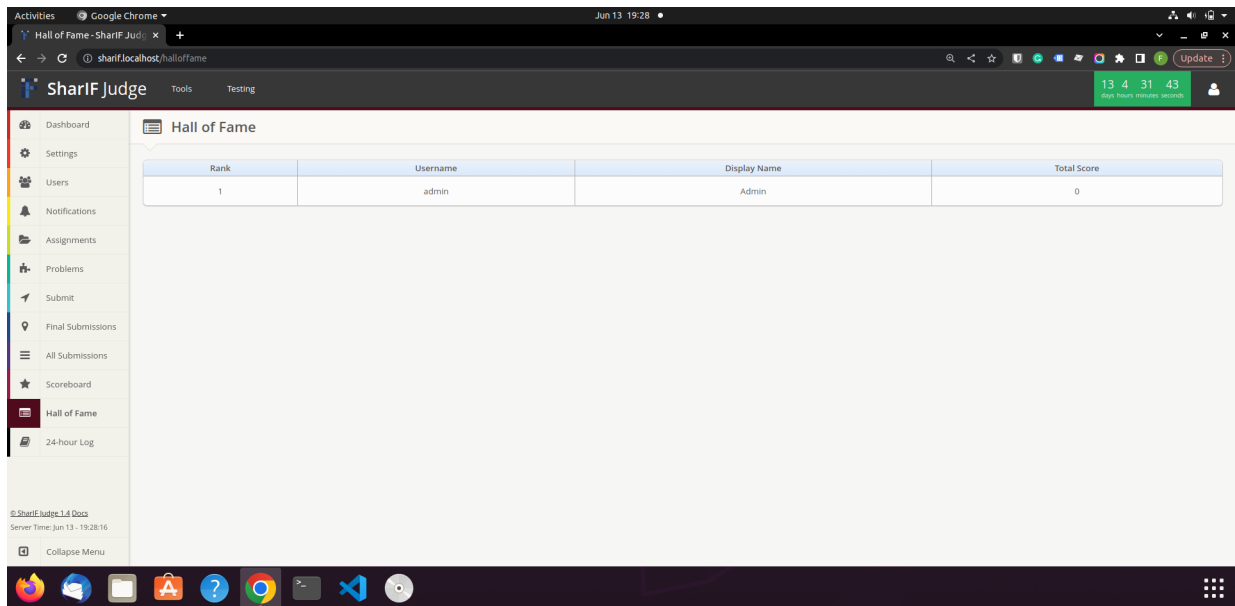
2 Gambar 3.10 merupakan tampilan halaman *All Submission* yang terdapat pada semua *role* pengguna.

3 *Scoreboard*

Gambar 3.11: Tampilan Halaman Scoreboard

4 Gambar 3.10 merupakan tampilan halaman *All Submission* yang terdapat pada semua *role* pengguna.

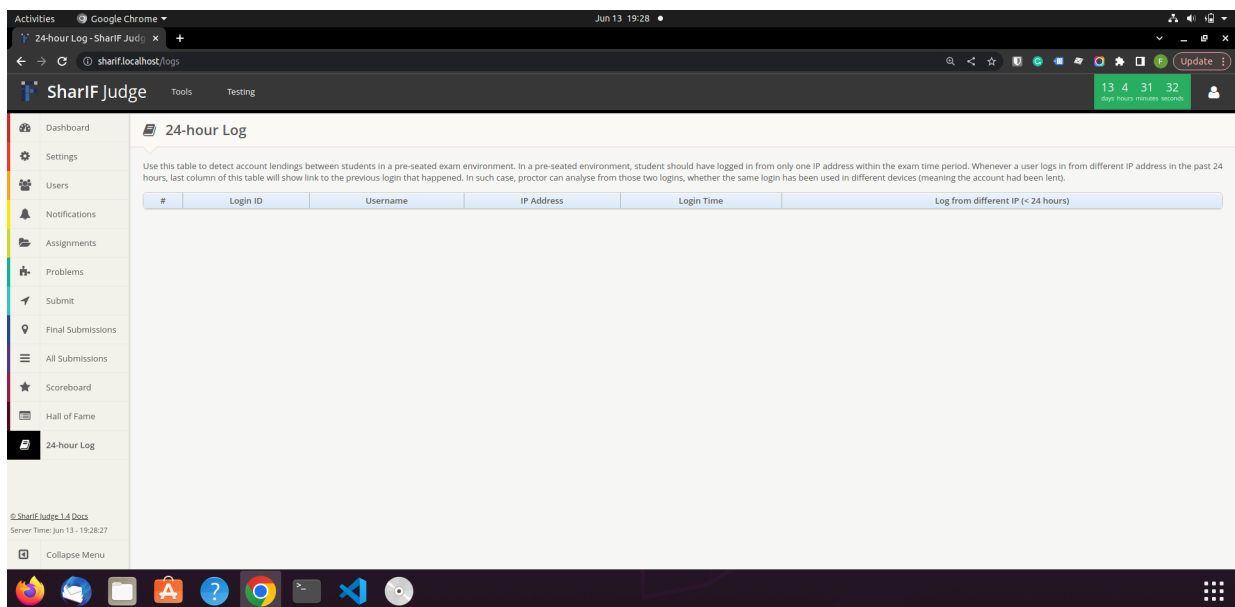
## 1 *Hall of Fame*



Gambar 3.12: Tampilan Halaman Hall of Fame

2 Gambar 3.12 merupakan tampilan halaman *Hall of Fame* yang terdapat pada semua *role* pengguna.

## 3 *24-hour Log*

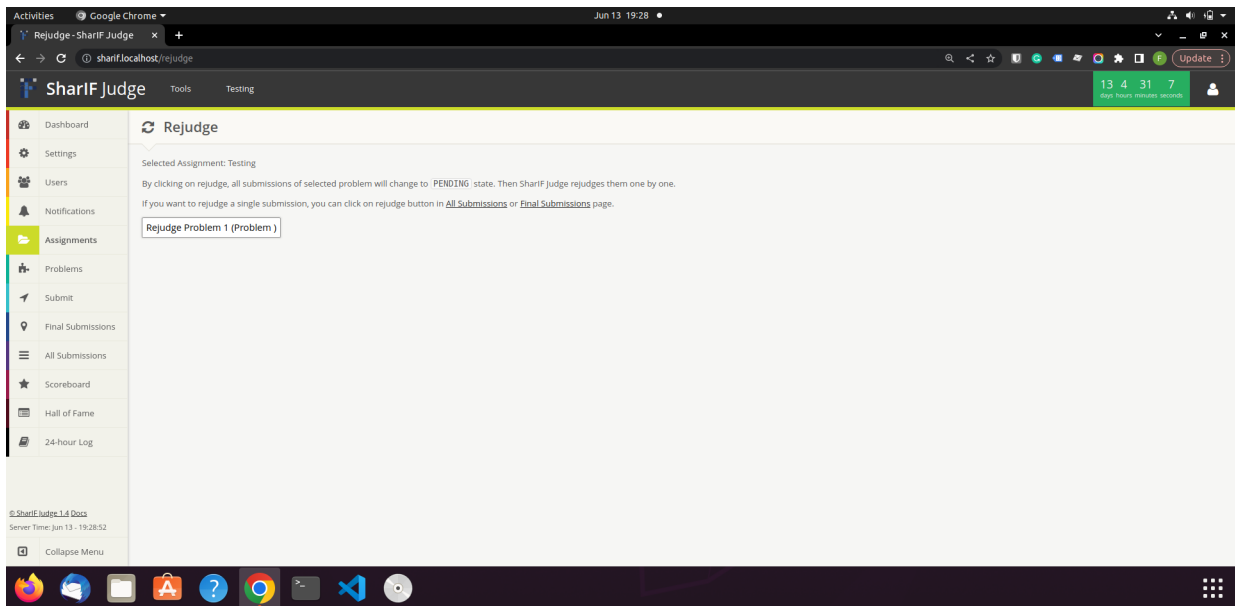


Gambar 3.13: Tampilan Halaman 24-hour Log

4 Gambar 3.13 merupakan tampilan halaman *24-hour Log* yang terdapat hanya pada *role admin* dan  
 5 *head instructor*.



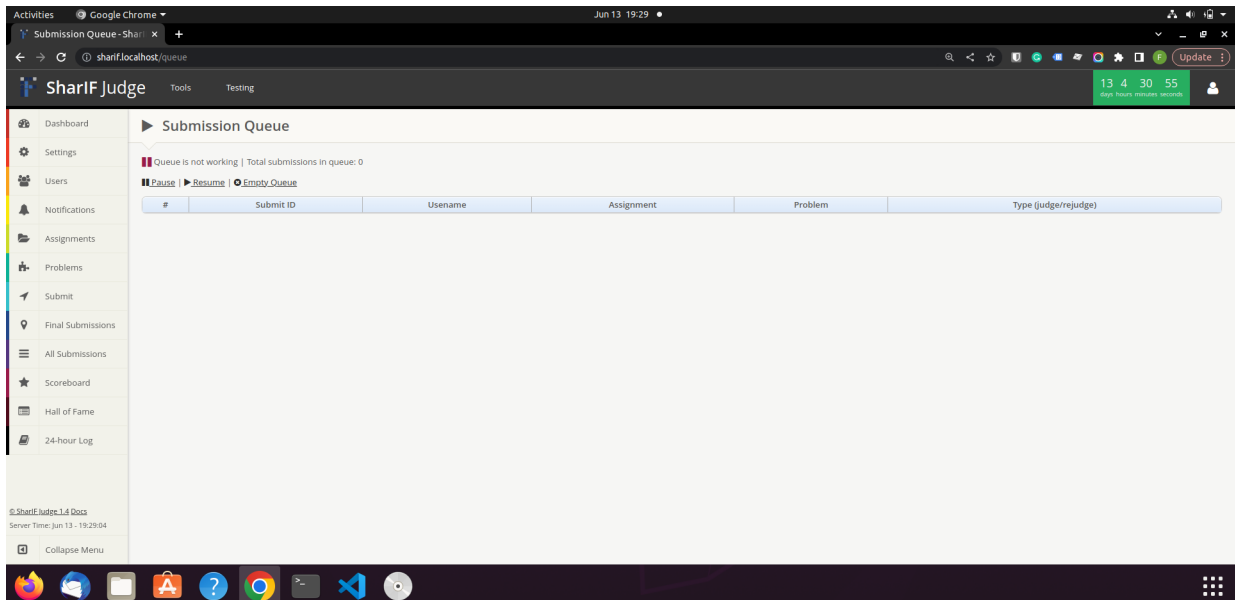
## 1 *Rejudge*



Gambar 3.14: Tampilan Halaman ReJudge

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

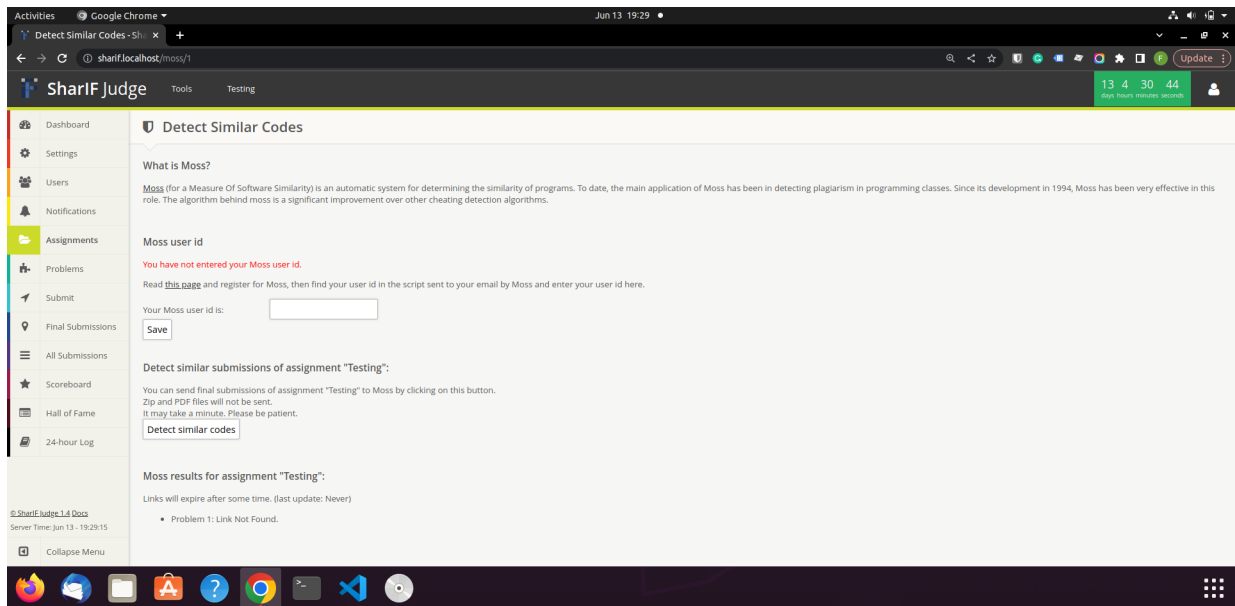
## 4 *Submission Queue*



Gambar 3.15: Tampilan Halaman Submission Queue

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

## 1 *Cheat Detection*



Gambar 3.16: Tampilan Halaman Cheat Detection

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

### 4 3.1.3 *Controller*

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

#### 8 **Assignments.php**

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

- 10 • **index**  
 11 Fungsi ini berguna untuk mengambil dan memberikan data menuju halaman `assignments.twig`  
 12 menggunakan `Assignment_model`.
- 13 • **select**  
 14 Fungsi ini berguna untuk memilih *assignment* menggunakan *ajax*.
- 15 • **pdf**  
 16 Fungsi ini berguna untuk mengunduh *assignment* atau *problem* dalam bentuk pdf.
- 17 • **downloadtestsdesc**  
 18 Fungsi ini berguna untuk mengunduh dan mengompres data *test* dan deskripsi sebuah  
 19 *assignment*.
- 20 • **download\_submissions**  
 21 Fungsi ini berguna untuk mengunduh dan mengompres kode terakhir sebuah *assignment*  
 22 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.

## Install.php

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

## Login.php

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

- **\_registration\_code**

Fungsi ini berguna untuk memeriksa kode registrasi.

- **index**

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

- **register**

Fungsi ini berguna untuk melakukan validasi dalam pembentukan akun.

- **logout**

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

- **lost**

Fungsi ini berguna untuk mengirim *email* lupa password.

- `rest`

Fungsi ini berguna untuk melakukan *reset password* pengguna.

### **Logs.php**

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

### **Moss.php**

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

- `index`

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

- `update`

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

- `_detec`

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

### **Notification.php**

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

- `index`

Fungsi ini berguna untuk mengambil dan memberikam data pada halaman *notifications* 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 meberikan 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.

## 1 **Scoreboard.php**

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

- 3 • **index**

4 Fungsi ini berguna untuk memberikan data pada halaman *scoreboard* menggunakan dua buah  
5 *model*. *Model* tersebut adalah **Assignment\_model** dan **Scoreboard\_model**.

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

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

## 9 **Settings.php**

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

- 11 • **index**

12 Fungsi ini berguna untuk mengambil dan memberikan data pada halaman *settings* menggu-  
13 nakan **Settings\_model** dan **Assignment\_model**.

- 14 • **update**

15 Fungsi ini berguna untuk mengambil masukan dan memperbaharui data pada halaman  
16 *settings* berdasarkan masukan tersebut. Data tersebut nantinya akan disimpan pada *database*  
17 menggunakan fungsi **Settings\_model**.

## 18 **Submission.php**

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

- 20 • **\_download\_excel**

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

- 23 • **final\_excel**

24 Fungsi ini berguna untuk mengunduh data *final submissions*.

- 25 • **all\_excel**

26 Fungsi ini berguna untuk mengunduh data seluruh *submissions*.

- 27 • **the\_final**

28 Fungsi ini berguna untuk memberikan data pada halaman *Final Submissions* menggunakan be-  
29 berapa *model*. *Model* tersebut terdiri dari **Submit\_model**, **Settings\_model**, dan **User\_model**.

- 30 • **all**

31 Fungsi ini berguna untuk memberikan data pada halaman *All Submissions* menggunakan be-  
32 berapa *model*. *Model* tersebut terdiri dari **Submit\_model**, **Settings\_model**, dan **User\_model**.

- 33 • **select**

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

- 36 • **\_check\_type**

37 Fungsi ini berguna untuk melakukan pengecekan tipe *submission* yang telah dikumpulkan  
38 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.

### 7 `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 memasukkannya 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.

### 35 `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*.

## 3.2 Library

*SharIF Judge* menggunakan beberapa *library* yang dibentuk secara manual maupun yang sudah tersedia pada *CodeIgniter 3*. Berikut merupakan *library* yang dipakai oleh *SharIF Judge*:

### Unzip

*Unzip* merupakan sebuah *library* yang dibentuk oleh Phil Sturgeon. *Library* ini mewajibkan pengguna untuk menyalakan *extension Zlib* sebelum dapat digunakan. *Library Unzip* berfungsi untuk mengekstraksi file dengan *extension .zip* menuju direktori yang ditentukan dan dapat mengeluarkan *error* yang sesuai. *Library* ini juga dapat memberi batasan *extension* apa yang diinginkan dari file tersebut. Kode 3.1 merupakan contoh penggunaan *library Unzip* pada *SharIF Judge*.

Kode 3.1: Contoh kode penggunaan *Library Unzip*

```
$this->load->library('unzip');
$this->unzip->allow(array('txt', 'cpp', 'html', 'md', 'pdf'));
$extract_result = $this->unzip->extract($u_data['full_path'], $tmp_dir);
```

Kode 3.1 merupakan contoh penggunaan *library Unzip*. Sintaks `$this->load->library('unzip');` berfungsi untuk melakukan *load library Unzip* agar dapat digunakan pada fungsi tersebut. Selanjutnya sintaks `$this->unzip->allow(array('txt', 'cpp', 'html', 'md', 'pdf'));` berfungsi untuk memberikan batasan *extension* apa saja yang diinginkan dari file tersebut. Terakhir sintaks `$extract_result = $this->unzip->extract($u_data['full_path'], $tmp_dir);` berfungsi untuk melakukan ekstraksi terhadap file zip pada `$u_data['full_path']` tersebut menuju direktori pada variabel `tmp_dir`.

### Twig

*Twig* merupakan sebuah *template engine library* yang digunakan untuk mempermudah dalam membentuk *view* pada aplikasi. *Twig* terintegrasi dengan fungsi-fungsi pada *CodeIgniter 3* sehingga dapat menggunakan seluruh fungsi yang terdapat pada *CodeIgniter 3*. Kode 3.2 merupakan contoh penggunaan *Twig* pada *SharIF Judge*.

Kode 3.2: Contoh *view* menggunakan *library Twig*

```
{% set title = 'Login' %}
{% include 'templates/simple_header.twig' %}
{{ form_open() }}
<div class="box login">
    <div class="judge_logo">
        <a href="{{ site_url() }}"></a>
```



```

19      </div>
20
31      <div class="login_form">
42          <div class="login1">
53              <p>
64                  <label for="form_username">Username</label><br/>
75                  <input id="form_username" type="text" name="username" required="required" pattern="[0-9a-z]{3,20}" title="The
8                      Username field must be between 3 and 20 characters in length, and contain only digits and lowercase
9                      letters" class="sharif_input" value="{{ set_value('username') }}" autofocus="autofocus"/>
106                  {{ form_error('username', '<div class="shj_error">', '</div>') }}
117              </p>
128              <p>
139                  <label for="form_password">Password</label><br/>
140                  <input id="form_password" type="password" name="password" required="required" pattern=".{6,200}" title="The
15                      Password field must be at least 6 characters in length" class="sharif_input"/>
161                  {{ form_error('password', '<div class="shj_error">', '</div>') }}
172              </p>
183              {% if error %}
194                  <div class="shj_error">Incorrect username or password.</div>
205              {% endif %}
216          </div>
227          <div class="login2">
238              <p style="margin:0;">
249                  {% if registration_enabled %}
250                      <a href="{{ site_url('register') }}">Register</a> |
261                  {% endif %}
272                      <a href="{{ site_url('login/lost') }}">Reset Password</a>
283                  <input type="submit" value="Login" id="sharif_submit"/>
294              </p>
305          </div>
316      </div>
327
338  </div>
349 </form>
350 </body>
361 </html>

```

Twig pada *view SharIF Judge* menggunakan dua buah *delimiters* yakni `{{ }}` dan `{% %}`. *Delimiters* `{{ }}` memiliki fungsi untuk mengembalikan *expression* seperti variabel ataupun fungsi *CodeIgniter 3*. Contoh fungsi yang dikembalikan oleh *delimiters* pada kode diatas adalah `form_open` yang merupakan sebuah fungsi pada *CodeIgniter 3* untuk membuka *tag form*. Sedangkan *delimiters* `{% %}` memiliki fungsi untuk mengeksekusi fungsi PHP seperti *for-loops* atau *if else*. Contoh fungsi yang dieksekusi pada kode diatas adalah `if` yang berfungsi untuk mengecek kondisi tertentu.

#### 44 Password\_hash

*Password\_hash* merupakan sebuah *library* yang dibentuk oleh *phpass*. *Library* ini berfungsi untuk melakukan enkripsi *password* dan melakukan verifikasi *password*. *Library* ini mendukung beberapa metode enkripsi antara lain *CRYPT\_BLOWFISH* dan *CRYPT\_EXT\_DES*. Kode 3.3 merupakan contoh penggunaan *library* ini pada *SharIF Judge*.

Kode 3.3: Contoh kode penggunaan *Library Password\_hash*

```

49
50 1      $this->load->library('password_hash', array(8, FALSE));
51 2      $user['password'] = $this->password_hash->HashPassword($this->input->post('password'));

```

Kode 3.3 merupakan contoh penggunaan *library Passwords\_hash*. Sintaks pada baris pertama akan melakukan *load* pada *library Password\_hash*. Sintaks pada baris selanjutnya akan melakukan *hashing* pada *input* menggunakan algoritma yang ditentukan dan menyimpannya pada sebuah variabel.

### 1 *MY\_Form\_validation*

2 *MY\_Form\_validation* merupakan *library* yang dibentuk secara manual untuk menambahkan fungsi  
3 validasi yang sudah tersedia pada *CodeIgniter 3*. *Library* ini memiliki dua buah fungsi yakni:

- 4 • **required**  
5 Fungsi ini berguna untuk melakukan pengecekan terhadap sebuah *input* apakah berisikan  
6 sebuah *array* kosong ataupun *string* kosong.
- 7 • **lowercase**  
8 Fungsi ini berguna untuk melakukan pengecekan apakah *input* berisikan kata-kata dengan  
9 huruf kecil atau tidak.

### 10 *MY\_Profiler*

#### 11 *Parsedown*

12 *Parsedown* merupakan sebuah *library* yang dibentuk oleh Emanuil Rusev. *Library* ini berfungsi  
13 untuk mengubah teks dengan sintaks *markdown* menjadi teks dalam bentuk file lain seperti HTML.  
14 Kode 3.4 merupakan contoh penggunaan *library parsedown*.

Kode 3.4: Contoh kode penggunaan *Library Parsedown*

```
15  
16 1 $this->load->library('parsedown');  
17 2 $html = $this->parsedown->parse(file_get_contents("$assignment_dir/p$i/desc.md"));
```

19 Kode 3.4 akan menginisiasi *library parsedown* pada sintaks baris pertama. Selanjutnya isi dari  
20 file *desc.md* akan diambil dan dilakukan *parsedown* menjadi file HTML. Berikut merupakan contoh  
21 dari teks sebelum dan sesudah di *parsedown*:

22 Ini adalah list :

- 23 1. Satu
- 24 2. Dua
- 25 3. Tiga

Kode 3.5: Contoh kode sesudah dilakukan *parsedown*

```
26  
27 1 <p>Ini adalah list</p>  
28 2 <ol>  
29 3 <li>Satu</li>  
30 4 <li>Dua</li>  
31 5 <li>Tiga</li>
```

33 Kode 3.5 merupakan contoh kode HTML sesudah dilakukan *parsedown*. Angka 1, 2, dan 3 akan  
34 diubah menjadi *list* yang dapat ditampilkan pada HTML.

### 35 *Phpexcel*

36 *Phpexcel* merupakan sebuah *library* yang dibentuk oleh *PHPExcel*. *Library* ini memiliki fungsi untuk  
37 mengubah data yang ada pada PHP menjadi file excel.

### 38 *Shj\_pagination*

39 *Shj\_pagination* merupakan *library* yang dibentuk secara manual dengan fungsi untuk membatasi  
40 maksimal data pada setiap halaman sesuai dengan konfigurasinya.

#### ***upload***

*Library* ini merupakan fungsi yang tersedia pada *CodeIgniter 3*. *Library* ini berguna untuk menerima masukan dan mengunggah file yang dimasukan oleh pengguna.

#### ***email***

*Library* ini merupakan fungsi yang tersedia pada *CodeIgniter 3*. *Library* ini berguna untuk mengirimkan *email* kepada orang yang dituju sesuai dengan konfigurasinya.

#### ***form\_validation***

*Library* ini merupakan fungsi yang tersedia pada *CodeIgniter 3*. *Library* ini berguna untuk melakukan validasi terhadap data yang dimasukan oleh pengguna sesuai dengan aturan yang telah ditetapkan.

### **3.3 Analisis Sistem Usulan**

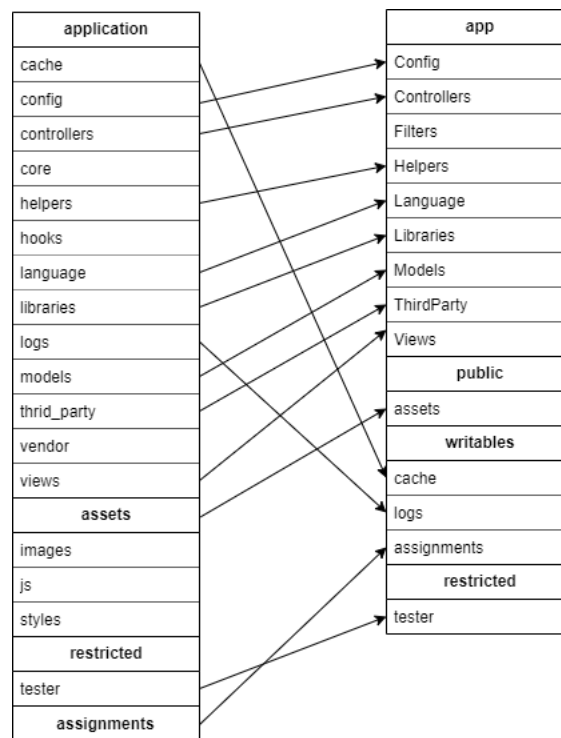
Konversi *CodeIgniter 3* menuju *CodeIgniter 4* diperlukan penulisan ulang karena terdapat perubahan struktur aplikasi dan beberapa fungsi yang memiliki pemanggilan berbeda dan harus dilakukan pembaharuan.

#### **3.3.1 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*.

#### **3.3.2 Struktur Aplikasi**

Struktur aplikasi pada *CodeIgniter 3* dan *CodeIgniter 4* memiliki perubahan sehingga perlu dilakukan pemindahan *file-file* menuju *CodeIgniter 4*. Gambar 3.17 merupakan pemindahan struktur aplikasi *SharIF Judge* pada *CodeIgniter 3* menuju *CodeIgniter 4*.



Gambar 3.17: 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 **3.3.3 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 fungsi dan metodenya.

### 19 **3.3.4 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 3.6 merupakan contoh fungsi yang digunakan untuk  
 30 mengambil data dari sebuah tabel.

Kode 3.6: 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 3.6 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 melakukan inisiasi dan 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 3.7: Contoh kode untuk menghapus data pada model

```
1 $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*. Seluruh *delimiters* akan diubah menggunakan fitur yang terdapat pada *CodeIgniter 4*. Kode 3.8 merupakan contoh konversi yang dilakukan.

### Kode 3.8: Contoh *view* menggunakan *twig*

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

menjadi kode berikut:

### Kode 3.9: Contoh *view* menggunakan *php*

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

Seluruh sintaks *twig* akan diubah menjadi sintaks *PHP* dari *CodeIgniter 4* seperti yang terdapat pada kode 3.9.

### Controller

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

Kode 3.10: Contoh penggunaan *initController* untuk variabel

```

12 protected $config;
13
14
15 public function initController(RequestInterface $request, ResponseInterface $response, LoggerInterface $logger)
16 {
17     // Do Not Edit This Line
18     parent::initController($request, $response, $logger);
19
20     // Preload any models, libraries, etc, here.
21
22     // E.g.: $this->session = \Config\Services::session();
23     $this->config = Config('Secrets');
24 }

```

Kode 3.10 akan memanggil variabel *\$config* berisikan data dari *file config Secrets.php* yang dapat digunakan seluruh *controller*.

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

### 3.3.5 Libraries

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

#### Emails

*Emails* pada *CodeIgniter 4* terdapat perubahan sintaks dan cara pemanggilan sehingga akan dipindahkan sesuai dengan sintaks yang baru. Sintaks berubah dari yang sebelumnya menggunakan *snakecase* menjadi menggunakan *camelcase*. Kode 4.6 merupakan contoh penggunaan *library email*.

Kode 3.11: Contoh perubahan *library emails*

```

38 $this->email->setFrom($this->settings_model->get_setting('mail_from'), $this->settings_model->get_setting('mail_from_name'));
39
40 $this->email->setTo($user[1]);
41
42 $this->email->setSubject('SharIF Judge Username and Password');
43
44 $text = $this->settings_model->get_setting('add_user_mail');
45
46 $text = str_replace('{SITE_URL}', base_url(), $text);
47
48 $text = str_replace('{ROLE}', $user[4], $text);
49
50 $text = str_replace('{USERNAME}', $user[0], $text);
51
52 $text = str_replace('{PASSWORD}', htmlspecialchars($user[3]), $text);
53
54 $text = str_replace('{LOGIN_URL}', base_url(), $text);
55
56 $this->email->setMessage($text);
57
58 $this->email->send()

```

Kode 4.6 memiliki sintaks dengan nama sama namun terdapat perubahan menjadi *camelcase*.

### **Working with Uploaded Files**

*Upload* akan digantikan dengan fungsi *Working with oploaded files* dengan beberapa penggantian dan penghapusan fungsi. *Working with uploaded files* terdapat perubahan pada beberapa sintaks dan validasi terhadap *file* yang telah diunggah. Konversi aplikasi *SharIF Judge* akan menggunakan fungsi ini dengan beberapa perubahan sintaks sesuai dengan dokumentasinya.

### **Validation**

*Form\_validation* akan digantikan dengan fungsi *validation* dengan perubahan dan penghapusan beberapa fungsi. Berikut merupakan contoh pembentukan aturan untuk mengumpulkan sebuah data pada *form*.

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

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

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

Sintaks diatas akan mengembalikan *error* terhadap *form* dengan nama *username* apabila tidak sesuai dengan aturan yang sudah ditentukan. Variabel *validation* akan dikirimkan dari *controller* berisikan *library* dari *validation* tersebut.

### **Zip Archive**

*Zip Encoding* akan digantikan dengan fungsi PHP *zip archive* karena sudah tidak tersedia pada *CodeIgniter 4*. Fungsi *zip archive* terdapat beberapa perbedaan sehingga akan disesuaikan dengan fungsi-fungsi yang ada.

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

### **Twig**

*Library* ini tidak akan digunakan untuk membentuk *view* pada *CodeIgniter 4* namun, akan ada penggunaan sebuah fungsi *Twig* yang akan dibentuk pada direktori *app/Libraries*. Fungsi tersebut bernama *extra\_time\_formatter* yang memiliki fungsi untuk mengubah input yang diberikan menjadi format jam dikali enam puluh menit.

### **Unzip**

*Library* ini akan digunakan kembali dan dipindahkan menuju direktori *app/Libraries*. *Library* terdapat penghapusan sintaks *defined* dan juga penambahan *namespace*.



### 1 *Password\_hash*

2 *Library* ini tidak akan digunakan dan akan digantikan oleh *password hash* yang disediakan oleh  
 3 PHP. *Library Password\_hash* merekomendasikan pengguna untuk menggunakan fungsi *native* yang  
 4 disediakan oleh PHP apabila aplikasi mendukung PHP versi 5.5 ke atas. Sehingga, akan dilakukan  
 5 konversi menggunakan fungsi yang disediakan oleh PHP bernama `password_hash()`. Seluruh  
 6 penggunaan *library* ini akan diubah menggunakan fungsi yang disediakan oleh PHP dengan metode  
 7 *hashing* sama yaitu *CRYPT\_BLOWFISH*. Perubahan fungsi *hashing* ini bersifat *backward compatible*  
 8 sehingga dapat menggunakan *database* aplikasi terdahulu tanpa perlu membentuk data baru. Berikut  
 9 merupakan contoh pengubahan kode dari *phpass* menjadi *password\_hash*.

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

11 menjadi

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

13 Sintaks `password_hash()` diatas menerima dua buah parameter yakni data yang ingin di enkripsi  
 14 dan tipe enkripsi. Enkripsi akan menggunakan sintaks `PASSWORD_BCRYPT` yang menggunakan tipe  
 15 *hash* berupa *CRYPT\_BLOWFISH*.

### 16 *MY\_Form\_validation*

17 *Library MY\_Form\_validation* akan dipindahkan menuju direktori `app/Libraries`. *Library* ini  
 18 akan digunakan kembali dengan perubahan *extends* menjadi menuju `Validation`, penghapusan  
 19 sintaks `defined`, dan akan ada penambahan *namespace* pada baris awal file. Kode 4.16 merupakan  
 20 contoh penambahan *namespace* dan penggantian *extends* pada *library* ini.

Kode 3.12: Contoh perubahan *library MY\_Form\_validation* pada *CodeIgniter 4*

```
21 namespace App\Libraries;  
22 1  
23 2  
24 3 use CodeIgniter\Validation\Validation;  
25 4  
26 5 class MY_Form_validation extends Validation
```

28 Kode 4.16 menghapus sintaks `defined` dan menggantikannya dengan penambahan *namespace*. Selain  
 29 itu, kelas *library* akan *extends Validation*.

### 30 *MY\_Profiler*

#### 31 *Parsedown*

32 *Library Parsedown* akan dipindahkan menuju direktori `app/Libraries`. *Library* ini akan digunakan  
 33 kembali dengan penambahan *namespace* pada baris awal file dan penghapusan sintaks `defined`.  
 34 Kode 4.17 merupakan contoh penambahan *namespace* dan juga penambahan sintaks *defined*.

Kode 3.13: Contoh perubahan *library Parsedown* pada *CodeIgniter 4*

```
35 namespace App\Libraries;  
36 1  
37 2  
38 3 class Parsedown
```

40 Kode 4.17 menghapus sintaks `defined` dan menggantikannya dengan penambahan *namespace*.

### 1 *Phpexcel*

2 *Library* ini akan digunakan kembali namun tidak akan dipindahkan menuju `app/Libraries`. *Library*  
3 akan dilakukan instalasi melalui *composer* dengan sintaks berikut:

```
4 composer require phpooffice/phpexcel
```

5 Sintaks diatas akan dijalankan pada akar dari aplikasi dan tidak terdapat perubahan terhadap  
6 penggunaan sintaks ini.

### 7 *Shj\_pagination*

8 *Library* ini akan digunakan kembali dan dipindahkan menuju direktori `app/Libraries`. Selain itu,  
9 terdapat penambahan *namespace* pada baris awal file dan penghapusan sintaks *defined*.

## 10 3.3.6 *Configuration*

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

- 15 • `$config['csrf_protection'] = TRUE;`
- 16 • `$config['csrf_token_name'] = 'shj_csrf_token';`
- 17 • `$config['csrf_cookie_name'] = 'shjcsrftoken';`
- 18 • `$config['csrf_expire'] = 7200;`
- 19 • `$config['csrf_regenerate'] = FALSE;`

20 *Configurations* yang telah dipindahkan akan diubah dari yang sebelumnya menggunakan *array*  
21 menjadi menggunakan *variable*. Seluruh *configurations* pada *CodeIgniter 3* akan dipindahkan  
22 menuju *CodeIgniter 4* sesuai dengan direktorinya dan fungsinya. Sedangkan `app/Config/App.php`  
23 dan dipindahkan menuju file `.env` karena alasan kemudahan untuk penggantian *environment* dalam  
24 melakukan *deploy*.

## 25 3.3.7 *Database*

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

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

```
35 $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`.

### 3.3.8 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*.



## BAB 4

### PERANCANGAN

Bab ini membahas perancangan untuk seluruh implementasi *SharIF Judge* pada *CodeIgniter 4*.

#### 4.1 Instalasi *CodeIgniter 4*

*CodeIgniter 4* akan dilakukan instalasi menggunakan *composer*. *Composer* merupakan sebuah *dependency manager* untuk PHP yang memungkinkan pengguna untuk melakukan instalasi seluruh kebutuhan untuk menjalankan program berbasis PHP. Instalasi akan dilakukan menggunakan kode sebagai berikut:

#### 4.2 Perubahan Struktur Aplikasi

Struktur aplikasi *SharIF Judge* akan dipindahkan seperti pemetaan pada bab ?? gambar 3.17.

Struktur aplikasi pada *CodeIgniter 4* akan berisikan sebagai berikut :

##### 4.2.1 app/Config

File *config* pada *CodeIgniter 3* akan dipindahkan sesuai dengan pemetaan pada gambar. Direktori berisikan data-data pada *application/Config*. Beberapa data pada direktori ini akan dipindahkan menuju file yang terdapat pada *CodeIgniter 4*. Terdapat juga penambahan file *Secrets.php* yang dibentuk secara manual. Berikut merupakan rincian isi direktori ini:

##### App.php

File ini tidak akan digunakan sehingga akan dibiarkan kosong dan seluruh data akan dipindahkan menuju file *.env*. Kode 4.1 merupakan isi dari file *.env* yang dipindahkan dari direktori *application/config*:

Kode 4.1: Kode *application/config/App.php* yang dipindahkan menuju *.env*

```
app.baseURL = 'http://sharif.localhost/'
```

Kode 4.1 akan menentukan *url* dasar dari aplikasi menjadi *http://sharif.localhost/*.

##### Autoload.php

File ini tidak akan digunakan karena pada *CodeIgniter 3* tidak mengikuti standar pada PSR-4. Sedangkan pada *CodeIgniter 4* mengikuti standar PSR-4 sehingga dapat melakukan inisiasi terhadap

1 sebuah kelas menggunakan sintaks `new Kelas`. Beberapa kelas yang diinisiasi pada file ini akan  
2 dipindahkan menuju `BaseController` dan juga akan dipanggil menggunakan PSR-4 pada file-file  
3 yang menggunakan kelas tersebut.

#### 4 `Cache.php`

5 File ini tidak terdapat perubahan karena akan tetap menggunakan konfigurasi *default*.

#### 6 `Constant.php`

7 File ini akan berisikan seluruh data yang dipindahkan dari `application/config/constants.php`.  
8 Kode .

#### 9 `Filters.php`

### 10 4.2.2 *Controllers*

11 *Controller* terdapat perubahan pada bagian `__construct()` dimana sekarang tidak dapat mengem-  
12 balikan sesuatu. Oleh karena itu, akan dibentuk beberapa *filters* 4.2.1 untuk melakukan pengecekan  
13 terhadap fungsi yang sebelumnya terdapat pada `__construct()`.

#### 14 `app/Controllers`

15 Direktori ini berisikan seluruh *controller* yang dipindahkan dari *CodeIgniter 3*. Berikut merupakan  
16 rincian isi pada direktori ini:

- 17 • `Assignments.php`
- 18 • `BaseController.php`
- 19 • `Dashboard.php`
- 20 • `Halloffame.php`
- 21 • `Install.php`
- 22 • `Login.php`
- 23 • `Logs.php`
- 24 • `Moss.php`
- 25 • `Notifications.php`
- 26 • `Problems.php`
- 27 • `Profile.php`
- 28 • `Queue.php`
- 29 • `Queueprocess.php`
- 30 • `Rejudge.php`
- 31 • `Scoreboard.php`
- 32 • `Server_time.php`
- 33 • `Settings.php`
- 34 • `Submissions.php`
- 35 • `Submit.php`
- 36 • `Users.php`

*Controllers* terdapat perubahan dan penambahan baik dalam *extends* maupun dalam pemanggilan kelas lain seperti *model*. Kode 4.2 merupakan perubahan yang terdapat pada *controller Logs.php*.

Kode 4.2: Perubahan kode *controllers* pada *CodeIgniter 4*

```

4 namespace App\Controllers;
5
6
7 use App\Controllers\BaseController;
8 use App\Models\AssignmentModel;
9 use App\Models\LogsModel;
10 use App\Models\User;
11
12 class Logs extends BaseController
13 {
14     protected $session;
15     protected $user;
16     protected $logs_model;
17     protected $assignment_model;
18
19     public function __construct()
20     {
21         $this->session = session();
22         $this->logs_model = new LogsModel();
23         $this->assignment_model = new AssignmentModel();
24         $this->user = new User();
25         if ( $this->user->level <= 2 ) // permission denied
26             throw \CodeIgniter\Exceptions\PageNotFoundException::forPageNotFound();
27     }
28
29     public function index()
30     {
31
32         $data = array(
33             'logs' => $this->logs_model->get_all_logs(),
34             'selected' => 'logs',
35             'user' => $this->user,
36             'all_assignments' => $this->assignment_model->all_assignments(),
37             'finish_time' => $this->user->selected_assignment['finish_time'],
38             'extra_time' => $this->user->selected_assignment['extra_time'],
39         );
40
41         return view('pages/admin/logs', $data);
42     }
43 }

```

Kode 4.2 terdapat perubahan dimana sekarang akan *extends BaseController*. Terdapat juga penghapusan sintaks `defined('BASEPATH')` OR `exit('No direct script access allowed')`; . Terdapat penambahan sintaks *namespace* dan juga beberapa sintaks untuk memanggil *models*. *Controller* juga memiliki perubahan dalam mengembalikan *view* dimana sekarang menggunakan sintaks `return view`. Selain itu terdapat penambahan pada *BaseController.php* untuk melakukan inisiasi terhadap *helpers* dan juga beberapa *library* yang akan digunakan.

### 4.2.3 Filters

Pada *CodeIgniter 4* `__construct()` tidak dapat mengembalikan sesuatu oleh karena itu akan dibentuk beberapa *filters* untuk melakukan pengecekan. Beberapa *filters* yang dibentuk antara lain berfungsi untuk mengecek apakah dijalankan dari *command line interface*, apakah sudah *install* dan *login*, apakah sudah *login*, apakah sudah *login* dan dijalankan dari *command line interface*, apakah sudah *login* dan apakah *request* berupa *AJAX*, dan apakah sudah *login* dan pengecekan terhadap *role* pengguna. Kode 4.3 merupakan sintaks untuk melakukan pengecekan apakah dijalankan dari *command line interface*.

Kode 4.3: Pemindahan kode pada *Filters*

```

11 public function before(RequestInterface $request, $arguments = null)
12 {
13     $request = \Config\Services::request();
14
15     if ($request->isCLI())
16         throw \CodeIgniter\Exceptions\PageNotFoundException::forPageNotFound();
17 }

```

Kode 4.3 merupakan pemindahan kode dari `__construct` menuju `filters CheckCLI`. Kode ini akan mengecek apakah `request` yang diberikan oleh pengguna merupakan *command line interface*. Apabila `request` yang diberikan bukan berupa itu maka akan diberikan error berupa halaman tidak ditemukan. Setelah dibentuk `filters`, selanjutnya akan ditambahkan menuju file `Filters.php` untuk mendefinisikan nama untuk dimasukkan menuju `routes`. Kode 4.4 merupakan penambahan nama pada file `Filters.php`.

Kode 4.4: Penambahan nama *filters* untuk didefinisikan menuju *routes*

```

15 public array $aliases = [
16     'csrf' => CSRF::class,
17     'toolbar' => DebugToolbar::class,
18     'honeypot' => HoneyPot::class,
19     'invalidchars' => InvalidChars::class,
20     'secureheaders' => SecureHeaders::class,
21     'check-installandlogin' => CheckInstallAndLogin::class,
22 ]

```

Kode 4.4 merupakan penambahan nama *filters* untuk dipanggil menuju *routes*. Penamaan ini akan ditambahkan pada array `$aliases` dengan *index* dan nama kelasnya. Setelah ditambahkan, *filters* akan dipanggil pada *routes* yang membutuhkan pengecekan. Kode 4.5 merupakan penambahan *filters* pada *routes* sesuai dengan kebutuhannya.

Kode 4.5: Penambahan *filter* pada *routes*

```

29 $routes->get('/settings', 'Settings::index', ['filter' => 'check-loginandlevelAdmin:dual,noreturn']);
30

```

Kode 4.5 menambahkan *filter* yang sudah dibentuk dan dinamakan setelah penulisan nama *controller* dan fungsinya.

#### 4.2.4 Libraries

*Libraries* terdapat beberapa perubahan dan penghapusan fungsi sehingga akan digantikan. Berikut merupakan perancangan perubahan fungsi pada *CodeIgniter 4*.

##### Emails

*Emails* pada *CodeIgnier 4* terdapat perubahan sintaks dan cara pemanggilan sehingga akan dipindahkan sesuai dengan sintaks yang baru. Sintaks berubah dari yang sebelumnya menggunakan *snakecase* menjadi menggunakan *camelcase*. Kode 4.6 merupakan contoh penggunaan *library email*.

Kode 4.6: Contoh perubahan *library emails*

```

41 $this->email->setFrom($this->settings_model->get_setting('mail_from'), $this->settings_model->get_setting('mail_from_name'));
42     $this->email->setTo($user[1]);
43     $this->email->setSubject('SharIF Judge Username and Password');
44     $text = $this->settings_model->get_setting('add_user_mail');
45     $text = str_replace('{SITE_URL}', base_url(), $text);
46     $text = str_replace('{ROLE}', $user[4], $text);
47     $text = str_replace('{USERNAME}', $user[0], $text);
48     $text = str_replace('{PASSWORD}', htmlspecialchars($user[3]), $text);
49     $text = str_replace('{LOGIN_URL}', base_url(), $text);
50     $this->email->setMessage($text);
51     $this->email->send()

```



Kode 4.6 memiliki sintaks dengan nama sama namun terdapat perubahan menjadi *camelcase*.

### Working with Uploaded Files

*Working with uploaded files* terdapat perubahan pada beberapa sintaks dan validasi terhadap *file* yang telah diunggah. Konversi aplikasi *SharIF Judge* akan menggunakan fungsi ini dengan beberapa perubahan sintaks sesuai dengan dokumentasinya. Kode 4.7 merupakan perubahan yang terdapat pada *library* ini.

Kode 4.7: Perancangan perubahan *library upload* pada *CodeIgniter 4*

```

7
81 $zip_uploaded = $this->request->getFile('tests_desc');
92 if ( $_FILES['tests_desc']['error'] === UPLOAD_ERR_NO_FILE ){
103     $this->messages[] = array(
114         'type' => 'notice',
125         'text' => "Notice: You did not upload any zip file for tests. If needed, upload by editing assignment."
136     );
147 }
158 elseif ( !$zip_uploaded){
169     $this->messages[] = array(
170         'type' => 'error',
181         'text' => "Error: Error uploading tests zip file: ".$zip_uploaded->getErrorString()
192     );
203 }
214 else{
225     $zip_uploaded->move($assignments_root);
236     $this->messages[] = array(
247         'type' => 'success',
258         'text' => "Tests (zip file) uploaded successfully."
269     );
270 }

```

Kode 4.7 merupakan perubahan yang terdapat pada *library upload*. . Pengambilan file akan digantikan dengan sintaks `$this->request->getFile('')` dengan parameter berupa nama dari *tag form* yang sudah dibentuk. Selanjutnya akan dilakukan pengecekan terhadap file yang sudah di unggah dan memberikan beberapa *error message* sesuai dengan kondisinya. File yang sudah di unggah akan dipindahkan menuju direktori `assignments_root`.

### Validation

*Form\_validation* akan digantikan dengan fungsi *validation* dengan perubahan dan penghapusan beberapa fungsi. *Validation* akan diinisiasikan pada fungsi `__construct` pada setiap *controller* yang membutuhkan. Kode 4.8 merupakan inisiasi *validation* pada *controller*.

Kode 4.8: Perancangan inisiasi *validation* pada `__construct`

```

38
39 1
40 2 protected $validation;
41 3
42 4     public function __construct()
43 5     {
44 6         $this->validation = \Config\Services::validation();
45 7     }

```

Kode 4.8 merupakan sintaks untuk melakukan inisiasi terhadap *validation*. Setiap *controller* yang menggunakan *validation* akan dideklarasikan sebuah variabel bernama *validation* agar dapat dipanggil pada seluruh fungsi yang membutuhkan pada kelas tersebut. *Validation* dilanjutkan dengan penetapan aturan untuk *tag form* yang diinginkan. Berikut merupakan contoh pembentukan aturan untuk mengumpulkan sebuah data pada *form*.

Kode 4.9: Perancangan perubahan konfigurasi aturan pada *library validation*

```

1
31 $this->validation->setRule('username', 'username', 'required|min\length[3]|max\length[20]);

```

Kode 4.9 akan menetapkan aturan terhadap *input* yang akan masukan oleh pengguna sesuai dengan nama *formnya*. Sedangkan penetapan aturan berubah menggunakan *camelCase*. Aturan yang dibentuk secara manual akan dipindahkan menuju file **Validation.php**. Kode 4.10 merupakan aturan yang dibentuk secara manual.

Kode 4.10: Perancangan aturan yang dibentuk secara manual pada file **Validation.php**

```

8
91 class MyRules
102 {
113     public function password_check($str): bool
124     {
135         if (strlen($str) == 0 OR (strlen($str) >= 6 && strlen($str) <= 200))
146             return TRUE;
157         return FALSE;
168     }
179
180     public function password_again_check($str) :bool
191     {
202         $request = \Config\Services::request();
213         if ($request->getPost('password') != $request->getPost('password_again'))
224             return FALSE;
235         return TRUE;
246     }
257
268     public function role_check($user,$role)
279     {
280         $user = new User();
291         if ($user->level <= 2)
302             return ($role == '');
313
324         // Admins can change everybody's user role:
335         $roles = array('admin', 'head_instructor', 'instructor', 'student');
346         return in_array($role, $roles);
357     }
368
379     /**
380      * Checks whether a user with this email exists
391      */
402     public function email_check($email,$edit_username):bool
413     {
424         $user_model = new UserModel();
435         if ($user_model->have_email($email, $edit_username))
446             return FALSE;
457         return TRUE;
468     }
479
480     /**
491      * checks whether the entered registration code is correct or not
502      *
513      */
524     public function _registration_code($code){
535         $settings_model = new SettingsModel();
546         $src = $settings_model->get_setting('registration_code');
557         if ($src == '0')
568             return TRUE;
579         if ($src == $code)
580             return TRUE;
591         return FALSE;
602     }
613
624     /**
635      * Required
646      *
657      * @param string
668      * @return bool
679      */
680     public function required($str)
691     {
702         return is_array($str) ? (bool) count($str) : ($str != '');
713     }
724
735
746     // -----
757

```

```

68
69 /**
70  * Is Lowercase
71  *
72  * @param $str
73  * @return bool
74  */
75 public function lowercase($str)
76 {
77     return (strtolower($str) === $str);
78 }
79
80 // -----
81
82 public function _check_language($str)
83 {
84     if ($str=='')
85         return FALSE;
86     if (in_array( strtolower($str),array('c', 'c+', 'python 2', 'python 3', 'java', 'zip', 'pdf', 'txt')))
87         return TRUE;
88     return FALSE;
89 }
90
91 // -----
92 // Used in Submissions.php
93
94 public function _check_type($type)
95 {
96     return ($type === 'code' || $type === 'result' || $type === 'log');
97 }
98
99 }

```

Aturan-aturan diatas dipindahkan dari *controller* menuju **Validation.php** agar dapat digunakan untuk melakukan validasi. Aturan yang sudah dibentuk dapat digunakan seperti aturan lainnya dengan cara menulis nama kelasnya. Setelah aturan ditetapkan akan dieksekusi berdasarkan *request* dari pengguna dan dilakukan validasi. Kode 4.11 merupakan perubahan penggunaan *validation* terhadap data yang sudah diberikan oleh pengguna.

Kode 4.11: Perancangan perubahan penggunaan *validation* pada *CodeIgniter 4*

```

40
41 if ($this->validation->withRequest($this->request)->run())
42 {
43     if ( !$this->request->isAJAX() ){
44         exit;
45     }else{
46         list($ok, $error) = $this->user_model->add_users(
47             $this->request->getPost('new_users'),
48             $this->request->getPost('send_mail'),
49             $this->request->getPost('delay')
50         );
51         return view('pages/admin/add_user_result', array('ok' => $ok, 'error' => $error));
52     }
53 }

```

Kode 4.11 akan menjalankan *validation* berdasarkan *request* dari pengguna. *Validation* akan tetap menggunakan sintaks `run()` namun akan ada penambahan sintaks `withRequest` dimana validasi akan dijalankan setiap ada *HTTP Request* dari pengguna. Namun, *CodeIgniter 4* tidak menyediakan fungsi `form_error` sehingga akan diubah dengan menggunakan fungsi baru bernama `validation_errors()`. Fungsi tersebut dapat digunakan untuk mengembalikan *error* apabila terdapat data yang tidak sesuai dengan aturan. *Error* tersebut dapat ditampilkan pada halaman *view* menggunakan sintaks berikut.

```
<?= $validation->hasError('username') ? $validation->getError('username') : '' ?>
```

Sintaks diatas akan melakukan pengecekan apakah terdapat *error* dari *form* bernama **username**

- 1 dan apabila terdapat maka akan dikembalikan data *error* yang berasal dari *validation*. Variabel  
 2 *validation* akan dikirimkan dari *controller* berisikan *library validation*.

### 3 *Zip Archive*

- 4 *Zip Encoding* akan digantikan dengan fungsi PHP *zip archive* karena sudah tidak tersedia pada  
 5 *CodeIgniter 4*. Fungsi *zip archive* terdapat beberapa perbedaan sehingga akan disesuaikan dengan  
 6 fungsi-fungsi yang ada. Kode 4.12 merupakan perubahan yang terdapat pada *zip encoding*.

Kode 4.12: Perancangan perubahan *zip encoding* menjadi *zip archive*

```

7
81 $this->zip = new \ZipArchive();
92 $this->zip->open($zipname, ZipArchive::CREATE);
103 for ($i=1; $i<=$number_of_problems; $i++)
114 {
125
136     $path = "$root_path/p{$i}/in";
147     $options = ['add_path' => "p{$i}/in/", 'remove_all_path' => TRUE];
158     $this->zip->addGlob($path.'/*.txt', GLOB_BRACE, $options);
169
170     $path = "$root_path/p{$i}/out";
181     $options = ['add_path' => "p{$i}/out/", 'remove_all_path' => TRUE];
192     $this->zip->addGlob($path.'/*.txt', GLOB_BRACE, $options);
203
214     $path = "$root_path/p{$i}/tester.cpp";
225     if (file_exists($path))
236         $this->zip->addFile($path, "p{$i}/tester.cpp");
247
258     $pdf_files = glob("$root_path/p{$i}/*.pdf");
269     if ($pdf_files)
270     {
281         $path = $pdf_files[0];
292         $this->zip->addFile($path, "p{$i}/".shj_basename($path));
303     }
314
325     $path = "$root_path/p{$i}/desc.html";
336     if (file_exists($path))
347         $this->zip->addFile($path, "p{$i}/desc.html");
358
369     $path = "$root_path/p{$i}/desc.md";
370     if (file_exists($path))
381         $this->zip->addFile($path, "p{$i}/desc.md");
392 }
403
414 $pdf_files = glob("$root_path/*.pdf");
425 if ($pdf_files)
436 {
447     $path = $pdf_files[0];
458     $this->zip->addFile($path,shj_basename($path));
469 }
470 $this->zip->close();
481
492 header('Content-Type: application/zip');
503 header('Content-disposition: attachment; filename=' . $zipname);
514 header('Content-Length: ' . filesize($zipname));
525 readfile($zipname);
53

```

- 54 Kode 4.12 merupakan perubahan dari *zip encoding* menjadi *zip archive*. *Zip archive* akan  
 55 dilakukan inisiasi pada variabel *zip*. Selanjutnya akan dibuka file *zip* menggunakan sintaks *open*  
 56 yang menerima dua buah parameter. Parameter pertama berisikan nama *zip* file yang ingin dibentuk  
 57 sedangkan parameter kedua berisikan mode *zip* yang diinginkan. Fungsi  
 58 *Library* yang terdapat pada *CodeIgniter 4* juga dapat di*extend* dan dibentuk sesuai dengan kebutuhan.  
 59 Berikut merupakan *library* yang dibentuk oleh pengguna.

### 60 *Twig*

- 61 *Library* ini tidak akan digunakan untuk membentuk *view* pada *CodeIgniter 4* namun, akan ada  
 62 penggunaan sebuah fungsi *Twig* yang akan dibentuk pada direktori *app/Libraries*. Fungsi tersebut

1 bernama `extra_time_formatter` yang memiliki fungsi untuk mengubah input yang diberikan  
 2 menjadi format jam dikali enam puluh menit. Kode 4.13 merupakan fungsi yang akan dibentuk  
 3 pada direktori `app/Libraries` dengan nama `Twig.php`.

Kode 4.13: Perancangan perubahan *library MY\_Form\_validation* pada *CodeIgniter 4*

```

4  <?php
5  namespace App\Libraries;
6
7
8  class Twig
9  {
10
11      /**
12       * Required
13       *
14       * @param string
15       * @return bool
16       */
17      public function extra_time_formatter($extra_time)
18      {
19          // convert to minutes
20          $extra_time = floor($extra_time/60);
21          // convert to H*60
22          if ($extra_time % 60 == 0 )
23              $extra_time = ($extra_time/60) . '*60';
24          return $extra_time;
25      }
26  }
  
```

28 Kode 4.13 merupakan fungsi yang akan dibentuk pada file `Twig.php`. Fungsi ini akan dipanggil  
 29 pada halaman `add_assignment.php`.

### 30 *Unzip*

31 *Library* ini akan digunakan kembali dan dipindahkan menuju direktori `app/Libraries`. *Library*  
 32 terdapat penghapusan sintaks *defined* dan juga penambahan *namespace*. Kode 4.14 merupakan  
 33 penghapusan sintaks *defined* dan juga penambahan *namespace*.

Kode 4.14: Perancangan perubahan *library Unzip* pada *CodeIgniter 4*

```

34
35 namespace App\Libraries;
36
37 class Unzip
38 {
  
```

40 Kode 4.14 merupakan perubahan yang terdapat pada *library* ini. Terdapat penghapusan sintaks  
 41 *defined* dan penggantian menjadi *namespace*. Selanjutnya, *library* ini akan diinisiasikan pada fungsi  
 42 *controller* yang menggunakan. Kode 4.15 merupakan inisiasi *library unzip* pada *controller*.

Kode 4.15: Perancangan inisiasi *library unzip* pada *controller*

```

43
44
45 $this->unzip = new Unzip();
46 // Create a temp directory
47 $tmp_dir_name = "shj_tmp_directory";
48 $tmp_dir = "$assignments_root/$tmp_dir_name";
49 shell_exec("rm -rf $tmp_dir; mkdir $tmp_dir;");
50
51 // Extract new test cases and descriptions in temp directory
52 $this->unzip->allow(array('txt', 'cpp', 'html', 'md', 'pdf'));
53 $extract_result = $this->unzip->extract("$assignments_root/".$zip_uploaded->getName(), $tmp_dir);
  
```

55 Kode 4.15 merupakan inisiasi *library unzip* pada *controller*. Inisiasi akan dilakukan pada variabel  
 56 `unzip` yang sudah dibentuk diluar fungsi tersebut. Penggunaan akan tetap sama seperti sebelumnya  
 57 sehingga tidak terdapat perubahan sintaks.

## 1 *Password\_hash*

2 *Library* ini tidak akan digunakan dan akan digantikan oleh *password hash* yang disediakan oleh  
 3 PHP. *Library Password\_hash* merekomendasikan pengguna untuk menggunakan fungsi *native* yang  
 4 disediakan oleh PHP apabila aplikasi mendukung PHP versi 5.5 ke atas. Sehingga, akan dilakukan  
 5 konversi menggunakan fungsi yang disediakan oleh PHP bernama `password_hash()`. Seluruh  
 6 penggunaan *library* ini akan diubah menggunakan fungsi yang disediakan oleh PHP dengan metode  
 7 *hashing* sama yaitu *CRYPT\_BLOWFISH*. Perubahan fungsi *hashing* ini bersifat *backward compatible*  
 8 sehingga dapat menggunakan *database* aplikasi terdahulu tanpa perlu membentuk data baru. Berikut  
 9 merupakan contoh pengubahan kode dari *phpass* menjadi *password\_hash*.

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

11 menjadi

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

13 Sintaks `password_hash()` diatas menerima dua buah parameter yakni data yang ingin di enkripsi  
 14 dan tipe enkripsi. Enkripsi akan menggunakan sintaks `PASSWORD_BCRYPT` yang menggunakan tipe  
 15 *hash* berupa *CRYPT\_BLOWFISH*. Selain itu, terdapat fungsi untuk melakukan pengecekan *password*  
 16 yang sudah di enkripsi. Berikut merupakan contoh pengubahan kode untuk melakukan pengecekan  
 17 *password* yang sudah di enkripsi.

```
18         password_verify($password, $query->getRow()->password)
```

19 Sintaks diatas menerima dua buah parameter dengan parameter pertama berupa masukan dari  
 20 pengguna dan parameter berikutnya merupakan *hash* dari *password* yang sudah disimpan. Fungsi ini  
 21 akan mengembalikan data berupa *true* apabila *password* sama dan *false* apabila *password* berbeda.

## 22 *MY\_Form\_validation*

23 *Library MY\_Form\_validation* akan dipindahkan menuju direktori `app/Libraries`. *Library* ini  
 24 akan digunakan kembali dengan perubahan *extends* menjadi menuju `Validation`, penghapusan  
 25 sintaks `defined`, dan akan ada penambahan *namespace* pada baris awal file. Kode 4.16 merupakan  
 26 contoh penambahan *namespace* dan penggantian *extends* pada *library* ini.

Kode 4.16: Contoh perubahan *library MY\_Form\_validation* pada *CodeIgniter 4*

```
27 1 namespace App\Libraries;  
28 2  
29 3  
30 3 use CodeIgniter\Validation\Validation;  
31 4  
32 5  
33 5 class MY_Form_validation extends Validation
```

34 Kode 4.16 mengapus sintaks `defined` dan menggantikannya dengan penambahan *namespace*. Selain  
 35 itu, kelas *library* akan *extends Validation*.

## 36 *MY\_Profiler*

### 37 *Parsedown*

38 *Library Parsedown* akan dipindahkan menuju direktori `app/Libraries`. *Library* ini akan digunakan  
 39 kembali dengan penambahan *namespace* pada baris awal file dan penghapusan sintaks `defined`.  
 40 Kode 4.17 merupakan contoh penambahan *namespace* dan juga penambahan sintaks *defined*.

Kode 4.17: Perancangan perubahan *library Parsedown* pada *CodeIgniter 4*

```

1
2 1 namespace App\Libraries;
3 2
4 3 class Parsedown

```

6 Kode 4.17 menghapus sintaks **defined** dan menggantikannya dengan penambahan *namespace*.  
 7 Penggunaan *library* ini tidak akan berubah sehingga tidak terdapat perbedaan sintaks. Namun,  
 8 terdapat perubahan cara inisiasi *library* ini dimana sekarang akan menggunakan sintaks *new* dan  
 9 dilakukan inisiasi pada *BaseController*. Kode 4.18 merupakan perubahan cara melakukan inisiasi  
 10 *library parsedown*.

Kode 4.18: Perancangan perubahan inisiasi *library Parsedown* pada *controller CodeIgniter 4*

```

11
12 1 protected $parsedown;
13 2
14 3 /**
15 4  * Constructor.
16 5  */
17 6 public function initController(RequestInterface $request, ResponseInterface $response, LoggerInterface $logger)
18 7 {
19 8     // Do Not Edit This Line
20 9     parent::initController($request, $response, $logger);
21 10
22 11     // Preload any models, libraries, etc, here.
23 12
24 13     // E.g.: $this->session = \Config\Services::session();
25 14     $this->parsedown = new Parsedown();
26 15 }

```

28 Kode 4.18 merupakan perubahan inisiasi pada *CodeIgniter 4*. *Library parsedown* akan dilakukan  
 29 inisiasi menuju variabel **parsedown** yang sudah dibentuk pada luar fungsi. Inisiasi dilakukan pada  
 30 *BaseController* karena terdapat pemakaian pada beberapa *model* dan *controller*.

### 31 *Phpexcel*

32 *Library* ini akan digunakan kembali namun tidak akan dipindahkan menuju **app/Libraries**. *Library*  
 33 akan dilakukan instalasi melalui *composer* dengan sintaks berikut:

```

34
35 composer require phpooffice/phpexcel

```

35 Sintaks diatas akan dijalankan pada akar dari aplikasi dan tidak terdapat perubahan terhadap  
 36 penggunaan sintaks ini.

### 37 *Shj\_pagination*

38 *Library* ini akan digunakan kembali dan dipindahkan menuju direktori **app/Libraries**. Selain itu,  
 39 terdapat penambahan *namespace* pada baris awal file dan penghapusan sintaks *defined*. Kode 4.19  
 40 merupakan penambahan perubahan dan penambahan sintaks pada *library* ini.

Kode 4.19: Perancangan perubahan *library Shj\_pagination* pada *CodeIgniter 4*

```

41
42 1 namespace App\Libraries;
43 2
44 3 class Shj_pagination
45 4 {

```

### 4.2.5 Model

Model akan dipindahkan sesuai dengan direktori 4.2 dan diubah sesuai dengan dokumentasi *CodeIgniter 4*. Seluruh *Model* akan diganti penamaannya dari yang sebelumnya menggunakan *snakecase* menjadi *camelcase*. Berikut merupakan rincian

### 4.2.6 View

*View* akan diubah menggunakan *extension .php* sesuai pada dokumentasi *CodeIgniter 4*. Seluruh *file view* akan diubah menjadi *extension .php* dari yang sebelumnya menggunakan *.twig*. Seluruh *delimiters* juga akan diubah menggunakan fungsi pada *CodeIgniter 4*. Perubahan *view* dapat dilihat pada kode 4.20.

Kode 4.20: Perubahan *view* pada *Login.php*

```

10  <!-- {#
11  # SharIF Judge
12  # file: login.twig
13  # author: Mohammad Javad Naderi <mjnaderi@gmail.com>
14  #} -->
15  <!DOCTYPE html>
16  <html lang="en">
17  <?=$this->include('templates/simple_header')?>
18
19
20  <?=$form_open() ?>
21      <div class="box login">
22
23          <div class="judge_logo">
24              <a href="<?=$site_url() ?>"></a>
25          </div>
26
27          <div class="login_form">
28              <div class="login1">
29                  <p>
30                      <label for="form_username">Username</label><br/>
31                      <input id="form_username" type="text" name="username" required="required" pattern="[0-9a-z]{3,20}" title="The
32                          Username field must be between 3 and 20 characters in length, and contain only digits and lowercase
33                          letters" class="sharif_input" value="<?=$set_value('username') ?>" autofocus="autofocus"/>
34                      <?=$isset($this->errors['username'])?>
35                  </p>
36                  <p>
37                      <label for="form_password">Password</label><br/>
38                      <input id="form_password" type="password" name="password" required="required" pattern=".{6,200}" title="The
39                          Password field must be at least 6 characters in length" class="sharif_input"/>
40                      <?=$isset($this->errors['password'])?>
41                  </p>
42                  <?php if ($error): ?>
43                      <div class="shj_error">Incorrect username or password.</div>
44                  <?php endif ?>
45              </div>
46              <div class="login2">
47                  <p style="margin:0;">
48                      <?php if ($registration_enabled): ?>
49                          <a href="<?=$site_url('register') ?>">Register</a> |
50                      <?php endif ?>
51                      <a href="<?=$site_url('login/lost') ?>">Reset Password</a>
52                      <input type="submit" value="Login" id="sharif_submit"/>
53                  </p>
54              </div>
55          </div>
56      </div>
57  </div>
58  <?=$form_close() ?>
59  </body>
60  </html>
61

```

Kode 4.20 merupakan perubahan yang terdapat pada halaman *view*. *Delimiters* `{{ }}` akan digantikan menjadi `<?=$ ?>` sedangkan `{% %}` akan digantikan menjadi `<?php ?>`. *Delimiters* yang memanggil fungsi pada *CodeIgniter 4* akan digantikan menjadi `<?=$ ?>`. Perubahan juga terdapat pada sintaks dari yang sebelumnya menggunakan `include` akan digantikan menggunakan



- 1 fungsi *CodeIgniter* 4 berupa `$this->include`. Selain terdapat perubahan *extension* dan *delimiters*,  
2 terdapat juga penambahan kode pada *Controller* karena tidak mendukung pembentukan variabel  
3 *global* pada *View*. Kode 4.21 merupakan contoh penambahan kode pada *Controller*.

Kode 4.21: Penambahan kode pada *Login.php*

```
4  
5 1 $data = [  
6 2     'error' => FALSE,  
7 3     'registration_enabled' => $this->settings_model->get_setting('enable_registration'),  
8 4     'title' => 'Login',  
9 5     'validationError' => $this->validation  
10 6 ];
```

- 12 Kode 4.21 merupakan contoh penambahan data pada *controller*. Penambahan data terjadi  
13 karena halaman *view* PHP tidak dapat mendeklarasikan variabel secara *global* sehingga data-data  
14 seperti *title* tidak dapat diakses oleh *view* lainnya.

#### 15 4.2.7 public

##### 16 assets

- 17 Direktori ini berisikan seluruh data yang dapat dilihat oleh pengguna seperti *javascript*, gambar,  
18 dan lainnya. Berikut merupakan rincian isi pada direktori ini:

- 19 • ace
- 20 • font
- 21 • fullcalendar
- 22 • gridster
- 23 • images
- 24 • js
- 25 • nano\_scroller
- 26 • noty
- 27 • pdfjs
- 28 • reveal
- 29 • snippet
- 30 • styles
- 31 • tinymce



## 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