

SKRIPSI

KUSTOMISASI SHARIF JUDGE UNTUK KEBUTUHAN PROGRAM STUDI TEKNIK INFORMATIKA



Stillmen Vallian

NPM: 2014730083

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

«tahun»

UNDERGRADUATE THESIS

«JUDUL BAHASA INGGRIS»



Stillmen Vallian

NPM: 2014730083

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

«tahun»

LEMBAR PENGESAHAN

KUSTOMISASI SHARIF JUDGE UNTUK KEBUTUHAN PROGRAM STUDI TEKNIK INFORMATIKA

Stillmen Vallian

NPM: 2014730083

Bandung, «tanggal» «bulan» «tahun»

Menyetujui,

Pembimbing Utama

Pembimbing Pendamping

Pascal Alfadian, M.Comp.

«pembimbing pendamping/2»

Ketua Tim Penguji

Anggota Tim Penguji

«penguji 1»

«penguji 2»

Mengetahui,

Ketua Program Studi

Mariskha Tri Adithia, P.D.Eng

PERNYATAAN

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

KUSTOMISASI SHARIF JUDGE UNTUK KEBUTUHAN PROGRAM STUDI TEKNIK INFORMATIKA

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

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

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

Meterai Rp. 6000

Stillmen Vallian
NPM: 2014730083

ABSTRAK

«Tuliskan abstrak anda di sini, dalam bahasa Indonesia»

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

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

ABSTRACT

«Tuliskan abstrak anda di sini, dalam bahasa Inggris»

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

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

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

KATA PENGANTAR

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

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

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

Bandung, «bulan» «tahun»

Penulis

DAFTAR ISI

KATA PENGANTAR	xv
DAFTAR ISI	xvii
DAFTAR GAMBAR	xix
DAFTAR TABEL	xxi
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	1
1.3 Tujuan	1
1.4 Batasan Masalah	2
1.5 Metodologi	2
1.6 Sistematika Pembahasan	2
2 LANDASAN TEORI	3
2.1 <i>CodeIgniter</i>	3
2.1.1 Fitur-fitur CodeIgniter	4
2.1.2 <i>Flow Chart</i> Aplikasi	5
2.1.3 <i>Model-View-Controller</i>	5
2.1.4 Desain dan Tujuan Arsitektur	9
2.2 <i>Sharif Judge</i>	10
2.2.1 Instalasi	10
2.2.2 <i>Clean URLs</i>	11
2.2.3 <i>Users</i>	12
2.2.4 Menambah Tugas	13
2.2.5 Struktur Pengujian	20
2.2.6 Deteksi Kecurangan	23
3 ANALISIS	25
3.1 Analisis Kebutuhan Perangkat Lunak <i>Sharif Judge</i>	25
3.1.1 <i>Security with PHP</i>	27
3.1.2 <i>Securing Assignment</i>	27
3.1.3 <i>New Function</i>	28
3.1.4 <i>Solve Problem Indicator</i>	28
3.1.5 <i>Some Problem & Sugestion</i>	28
3.1.6 <i>Queue failed to process if submission take too long to complete?</i>	28
3.1.7 <i>Compilation Error on All Language</i>	29
3.1.8 Membatasi soal (deskripsi & PDF) hanya bisa diakses saat <i>assignment "open"</i> dan setelah waktu mulai	29
3.1.9 Menguji kemiripan kode antar mahasiswa	29
3.1.10 Satu Akun hanya dapat login satu waktu	29

3.1.11	Membatasi soal (deskripsi & PDF) hanya bisa diakses saat <i>assignment "open"</i> dan setelah waktu mulai	30
3.1.12	Mengumpulkan file dengan format TXT	30
3.1.13	Membatasi soal (deskripsi & PDF) hanya bisa diakses saat <i>assignment "open"</i> dan setelah waktu mulai	30
3.1.14	Mengumpulkan file dengan format TXT	31
3.1.15	Perlu ditambah petunjuk penamaan file <i>input</i> dan <i>output</i>	31
3.1.16	Membatasi soal (deskripsi & PDF) hanya bisa diakses saat <i>assignment "open"</i> dan setelah waktu mulai	31
3.1.17	Pendaftaran peserta disertai dengan <i>Display Name</i>	31
3.1.18	Nama pengguna <i>Sharif Judge</i> seharusnya tidak bisa diubah	31
3.1.19	<i>Sharif Judge</i> diharapkan memiliki fungsi dimana <i>assignment</i> dapat dikumpulkan tanpa adanya batasan waktu	32
3.1.20	<i>Sharif Judge</i> diharapkan memiliki <i>Scoreboard global</i> untuk semua <i>assignment</i>	32
3.1.21	Membatasi soal (deskripsi & PDF) hanya bisa diakses saat <i>assignment "open"</i> dan setelah waktu mulai	33
3.1.22	Mengumpulkan file dengan format TXT	33
3.1.23	UI masih merepotkan	34
3.1.24	UI ada yang tidak berguna	34
3.1.25	<i>Sharif Judge</i> diharapkan memiliki fungsi <i>reminder</i>	34
3.1.26	Membatasi soal (deskripsi & PDF) hanya bisa diakses saat <i>assignment "open"</i> dan setelah waktu mulai	34
3.1.27	Integrasi <i>login</i> ke <i>RADIUS</i>	34
3.1.28	Mengumpulkan file dengan format TXT	35
3.1.29	Mengumpulkan file JAR (java multi kelas)	35
3.1.30	Branding Teknik Informatika	35
3.1.31	Download Excel tidak berfungsi pada halaman <i>Submission</i>	36
3.2	Analisis Sistem Kini	38
3.2.1	<i>Model</i>	38
3.2.2	<i>View</i>	41
3.2.3	<i>Controller</i>	51
4	PERANCANGAN	55
4.1	Mengganti method <i>shell_exec("rm ...")</i> menjadi <i>unlink()</i>	55
4.2	Menambahkan method rekoneksi ke <i>database</i>	55
4.3	Membatasi soal (deskripsi & PDF) hanya bisa diunduh saat <i>assignment "open"</i> dan setelah waktu mulai	56
4.4	Mensupport <i>file</i> dengan ekstensi TXT	58
4.5	Membuat halaman <i>Logs</i> yang mencatat aktivitas <i>login</i> pengguna	60
4.6	Menambahkan parameter " <i>Display Name</i> " pada pendaftaran peserta <i>Sharif Judge</i>	63
4.7	Menambahkan fitur " <i>Lock Student's Display Name</i> "	68
4.8	Menambahkan fitur " <i>Archived Assignment</i> "	71
4.9	Menambahkan halaman <i>Hall of Fame</i>	75
4.10	Integrasi <i>login</i> ke <i>server RADIUS</i>	78
	DAFTAR REFERENSI	81
	A KODE PROGRAM	83
	B HASIL EKSPERIMEN	85

DAFTAR GAMBAR

2.1	<i>Flow Chart Aplikasi</i>	5
2.2	Tampilan Halaman <i>Assignments</i>	13
3.1	<i>Mockup</i> Halaman <i>Logs</i>	30
3.2	<i>Mockup</i> Halaman <i>Hall of Fame</i>	33
3.3	Detail Skor	33
3.4	Logo dan Ikon	35
3.5	<i>Banner SharIF Judge</i>	35
3.6	Halaman Login SharIF Judge	36
3.7	Ikon SharIF Judge pada Title Bar	36
3.8	Logo SharIF Judge pada Top Bar	36
3.9	<i>Dashboard</i>	41
3.10	<i>Notifications</i>	42
3.11	<i>Assignments</i>	43
3.12	<i>Problems</i>	43
3.13	<i>Submit</i>	44
3.14	<i>Profile</i>	44
3.15	<i>Scoreboard</i>	45
3.16	<i>All Submission</i>	46
3.17	<i>Final Submission</i>	46
3.18	<i>Settings</i>	47
3.19	<i>User</i>	47
3.20	<i>Add User</i>	48
3.21	<i>Add Notification</i>	48
3.22	<i>Add Assignment</i>	49
3.23	<i>Rejudge</i>	49
3.24	<i>Login</i>	50
3.25	<i>Register</i>	50
3.26	<i>Lost</i>	51
4.1	Rancangan tampilan halaman <i>logs</i>	62
4.2	Rancangan tampilan halaman <i>Hall of Fame</i>	76
4.3	Rancangan tampilan <i>details Hall of Fame</i> peserta tertentu	76
B.1	Hasil 1	85
B.2	Hasil 2	85
B.3	Hasil 3	85
B.4	Hasil 4	85

DAFTAR TABEL

2.1	<i>User Roles Table</i>	12
2.2	<i>Permission Table</i>	12
2.3	Masalah 1 (Penjumlahan)	16
2.4	Masalah 2 (<i>Max</i>)	17
3.1	Tabel Analisis Kebutuhan <i>Sharif Judge</i>	26
4.1	Perancangan Tabel <i>shj_logins</i>	61
4.2	Perincian fungsi <i>insert_to_logs</i>	61
4.3	Perincian fungsi <i>get_all_logs</i>	62
4.4	Perincian fungsi <i>consturct__</i>	63
4.5	Perincian fungsi <i>index</i>	63
4.6	Perincian fungsi <i>get_all_final_submission</i>	75
4.7	Perincian fungsi <i>get_all_user_assignments</i>	75
4.8	Perincian fungsi <i>consturct__</i>	77
4.9	Perincian fungsi <i>index</i>	77
4.10	Perincian fungsi <i>hof_details</i>	77

BAB 1

PENDAHULUAN

1.1 Latar Belakang

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

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

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

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

1.2 Rumusan Masalah

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

1.3 Tujuan

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

1.4 Batasan Masalah

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

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

1.5 Metodologi

Metodologi yang dilakukan dalam pengerjaan skripsi ini adalah sebagai berikut:

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

1.6 Sistematika Pembahasan

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

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

BAB 2

LANDASAN TEORI

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

2.1 *CodeIgniter*

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

Beberapa keunggulan dari *CodeIgniter* yaitu:

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

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

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

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

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

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

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

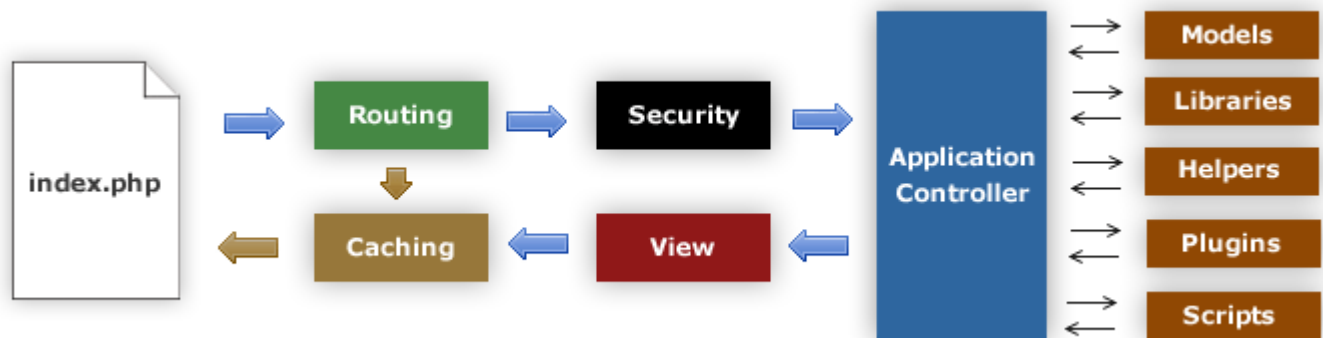
2.1.1 Fitur-fitur CodeIgniter

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

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

2.1.2 *Flow Chart Aplikasi*

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



Gambar 2.1: *Flow Chart Aplikasi*

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

2.1.3 *Model-View-Controller*

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

2.1.3.1 *Model*

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

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

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

```

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

```

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

- Menghubungkan Sebuah Model

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

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

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

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

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

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

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

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

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

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

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

```

class Blog_controller extends CI_Controller {

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

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

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

```

- Auto-loading Model

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

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

- Koneksi ke Database

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

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

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

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

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

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

2.1.3.2 View

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

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

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

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

2.1.3.3 Controller

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

```
<?php
class Blog extends CI_Controller {

}
```

Contoh yang salah :

```
<?php
class blog extends CI_Controller {

}
```

Berikut beberapa hal penting yang terdapat pada Controller:

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

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

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

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

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

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

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

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

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

- Mengorganisir Controller ke Dalam Sub Direktori

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

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

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

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

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

2.1.4 Desain dan Tujuan Arsitektur

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

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

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

2.2 *Sharif Judge*

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

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

2.2.1 Instalasi

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

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

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

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

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

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

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

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

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

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

2.2.2 Clean URLs

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

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

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

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

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

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

```
RewriteEngine on
```

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

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

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

menjadi

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

2.2.3 Users

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

Tabel 2.1: <i>User Roles Table</i>	
<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 *users* memiliki aksi yang berbeda-beda. Aksi yang dapat dilakukan para *users* akan disesuaikan dengan *level* masing-masing. Perhatikan tabel 2.2 berikut:

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

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

```
USERNAME EMAIL PASSWORD ROLE
```

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

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

Contoh:

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

2.2.4 Menambah Tugas

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

Gambar 2.2: Tampilan Halaman *Assignments*

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

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

- *Participants*

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

- *Open*

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

- *Scoreboard*

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

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

```

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

```

- *Coefficient Rule*

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

```

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

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

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

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

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

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

```

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

2.2.4.1 Contoh Tugas

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

1. Masalah 1 (Penjumlahan)

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

Tabel 2.3: Masalah 1 (Penjumlahan)

Sample Input	Sample Output
5 54 78 0 4 9	145

2. Masalah 2 (*Max*)

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

Tabel 2.4: Masalah 2 (*Max*)

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

3. Masalah 2 (*Upload!*)

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

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

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

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

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

2.2.4.2 Contoh Solusi

Permasalahan diatas dapat diselesaikan menggunakan contoh solusi berikut ini:

- Solusi Masalah 1

Menggunakan bahasa *C*

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

Menggunakan bahasa *C++*

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

Menggunakan bahasa *Java*

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



```

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

```

- Solusi Masalah 2
Menggunakan bahasa C

```

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

```

Menggunakan bahasa C++

```

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

```

```

        return 0;
    }

```

2.2.5 Struktur Pengujian

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

2.2.5.1 Metode Pengecekan

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

- *Metode Input/Output Comparison*

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

- *Metode Tester*

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

```

/*
* tester.cpp
*/

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

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

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

    ...

```

```
}

```

2.2.5.2 Contoh *File*

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

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

```

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

```
/*
* tester.cpp

```

```

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

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

    int sum, user_output;
    user_out >> user_output;

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

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

```

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

2.2.6 Deteksi Kecurangan

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

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

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

```
...

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

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

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

...
}
```

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

BAB 3

ANALISIS

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

3.1 Analisis Kebutuhan Perangkat Lunak *Sharif Judge*

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

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

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

Tabel 3.1: Tabel Analisis Kebutuhan *Sharif Judge*

No	Deskripsi	Sumber	Issue Number / Nama Mata Kuliah	Pembuat Issue Nama Dosen	Status
1	<i>Security with PHP</i>	<i>GitHub</i>	#61	kathiedart	Diimplemntasi
2	<i>Securing Assignment</i>	<i>GitHub</i>	#55	wojcik13	Tidak diimplementasi
3	<i>New Function</i>	<i>GitHub</i>	#53	wojcik13	Tidak diimplementasi
4	<i>Solved Problem Indicator</i>	<i>GitHub</i>	#46	atiabjobayer	Tidak diimplementasi
5	<i>Some Problem & Sugestion</i>	<i>GitHub</i>	#45	atiabjobayer	Tidak diimplementasi
6	<i>Queue failed to process if submission take too long to complete?</i>	<i>GitHub</i>	#32	truongan	Diimplemntasi
7	<i>Compilation Error on all language</i>	<i>Git</i> Hub	#34	Eirin07	Tidak diimplementasi
8	Membatasi soal (deskripsi dan PDF) hanya bisa diakses saat <i>assignment</i> "open" dan setelah waktu mulai	Dosen	ASD	Husnul Hakim	Diimplementasi
9	Menguji kemiripan kode antar mahasiswa (Contek)	Dosen	ASD	Husnul Hakim	Tidak diimplementasi
10	1 Akun hanya dapat <i>login</i> 1 waktu (Jika suatu akun sedang <i>login</i> , tidak ada lagi yang bisa <i>login</i> akun tersebut)	Dosen	ASD	Husnul Hakim	Diimplementasi
11	Membatasi soal (deskripsi dan PDF) hanya bisa diakses saat <i>assignment</i> "open" dan setelah waktu mulai	Dosen	ASD	Vania Natali	Diimplementasi
12	<i>Sharif Judge</i> tidak dapat menerima file dengan ekstensi *.txt untuk <i>Pre-Test</i>	Dosen	ASD	Vania Natali	Diimplementasi
13	Membatasi soal (deskripsi dan PDF) hanya bisa diakses saat <i>assignment</i> "open" dan setelah waktu mulai	Dosen	DAA	Luciana Abednego	Diimplementasi
14	<i>Sharif Judge</i> tidak dapat menerima file dengan ekstensi *.txt untuk <i>Pre-Test</i>	Dosen	DAA	Luciana Abednego	Diimplementasi
15	Perlu ditambah petunjuk penamaan file <i>input</i> & <i>output</i> yg langsung bisa dilihat ketika hendak <i>upload</i>	Dosen	DAA	Luciana Abednego	Tidak diimplementasi
16	Membatasi soal (deskripsi dan PDF) hanya bisa diakses saat <i>assignment</i> "open" dan setelah waktu mulai	Dosen	DAA	Joanna Helga	Diimplementasi
17	<i>Register</i> peserta yg <i>mode batch</i> , <i>Sharif Judge</i> tidak minta nama orangnya (lebih baik ada nama orangnya)	Dosen	DAA	Joanna Helga	Diimplementasi
18	Nama peserta seharusnya tidak bisa diganti (Bisa menjadi "mainan" dan tindak kecurangan karena dapat memberikan <i>hint</i>)	Dosen	DAA	Joanna Helga	Diimplementasi
19	Ingin memiliki fungsi dimana <i>Assignment</i> tidak memiliki batasan waktu (arsip soal tahun lalu dapat dikerjakan kapan saja)	Dosen	DAA	Joanna Helga	Diimplementasi
20	Ingin memiliki <i>scoreboard global</i> untuk semua <i>assignment</i> .	Dosen	DAA	Joanna Helga	Diimplementasi
21	Membatasi soal (deskripsi dan PDF) hanya bisa diakses saat <i>assignment</i> "open" dan setelah waktu mulai	Dosen	ASD & DAA	Claudio Fransiscus	Diimplementasi
22	<i>Sharif Judge</i> tidak dapat menerima file dengan ekstensi *.txt untuk <i>Pre-Test</i>	Dosen	ASD & DAA	Claudio Fransiscus	Diimplementasi
23	UI masih merepotkan	Dosen	ASD & DAA	Claudio Fransiscus	Tidak diimplementasi
24	UI ada yang tidak berguna (yang lebih banyak digunakan <i>assignment</i> , <i>submit</i> , <i>scoreboard</i> , dan hasil <i>submit</i>)	Dosen	ASD & DAA	Claudio Fransiscus	Tidak diimplementasi

25	Ingin memiliki fungsi <i>reminder</i> . Banyak mahasiswa lupa mengerjakan tugas dan tidak bisa mengumpulkan. Fungsi <i>reminder</i> akan mengirimkan <i>reminder</i> ke <i>email</i> mahasiswa	Dosen	ASD & DAA	Claudio Fransiscus	Tidak diimplementasi
26	Membatasi soal (deskripsi dan PDF) hanya bisa diakses saat <i>assignment</i> "open" dan setelah waktu mulai	Dosen	ASD & DAA	Pascal Alfadian	Diimplementasi
27	Integrasi login ke <i>RADIUS</i> (<i>password</i> sama dengan <i>login Windows</i>)	Dosen	AIF401	Pascal Alfadian	Diimplementasi
28	Mengumpulkan file TXT	Dosen	AIF401	Pascal Alfadian	Diimplementasi
29	Mengumpulkan file JAR (java multi kelas)	Dosen	AIF401	Pascal Alfadian	Tidak diimplementasi
30	Branding Teknik Informatika	Dosen	AIF401	Pascal Alfadian	Diimplementasi
31	Download <i>Excel</i> tidak berfungsi pada halaman <i>Submission</i>	Asisten Dosen	DAA	Kresna Dwi Cahyo	Diimplementasi

Pada tabel di atas terdapat beberapa kolom, yaitu:

- **Deskripsi**
Jika sumber kebutuhan berasal dari *GitHub*, maka pada kolom deskripsi akan ditulis judul dari isu yang ada pada repositori. Jika sumber kebutuhan berasal dari dosen, maka deskripsi kebutuhan langsung ditulis pada kolom tersebut.
- **Sumber**
Kolom sumber berisikan sumber dari kebutuhan pengembangan *Sharif Judge* yaitu *GitHub* atau Dosen.
- **Issue Number / Nama Mata Kuliah**
Jika sumber kebutuhan berasal dari *GitHub*, maka pada kolom ini akan ditulis **kode unik** yang ada pada setiap isu. Jika sumber kebutuhan berasal dari dosen, maka pada kolom ini akan ditulis mata kuliah yang diajar oleh dosen tersebut.
- **Pembuat Issue / Nama Dosen**
Jika sumber kebutuhan berasal dari *GitHub*, maka pada kolom ini akan ditulis *username* pembuat isu tersebut. Jika sumber kebutuhan berasal dari dosen, maka pada kolom ini akan ditulis nama dosen yang memberikan daftar kebutuhan.
- **Status**
Kolom Status berisikan status dari kebutuhan tersebut apakah akan diimplementasi atau tidak diimplementasi.

3.1.1 *Security with PHP*

Isu dengan kode unik #61 dibuat oleh pengguna *GitHub* dengan username *danwdart* pada tanggal 6 April 2017. Pada isu tersebut dikatakan bahwa seseorang pengguna *Sharif Judge* dapat mengubah parameter *PHP shell_exec(rm ...)* yang mengakibatkan pengeksekusian kode bisa dilakukan secara sewenang-wenang. Untuk menghindari hal di atas, username *danwdart* menyarankan untuk mengganti perintah *PHP shell_exec(rm ...)* dengan method lain.

Pada skripsi ini, isu di atas akan diimplementasi. Solusi yang ditawarkan adalah mengganti penggunaan *PHP shell_exec(rm ...)* dengan *method* lain. Perintah *shell_exec("rm ...")* memiliki fungsi untuk menghapus sebuah file. Perintah tersebut dapat diganti menggunakan *method unlink()* yang memiliki fungsi sama yaitu menghapus sebuah file.

3.1.2 *Securing Assignment*

Isu dengan kode unik #55 dibuat oleh pengguna *GitHub* dengan username *wojcik13* pada tanggal 23 Oktober 2016. *Username wojcik13* menanyakan apakah ada pilihan pada *Sharif Judge* untuk mengamankan assignment dengan password.

Pada skripsi ini, isu di atas tidak diimplementasi. Isu di atas merupakan sebuah pertanyaan fitur pada *Sharif Judge* untuk mengamankan assignment dengan menggunakan password. Oleh karena isu tersebut merupakan sebuah pertanyaan, maka pada skripsi ini isu di atas tidak diimplementasi.

3.1.3 *New Function*

Isu dengan kode unik #53 dibuat oleh pengguna *GitHub* dengan username *wojcik13* pada tanggal 2 Oktober 2016. *Username wojcik13* menanyakan apakah ada fungsi pada *Sharif Judge* seperti menghentikan *scoreboard* dan fungsi mengatur sesi.

Pada skripsi ini, isu di atas tidak diimplementasi. Isu di atas merupakan sebuah pertanyaan fitur pada *Sharif Judge* untuk menghentikan *scoreboard* dan mengatur sesi. Oleh karena isu tersebut merupakan sebuah pertanyaan, maka pada skripsi ini isu di atas tidak diimplementasi.

3.1.4 *Solve Problem Indicator*

Isu dengan kode unik #46 dibuat oleh pengguna *GitHub* dengan username *atiabjobayer* pada tanggal 16 April 2016. *Username atiabjobayer* mengatakan bahwa para pengguna *Sharif Judge* tidak dapat melihat masalah yang telah diselesaikan.

Pada skripsi ini, isu di atas tidak diimplementasi. Pada isu di atas, username *atiabjobayer* kurang spesifik dalam menjelaskan kebutuhannya. Oleh karena isu tersebut kurang spesifik, maka pada skripsi ini isu di atas tidak diimplementasi.

3.1.5 *Some Problem & Sugestion*

Isu dengan kode unik #45 dibuat oleh pengguna *GitHub* dengan username *atiabjobayer* pada tanggal 16 April 2016. *Username atiabjobayer* mengatakan bahwa ada beberapa persoalan yang ada pada perangkat lunak *Sharif Judge*. Beberapa masalah tersebut yaitu:

1. Beberapa kontes besar pemrograman diadakan dengan metode *ACM ICPC* namun *Sharif Judge* tidak mendukung sistem penilaian *ICPC*.
2. Pengguna dapat melihat deskripsi masalah sebelum kontes dimulai. Hal ini dapat membahayakan keamanan kontes pemrograman.
3. Tampilan yang digunakan pada *Sharif Judge* tidak responsif. Pengguna tidak dapat melihat pada *device* kecil/
4. Seharusnya ada Sistem Klarifikasi. Fitur ini harus ada pada *online judge*.
5. Pengguna tidak dapat mengumpulkan kode mereka dari *text-editor*.

Pada skripsi ini, isu di atas tidak diperbaiki. Isu di atas memiliki cakupan persoalan yang terlalu luas. Oleh karena isu tersebut memiliki persoalan yang terlalu luas, maka pada skripsi ini isu di atas tidak diperbaiki.

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

Isu dengan kode unik #32 dibuat oleh pengguna *GitHub* dengan username *truongan*. Pada isu tersebut dikatakan bahwa *assignment* yang memiliki masalah dengan *test case* besar akan menyebabkan *submission status* menjadi *pending*. *User truongan* memperkirakan hal di atas terjadi dikarenakan *database connection times out*.

Pada skripsi ini, isu di atas akan diperbaiki. Solusi yang ditawarkan untuk mengatasi hal di atas adalah menambahkan *method reconnect database*. *Method reconnect database* akan menghubungkan kembali *database* ketika terjadi kasus seperti di atas.

3.1.7 *Compilation Error on All Language*

Isu dengan kode unik #34 dibuat oleh pengguna *GitHub* dengan *username Eririn07*. *Username Eririn07* mencoba untuk menilai sebuah kode dan mendapatkan beberapa *error*. Kode *error* yang muncul ketika menilai kode dengan bahasa *Java* adalah *Java HotSpot(TM) 64-Bit Server VM warning: INFO: os::commit_memory(0x00007f9cfd000000, 2555904, 1) failed; error='Permission denied' (errno=13)* dan Kode *error File Not Found* muncul ketika menilai kode dengan bahasa *C*. *Username Eririn07* menanyakan bagaimana mengatasi masalah di atas.

Pada skripsi ini, isu di atas tidak diperbaiki. Isu di atas merupakan sebuah pertanyaan untuk mengatasi kode *error* yang muncul. Oleh karena isu tersebut merupakan sebuah pertanyaan, maka pada skripsi ini isu di atas tidak diperbaiki.

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

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

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

3.1.9 Menguji kemiripan kode antar mahasiswa

Perangkat lunak *Sharif Judge* terkini sudah memiliki fitur untuk menguji kemiripan kode antar peserta dengan menggunakan *Moss (Measure Of Software Similarity)*. *Moss* adalah sistem otomatis untuk mendeteksi kemiripan program. Aplikasi *Moss* telah berkembang dari tahun 1994 hingga sekarang. Algoritma yang digunakan pada aplikasi *Moss* sangat efektif dibandingkan algoritma deteksi kecurangan lainnya [4].

Pada skripsi ini, kebutuhan di atas tidak diimplementasi karena aplikasi *Moss* sedang tidak dapat digunakan. Hal tersebut terjadi karena aplikasi *Moss* membutuhkan port 7690, namun port tersebut diblok oleh UNPAR.

3.1.10 Satu Akun hanya dapat login satu waktu

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

Pada skripsi ini, kebutuhan di atas tidak diimplementasi. Jika fitur di atas diimplementasi, maka dikhawatirkan akan merepotkan admin lab. Para admin harus membuka akun pengguna setiap kali ada akun yang terkunci pada sebuah komputer. Sebagai gantinya, penulis menawarkan sebuah solusi. Solusi yang ditawarkan untuk mengurangi tingkat kecurangan seperti di atas yaitu membuat halaman baru yang berisikan *logs username* yang berhasil login ke *Sharif Judge*. Halaman

logs tersebut akan mencatat *username* yang *login* menggunakan ip berbeda dalam waktu dibawah 24 jam. Dengan adanya halaman *Logs* ini, para dosen dapat memantau *username* yang *login* pada dua tempat berbeda.

Berikut *mockup* halaman *logs* yang akan diimplementasikan ke dalam *Sharif Judge*.

#	Login ID	Username	IP Address	Last Login	Last 24 Hour Login ID
1	145	admin	::1	2018-04-02 21:14:05	
2	144	admin	::1	2018-03-23 16:22:12	
3	143	admin	::1	2018-03-22 20:58:55	
4	142	stillmen	::1	2018-03-22 20:58:48	
5	141	stillmen	::1	2018-03-22 20:47:28	
6	140	admin	::1	2018-03-22 20:22:11	
7	139	admin	::1	2018-03-21 22:42:51	
8	138	admin	::1	2018-03-21 18:34:07	
9	137	admin	::1	2018-03-21 18:07:44	
10	136	instructor	::1	2018-03-21 18:07:26	
11	135	admin	::1	2018-03-21 17:51:24	
12	134	admin	::1	2018-03-21 17:49:44	

Gambar 3.1: *Mockup* Halaman *Logs*

Halaman ini akan terletak di bawah menu halaman *Settings* dan hanya dapat diakses oleh role admin.

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

Kebutuhan ini telah dibahas pada sub bab sebelumnya. Lihat sub bab 3.1.8 untuk status dari kebutuhan ini.

3.1.12 Mengumpulkan file dengan format TXT

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

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

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

Kebutuhan ini telah dibahas pada sub bab sebelumnya. Lihat sub bab 3.1.8 untuk status dari kebutuhan ini.

3.1.14 Mengumpulkan file dengan format TXT

Kebutuhan ini telah dibahas pada sub bab sebelumnya. Lihat sub bab 3.1.12 untuk status dari kebutuhan ini.

3.1.15 Perlu ditambah petunjuk penamaan file *input* dan *output*

Dalam membuat sebuah assignment pada perangkat lunak *Sharif Judge* terdapat *file test case* yang harus disertakan. *File test case* yang disertakan memiliki beberapa ketentuan. Beberapa ketentuan tersebut seperti struktur direktori dan penamaan dalam file test case.

Pada skripsi ini, kebutuhan di atas tidak diimplementasi. Pada halaman *add assignment* telah disediakan *link* menuju dokumentasi *Sharif Judge* di *GitHub* yang menjelaskan ketentuan dalam menyertakan *file test case*. Ketentuan tersebut harus terpenuhi agar sebuah *assignment* dapat berjalan dengan baik. Oleh sebab itu, kebutuhan ini tidak diimplementasi.

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

Kebutuhan ini telah dibahas pada sub bab sebelumnya. Lihat sub bab 3.1.8 untuk status dari kebutuhan ini.

3.1.17 Pendaftaran peserta disertai dengan *Display Name*

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

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

3.1.18 Nama pengguna *Sharif Judge* seharusnya tidak bisa diubah

Display Name pada perangkat lunak *Sharif Judge* berfungsi sebagai nama peserta. Selain itu, nama peserta akan muncul pada halaman *Scoreboard* sebuah *assignment* yang dapat dilihat oleh seluruh peserta. *Sharif Judge* yang terkini mengijinkan para peserta untuk mengubah *Display Name* pada halaman *Profile*. Hal tersebut dapat dijadikan sebuah "mainan" dan tindakan kecurangan karena dapat memberikan *hint* untuk peserta lain. Oleh karena itu, Ibu Joanna Helga menginginkan nama peserta yang terdaftar pada *Sharif Judge* tidak dapat diubah.

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

jika *admin* menonaktifkan fitur tersebut, maka *input text Display Name* pada halaman Profile akan kembali aktif.

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

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

Pada skripsi ini, kebutuhan di atas akan diimplementasi. Solusi yang ditawarkan untuk memenuhi kebutuhan di atas yaitu membuat sebuah fitur tambahan pada *assignment*. *Assignment* yang mengaktifkan fitur ini tidak akan muncul pada kalender yang terdapat di halaman *Dashboard*. Fitur tersebut akan membuat batas waktu pengumpulan menjadi tanggal 18 Januari 2038. Tanggal 18 Januari 2038 diambil karena merupakan batas dari waktu *UNIX*. Batas tersebut diperoleh karena setiap detik yang dilewati sejak tanggal 1 Januari 1970 disimpan menggunakan *integer 32-bit*. Pada tanggal 18 Januari 2038 waktu UTC+7 akan mencapai batas maksimum dari *integer 32-bit* tersebut. Masalah ini dikenal sebagai masalah "Year 2038 problem".

3.1.20 *Sharif Judge* diharapkan memiliki *Scoreboard global* untuk semua *assignment*

Sharif Judge terkini memiliki halaman *Scoreboard* yang berfungsi menampilkan seluruh nilai akhir para peserta dari sebuah *assignment*. Pada halaman *Socreboard* juga menampilkan nilai dari setiap *problem* yang ada pada sebuah *assignment*. Nilai yang muncul pada halaman ini adalah nilai para peserta yang telah mengumpulkan jawabannya. Nilai yang muncul tersebut akan diurutkan mulai dari yang tertinggi hingga terendah. Ibu Joanna Helga menginginkan sebuah *Scoreboard global* untuk semua *assignment*. *Scoreboard global* tersebut akan menampilkan total skor beserta detail skor setiap *problem* yang telah dikerjakan para peserta diseluruh *assignment* yang ada.

Pada skripsi ini, kebutuhan di atas akan diimplementasi. Solusi yang ditawarkan untuk memenuhi kebutuhan di atas yaitu membuat halaman baru yang diberi nama *Hall of Fame*. Halaman *Hall of Fame* akan menampilkan berapa *problem* yang telah dikerjakan oleh para peserta diseluruh *assignment* yang ada. Nama peserta yang muncul pada halaman ini diurutkan sesuai dengan total skor dari seluruh *assignment* yang telah dikerjakan oleh para peserta.

Berikut *mockup* halaman *Hall of Fame* yang akan diimplementasikan ke dalam perangkat lunak *Sharif Judge*.

#	Rank	Username	Total Score
1	1	admin	175
2	1	stillmen	175
3	2	dummy	100
4	3	lorem	0

Gambar 3.2: Mockup Halaman *Hall of Fame*

Halaman ini akan terletak di bawah menu halaman *Scoreboard* dan dapat diakses oleh seluruh *role*. Pada halaman ini, akan diberlakukan sistem ranking berdasarkan total skor yang diperoleh para peserta. Untuk melihat detail total skor yang dihasilkan dari setiap *assignment*, para peserta dapat mengklik pada baris yang ada di tabel. Berikut contoh detail skor yang ditampilkan.

Hall of Fame for username: admin	
Soal Kuis	Problem : 100
Dummy Assignment	Problem : 0 Problem Example: 75

Gambar 3.3: Detail Skor

Halaman kecil tersebut akan muncul dan berisikan nilai setiap *problem* dari semua *assignment* yang telah dikerjakan oleh peserta tertentu.

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

Kebutuhan ini telah dibahas pada sub bab sebelumnya. Lihat sub bab 3.1.8 untuk status dari kebutuhan ini.

3.1.22 Mengumpulkan file dengan format TXT

Kebutuhan ini telah dibahas pada sub bab sebelumnya. Lihat sub bab 3.1.12 untuk status dari kebutuhan ini.

3.1.23 UI masih merepotkan

Bapak Claudio Fransiscus mengeluhkan UI pada perangkat lunak *Sharif Judge* merepotkan. Contohnya seperti pada saat peserta ingin memilih *assignment*, para peserta harus masuk ke halaman *assignment* dan memilih *assignment* yang ingin dikerjakan. Contoh lainnya seperti skenario mengumpulkan jawaban, para peserta harus masuk ke halaman *submit*, memilih *problem* yang ingin dikumpulkan, memilih bahasa, memilih file yang akan dikumpulkan dan *submit*.

Pada skripsi ini, kebutuhan di atas tidak diimplementasi. Bapak Claudio Fransiscus masih kurang spesifik dalam menjelaskan kebutuhannya. Oleh karena kebutuhan tersebut kurang spesifik, maka pada skripsi ini kebutuhan di atas tidak diimplementasi.

3.1.24 UI ada yang tidak berguna

Bapak Claudio Fransiscus mengeluhkan beberapa UI pada perangkat lunak *Sharif Judge* tidak berguna. Pada *side bar Sharif Judge* terdapat beberapa menu halaman. Menurut Bapak Claudio Fransiscus, beberapa *menu* tersebut tidak semuanya terpakai. *Menu* yang sering digunakan hanya *Assignments*, *Submit*, *Submission* dan *Scoreboard*.

Pada skripsi ini, kebutuhan di atas tidak diimplementasi. Claudio Fransiscus masih kurang spesifik dalam menjelaskan kebutuhannya. Oleh karena kebutuhan tersebut kurang spesifik, maka pada skripsi ini kebutuhan di atas tidak diimplementasi.

3.1.25 *Sharif Judge* diharapkan memiliki fungsi *reminder*

Setiap *assignment* pada perangkat lunak *Sharif Judge* memiliki batas pengumpulan. Jika *assignment* telah melewati batas pengumpulan maka para peserta tidak dapat mengumpulkan tugasnya. Banyak peserta sering kali lupa untuk mengerjakan *assignment* dan pada akhirnya melewati batas pengumpulan. Bapak Claudio Fransiscus menginginkan perangkat lunak *Sharif Judge* yang memiliki fitur *reminder*. Fitur *reminder* akan mengirimkan *email* ke setiap peserta yang berisikan peringatan bahwa ada *assignment* yang harus diselesaikan.

Pada skripsi ini, kebutuhan di atas tidak diimplementasi. Kebutuhan di atas belum dapat diimplementasi karena masih belum ada sistem *scheduler*. Selain itu, bobot pekerjaan diperkirakan akan membutuhkan waktu lebih dari 1 semester.

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

Kebutuhan ini telah dibahas pada sub bab sebelumnya. Lihat sub bab 3.1.8 untuk status dari kebutuhan ini.

3.1.27 Integrasi *login* ke *RADIUS*

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

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

Pada skripsi ini, kebutuhan di atas akan diimplementasi. Solusi yang ditawarkan untuk memenuhi kebutuhan di atas yaitu mengintegrasikan *login server RADIUS* pada perangkat lunak *Sharif Judge*. Dengan pengimplementasian integrasi *login RADIUS* pada *Sharif Judge*, para peserta dapat *login* ke *Sharif Judge* menggunakan akun yang terdapat pada *server RADIUS*.

3.1.28 Mengumpulkan file dengan format TXT

Kebutuhan ini telah dibahas pada sub bab sebelumnya. Lihat sub bab 3.1.12 untuk status dari kebutuhan ini.

3.1.29 Mengumpulkan file JAR (java multi kelas)

JAR (Java ARchive) adalah format file platform-independen yang menggabungkan banyak file menjadi satu. File-file seperti kelas, gambar dan suara dapat digabungkan dalam file JAR. Perangkat lunak *Sharif Judge* yang terkini tidak mendukung pengumpulan file dengan ekstensi JAR. *Sharif Judge* hanya menerima beberapa tipe file yaitu C, C++, Java, Python 2, Python 3, Zip, dan PDF.

Pada skripsi ini, kebutuhan di atas tidak diimplementasi.

3.1.30 Branding Teknik Informatika

Branding Teknik Informatika akan dilakukan dengan cara mengubah logo dan ikon *Sharif Judge* menjadi logo Teknik Informatika. Selain mengubah logo dan ikon, perubahan juga akan dilakukan terhadap nama perangkat lunak menjadi SharIF Judge dan link dokumentasi yang ada pada perangkat lunak *Sharif Judge*. Hal di atas dapat dilakukan karena *Sharif Judge* sendiri menggunakan lisensi GPL versi 3. GPL merupakan kepanjangan dari *General Public License* yang memberikan beberapa kebebasan pada setiap penggunaanya [5]. Kebebasan tersebut antara lain:

- Kebebasan untuk menggunakan perangkat lunak dengan tujuan apapun
- Kebebasan untuk mengubah perangkat lunak sesuai dengan kebutuhan
- Kebebasan untuk membagikan perangkat lunak kepada teman dan kerabat
- Kebebasan untuk membagikan perubahan yang telah dilakukan

Pada skripsi ini, kebutuhan di atas akan diimplementasi. Beberapa logo yang akan digunakan seperti:

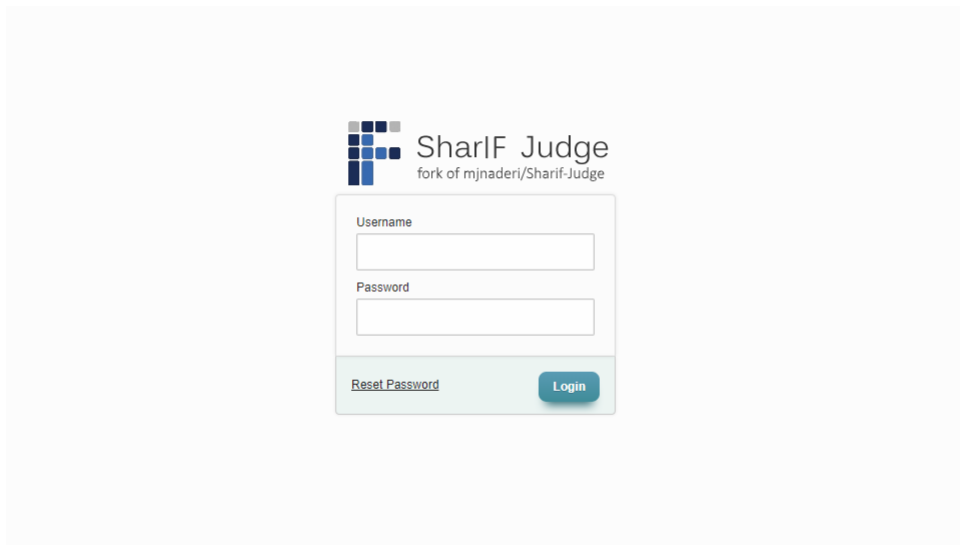


Gambar 3.4: Logo dan Ikon

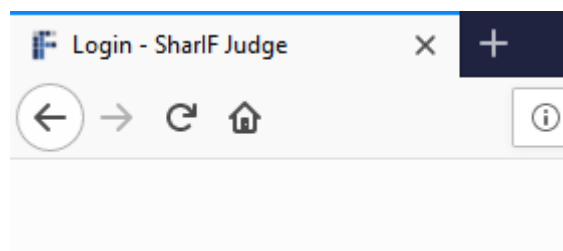


Gambar 3.5: Banner SharIF Judge

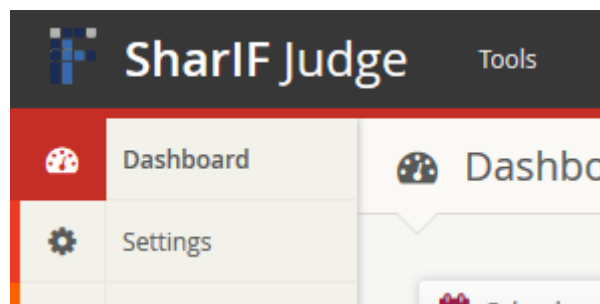
Berikut beberapa tampilan SharIF Judge yang baru.



Gambar 3.6: Halaman Login SharIF Judge



Gambar 3.7: Ikon SharIF Judge pada Title Bar



Gambar 3.8: Logo SharIF Judge pada Top Bar

3.1.31 Download Excel tidak berfungsi pada halaman *Submission*

Perangkat lunak Sharif Judge yang terkini memiliki fitur yang dapat mengunduh excel pada halaman *All Submission*, *Final Submission* dan *Users*. Fitur ini berfungsi untuk memuat seluruh daftar *All Submission*, *Final Submission* dan *Users* ke dalam format excel. Agar fitur ini dapat berjalan dengan baik, *Sharif Judge* menggunakan *library* bantuan yaitu *PHPExcel*. Versi *PHP* yang digunakan oleh *Sharif Judge* yang terkini tidak lagi mendukung *library PHPExcel*. Hal tersebut menyebabkan fitur diatas menjadi tidak berfungsi. Dalam pengembangannya, *PHPExcel* sudah tidak lagi digunakan. Para pengguna disarankan untuk migrasi ke *library* penerusnya yaitu *PhpSpreadsheet* atau alternatif

lainnya ².

Pada skripsi ini, kebutuhan di atas akan diimplementasi. Solusi yang ditawarkan untuk kebutuhan di atas yaitu migrasi dari *library PHPExcel* ke *PhpSpreadsheet* sehingga fitur mengunduh *excel* pada halaman *All Submission*, *Final Submission* dan *Users* dapat berfungsi kembali.

²Adrien Crivelli, "PHPExcel - DEPRECATED" terakhir diubah 25 Desember 2017. *https://github.com/PHPOffice/PHPExcel*

3.2 Analisis Sistem Kini

Seperti yang telah dibahas pada [Bab 2.2](#), *Sharif Judge* menggunakan *framework Codeigniter*. *Framework* ini menerapkan prinsip *Model-View-Controller (M-V-C)*. *Model-View-Controller* merupakan metode untuk membuat sebuah aplikasi dengan memisahkan data (*Model*) dari tampilan (*View*) dan cara memprosesnya (*Controller*). Pada perangkat lunak *Sharif Judge* *model*, *view* dan *controller* terdapat pada folder *Application* serta dipisahkan ke dalam masing-masing folder.

3.2.1 Model

Direktori model perangkat lunak *Sharif Judge* terdapat pada `Sharif-Judge\application\models`. Di dalam folder *models*, terdapat beberapa file model yang berisikan fungsi-fungsi untuk membantu pengguna mengambil, menyimpan, dan memperbarui informasi pada *database*.

3.2.1.1 Assignment_Model.php

Pada file *Assignment_Model.php* terdapat beberapa fungsi yaitu:

- *add_assignment*: menambahkan *assignment* baru ke dalam *database* atau mengedit *assignment* yang ada
- *delete_assignment*: menghapus *assignment*
- *all_assignments*: menampilkan semua daftar *assignment* dan informasinya
- *new_assignment_id*: mengembalikan nilai int terkecil yang dapat digunakan sebagai id untuk *assignment* baru
- *all_problems*: mengembalikan *array* yang berisi semua masalah dalam *assignment* yang diberikan
- *problem_info*: mengembalikan baris *database* untuk masalah tertentu
- *assignment_info*: mengembalikan baris *database* untuk *assignment* tertentu
- *is_participant*: mengecek apakah pengguna terdaftar sebagai peserta pada *assignment* tertentu
- *increase_total_submits*: meningkatkan jumlah total *submit* sebanyak satu
- *set_moss_time*: mengedit "*Moss Update Time*" untuk *assignment* tertentu
- *get_moss_time*: mengembalikan "*Moss Update Time*" untuk *assignment* tertentu
- *save_problem_description*: menambahkan atau Memperbarui deskripsi masalah
- *_update_coefficients*: memperbarui koefisien yang terdapat pada *assignment* tertentu. Fungsi ini akan dipanggil oleh fungsi *add_assignment*

3.2.1.2 Notifications_Model.php

Pada file *Notifications_Model.php* terdapat beberapa fungsi yaitu:

- *get_all_notifications*: mengembalikan semua notifikasi
- *get_latest_notifications*: mengembalikan 10 notifikasi terkini
- *add_notification*: menambahkan notifikasi baru
- *update_notification*: mengedit notifikasi tertentu

- *delete_notification*: menghapus notifikasi tertentu
- *get_notification*: menampilkan notifikasi tertentu
- *have_new_notification*: mengecek apakah terdapat notifikasi setelah melewati waktu tertentu

3.2.1.3 *Queue_Model.php*

Pada file *Queue_Model.php* terdapat beberapa fungsi yaitu:

- *in_queue*: mengecek apakah sebuah submission telah berada dalam antrean
- *get_queue*: mengembalikan semua antrean *submission*
- *empty_queue*: mengosongkan antrean
- *add_to_queue*: memasukan *submission* ke dalam antrean
- *rejudge*: menambahkan *submission* ke dalam antrean untuk di-*rejudge*
- *rejudge_single*: menambahkan *submission* tunggal ke dalam antrean untuk di-*rejudge*
- *get_first_item*: mengembalikan *item* pertama dari antrean
- *remove_item*: menghapus *item* tertentu dari antrean
- *save_judge_result_in_db*: menyimpan hasil dari *judge* ke dalam *database*. Fungsi ini akan dipanggil oleh fungsi *Queueprocess* di *Controller*

3.2.1.4 *Scoreboard_Model.php*

Pada file *Scoreboard_Model.php* terdapat beberapa fungsi yaitu:

- *_generate_scoreboard*: menghasilkan *scoreboard* dari *Final Submissions* untuk *assignment* tertentu. Fungsi ini akan dipanggil oleh *update_scoreboard*
- *update_scoreboards*: mengupdate *cache scoreboard* dari semua *assignment*. Fungsi ini akan dipanggil setiap kali pengguna menghapus atau semua *assignment* pengguna dihapus.
- *update_scoreboard*: mengupdate *cache scoreboard* dari semua *assignment* dan menyimpan kode *html scoreboard* ke dalam *database*. Fungsi ini dipanggil setelah *judging/rejudging* sebuah *submission* dan ketika beberapa *settings* diubah pada *assignment* tertentu. *Setting* tersebut terdiri dari *Extra Time*, *Start Time*, *Finish Time*, *Coefficient's Rule* dan *Enable/Disable Scoreboard*.
- *get_scoreboard*: mengembalikan *cache scoreboard* untuk *assignment* tertentu sebagai teks *html*

3.2.1.5 *Settings_Model.php*

Pada file *Settings_Model.php* terdapat beberapa fungsi yaitu:

- *get_setting*: mengembalikan pengaturan tertentu
- *set_setting*: mengupdate pengaturan tertentu
- *get_all_settings*: menampilkan semua pengaturan
- *set_setting*: mengupdate seluruh pengaturan

3.2.1.6 *Submit_Model.php*

Pada file *Submit_Model.php* terdapat beberapa fungsi yaitu:

- *get_submission*: mengembalikan baris tabel sebuah *submission* tertentu
- *get_final_submissions*: mengembalikan baris tabel *final submission* para peserta dari sebuah *submission* tertentu
- *get_all_submissions*: mengembalikan baris tabel seluruh *submission*
- *count_final_submissions*: mengembalikan jumlah total *final submission* dari peserta tertentu
- *count_all_submissions*: mengembalikan jumlah total *submission* dari peserta tertentu
- *set_final_submission*: menentukan *submission* tertentu agar menjadi *final submission*
- *add_upload_only*: menambahkan hasil dari *submit "Upload only"* ke *database*

3.2.1.7 *User.php*

Pada file *User.php* terdapat beberapa fungsi yaitu:

- *select_assignment*: menetapkan *assignment* yang dipilih untuk username tertentu
- *save_widget_positions*: mengupdate posisi dari *dashboard widget* ke *database*
- *get_widget_positions*: mengembalikan posisi *dashboard widget* dari *database*

3.2.1.8 *User_Model.php*

Pada file *User_Model.php* terdapat beberapa fungsi yaitu:

- *have_user*: mengecek apakah pengguna dengan *username* yang diinput terdapat di *database*
- *user_id_to_username*: mengubah *user id* menjadi *username*
- *username_to_user_id*: mengubah *username* menjadi *user id*
- *have_email*: mengecek apakah pengguna dengan *email* yang diberikan terdapat di *database*
- *add_user*: menambahkan pengguna tunggal
- *add_users*: menambahkan beberapa pengguna
- *delete_user*: menghapus pengguna tertentu
- *delete_submissions*: menghapus semua *submission* pengguna tertentu
- *validate_user*: mengecek apakah *user* dan *password* yang diinput cocok untuk keperluan *login*
- *selected_assignment*: mengembalikan *assignment* yang dipilih
- *get_names*: mengembalikan nama dari *username* tertentu
- *update_profile*: memperbarui *user profile* (*Name, Email, Password, Role*)
- *send_password_reset_mail*: Menghasilkan *password reset key* dan mengirim *email* berisi link untuk mereset *password*
- *passchange_is_valid*: mengecek apakah *password reset key* sesuai

- *reset_password*: mereset *password* untuk *password reset key* yang diberikan
- *get_all_users*: menampilkan seluruh *user* yang ada pada *database*
- *get_user*: mengembalikan baris tabel *user* dengan *user id* tertentu
- *update_login_time*: memperbarui *First Login Time* dan *Last Login Time username* tertentu

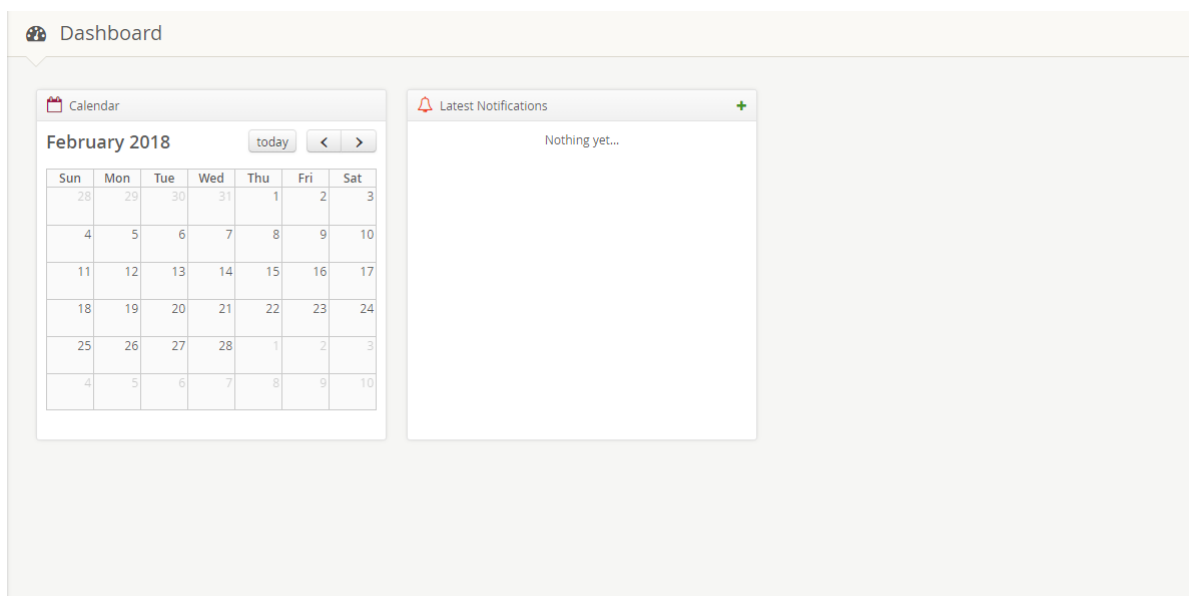
3.2.2 View

Pada perangkat lunak *Sharif Judge*, file view menggunakan *framework* aplikasi yaitu *Twig*. *Twig* merupakan sebuah *template engine modern* untuk *PHP*. Beberapa kelebihanannya adalah

- Cepat: *Twig* dapat mengcompile *template* ke dalam kode *HTML* dan dapat dioptimalkan. Dibandingkan dengan kode *PHP* biasa, *Twig* dapat menghasilkan kode *PHP* menjadi seminimum mungkin.
- Aman: *Twig* memiliki mode *sandbox* untuk mengevaluasi kode *template* yang tidak tepercaya. Hal ini memungkinkan *Twig* digunakan sebagai *template language* untuk banyak aplikasi dimana pengguna dapat memodifikasi desain *template* yang ada.
- Fleksibel: *Twig* didukung oleh *lexer* dan *parser* yang fleksibel. Hal ini memungkinkan para pengembang untuk menentukan tag dan filter khusus [6].

Direktori *view* perangkat lunak *Sharif Judge* terdapat pada *Sharif-Judge\application\views*. Di dalam *folder views*, terdapat beberapa *folder* yang memisahkan *file view* diantaranya adalah *folder error*, *pages* dan *templates*. *Folder error* berisikan tampilan ketika pengguna melakukan kesalahan seperti *404 Page Not Found* atau *Database Error*. *Folder template* berisikan tampilan dasar penyusun *Sharif Judge* seperti *Top Bar*, *Side Bar* dan *Header*. *Folder pages* berisikan tampilan utama *Sharif Judge*. Pada folder ini terdapat beberapa folder yaitu *admin* dan *authentication*. *Folder admin* berisikan tampilan yang hanya dapat dilihat oleh admin seperti tampilan *Settings*, *User*, *Install*, *Add Assignemnt*, *Add Notification* dan lain-lain. *Folder authentication* berisikan tampilan yang berhubungan dengan akses akun pengguna tampilan *Login*, *Register* dan *Reset Password*.

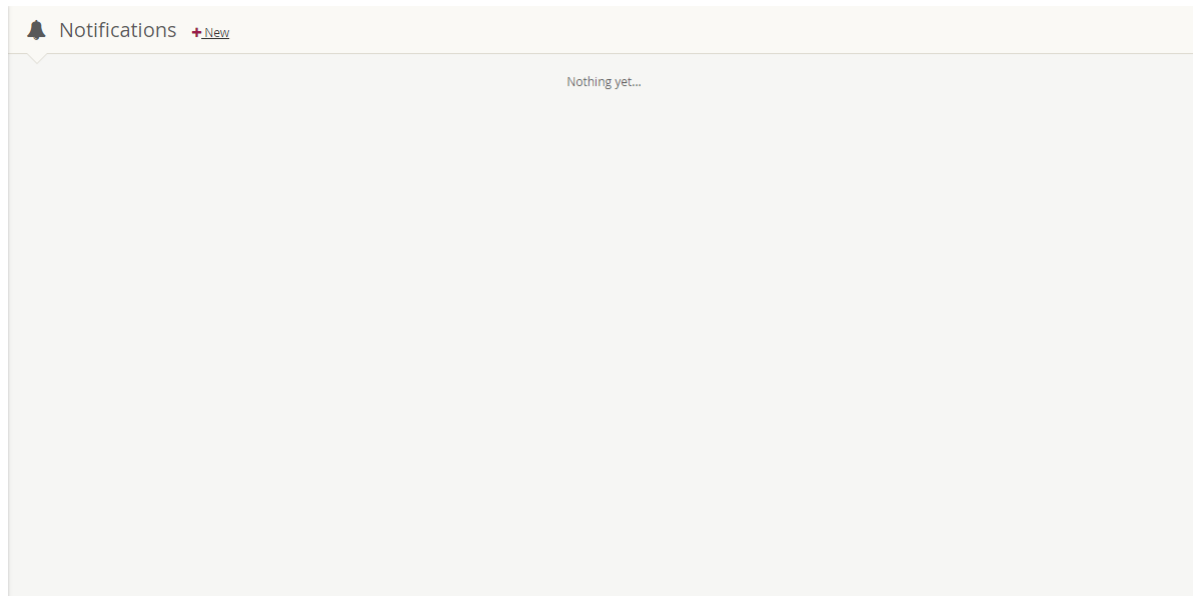
3.2.2.1 dashboard.twig



Gambar 3.9: Dashboard

Dashboard merupakan tampilan utama tepat setelah pengguna berhasil login pada perangkat lunak *Sharif Judge*. Pada tampilan dashboard, terdapat kalender dan tabel notifikasi. Kalender akan menampilkan keterangan *assignment* yang ada pada *Sharif Judge*, sedangkan tabel notifikasi akan berisikan 10 pengumuman terbaru yang telah dibuat oleh admin.

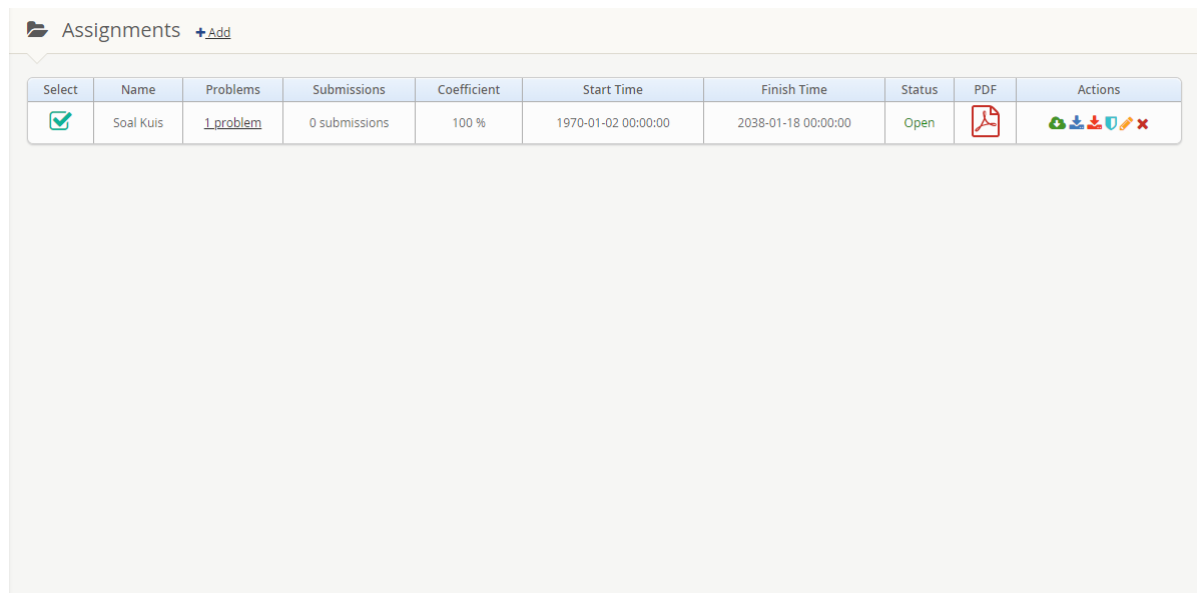
3.2.2.2 *notifications.twig*



Gambar 3.10: *Notifications*

Halaman notifikasi akan berisikan seluruh notifikasi yang telah dibuat oleh admin. Jika admin yang memasuki halaman tersebut, maka terdapat pilihan "*New*" dimana admin dapat menambahkan pengumuman baru untuk para pengguna *Sharif Judge*. Pengumuman tersebut akan muncul pada tabel notifikasi di *dashboard*.

3.2.2.3 *assignments.twig*

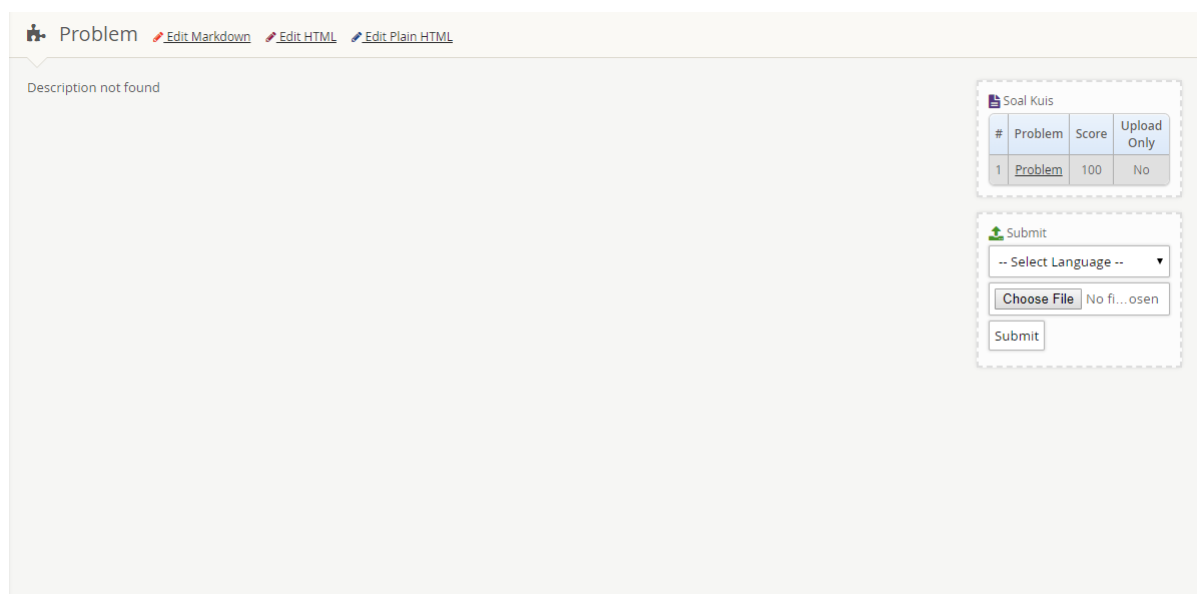


Select	Name	Problems	Submissions	Coefficient	Start Time	Finish Time	Status	PDF	Actions
<input checked="" type="checkbox"/>	Soal Kuis	1 problem	0 submissions	100 %	1970-01-02 00:00:00	2038-01-18 00:00:00	Open		

Gambar 3.11: *Assignments*

Halaman *assignments* akan berisikan seluruh *assignment* yang ada pada *Sharif Judge*. Pada halaman ini, pengguna dapat memilih *assignment* mana yang akan dikerjakan. Jika admin yang memasuki halaman tersebut, maka terdapat beberapa pilihan tambahan. Beberapa pilihan tambahan seperti fungsi "Add" akan mengarahkan admin ke halaman baru untuk menambah *assignment* dan beberapa fungsi lain untuk mengedit, menghapus atau mengunduh *assignment* yang sudah ada.

3.2.2.4 *problems.twig*



Problem [Edit Markdown](#) [Edit HTML](#) [Edit Plain HTML](#)

Description not found

#	Problem	Score	Upload Only
1	Problem	100	No

Submit

-- Select Language --

[Choose File](#) No file chosen

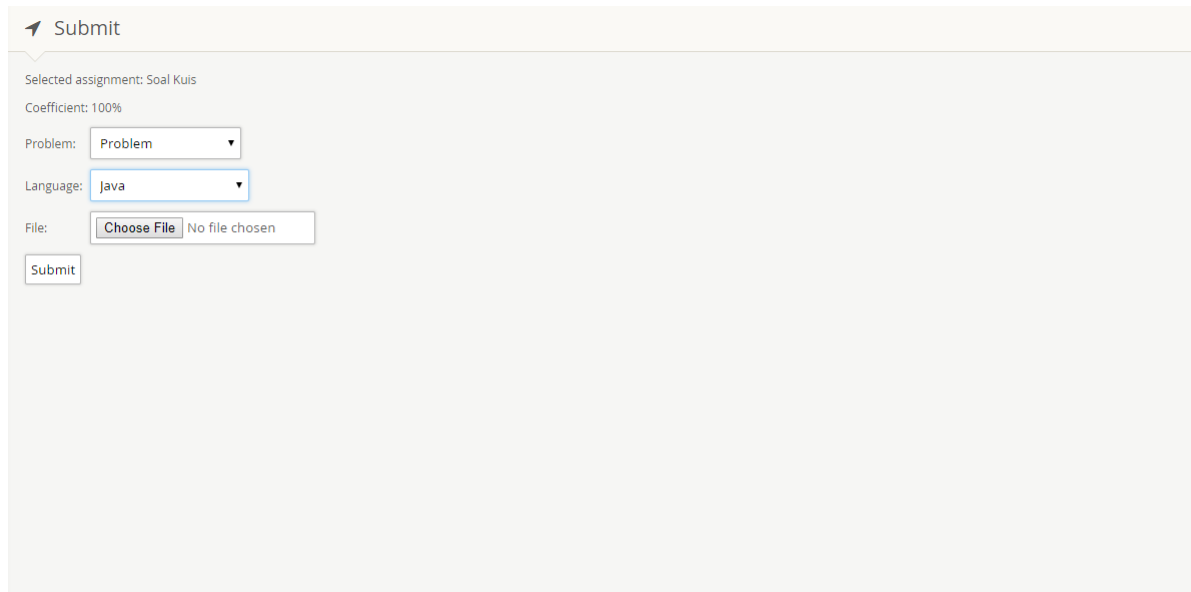
Submit

Gambar 3.12: *Problems*

Halaman *problems* akan berisikan masalah-masalah yang ada pada suatu *assignment*. Pada halaman ini, pengguna dapat melihat diskripsi pada masing-masing masalah. Selain melihat diskripsi masalah,

para pengguna juga dapat mengumpulkan jawaban dari setiap masalah tersebut.

3.2.2.5 *submit.twig*



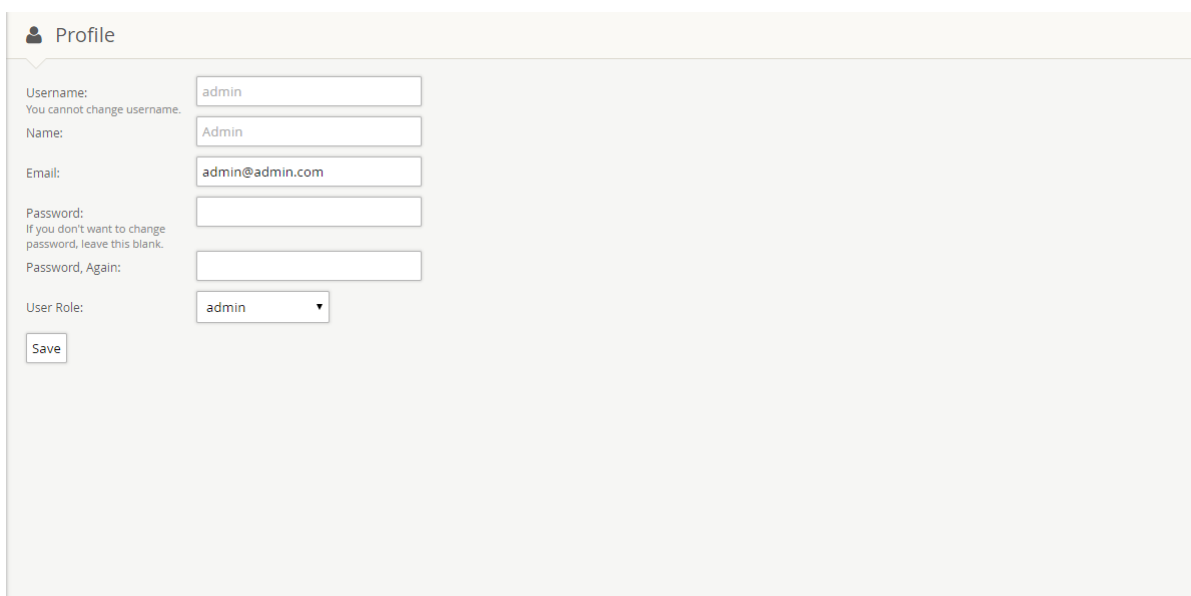
The screenshot shows a web form titled "Submit". It contains the following elements:

- Selected assignment: Soal Kuis
- Coefficient: 100%
- Problem: A dropdown menu with "Problem" selected.
- Language: A dropdown menu with "java" selected.
- File: A "Choose File" button and the text "No file chosen".
- A "Submit" button at the bottom left.

Gambar 3.13: *Submit*

Halaman *submit* berfungsi untuk mengumpulkan jawaban dari *assignment* yang telah dipilih pengguna sebelumnya. Pengguna dapat memilih masalah telah diselesaikan lalu memilih bahasa yang digunakan dan memilih file jawaban yang akan dikumpulkan. Setelah mengumpulkan jawaban, pengguna akan diarahkan ke halaman *All Submission* untuk melihat hasil dari jawaban yang telah dikumpulkan sebelumnya.

3.2.2.6 *profile.twig*



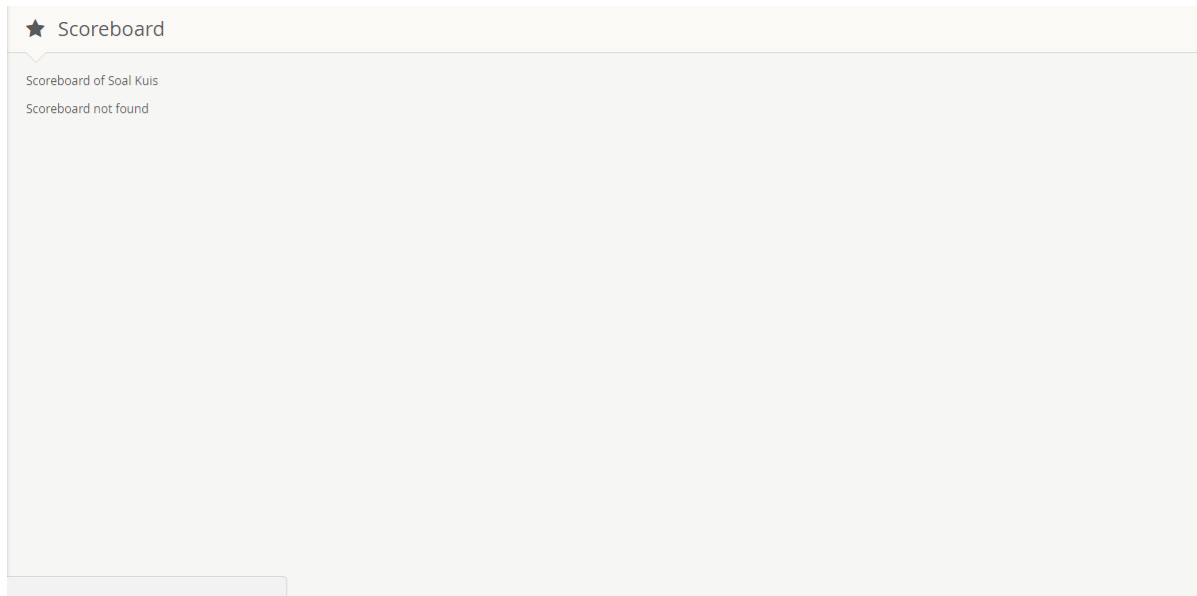
The screenshot shows a web form titled "Profile". It contains the following elements:

- Username: A text input field with "admin" entered. Below it, a note says "You cannot change username."
- Name: A text input field with "Admin" entered.
- Email: A text input field with "admin@admin.com" entered.
- Password: A text input field.
- Password, Again: A text input field.
- User Role: A dropdown menu with "admin" selected.
- A "Save" button at the bottom left.

Gambar 3.14: *Profile*

Halaman *profile* akan berisikan keterangan akun pengguna *Sharif Judge*. Pada halaman ini, pengguna dapat mengubah nama, *email* dan *password*. Jika admin yang memasuki halaman ini, maka terdapat tambahan pilihan yaitu *user role*. Admin dapat mengubah user role menjadi *head_instructor*, *instructor* dan *student*.

3.2.2.7 *scoreboard.twig*



Gambar 3.15: *Scoreboard*

Halaman *scoreboard* akan berisikan nilai seluruh pengguna *Sharif Judge* pada *assignment* tertentu. Nama pengguna yang muncul akan terurut secara menurun berdasarkan nilai *assignment* pengguna. Admin dapat menonaktifkan *scoreboard* dengan cara menghilangkan *checkbox* "Scoreboard" pada halaman *Add Assignment*.

3.2.2.8 *submission.twig*

Final	Problem	Submit Time	Score			Language	Status	Code
			Score	Delay %	Final Score			

Gambar 3.16: *All Submission*

Problem	Submit Time	Score			Language	Status	Code
		Score	Delay %	Final Score			

Gambar 3.17: *Final Submission*

File *submission.twig* akan terbagi menjadi dua halaman yaitu *All Submission* dan *Final Submission*. Pada halaman *All Submission*, pengguna dapat melihat seluruh jawaban yang telah dikumpulkan. Pengguna juga dapat memilih salah satu jawaban dari suatu masalah yang akan dijadikan jawaban akhir. Pada halaman *Final Submission*, pengguna dapat melihat seluruh jawaban akhir yang telah dipilih sebelumnya di halaman *All Submission*.

3.2.2.9 *settings.twig*

Settings [Help](#)

Server Time: Mar 21 - 17:51:26

Timezone: [list of timezones](#)

Week Start Day:

Full Path to **tester**:

Full Path to **assignments**:

Upload Size Limit (kB):

Output Size Limit (kB):
Sets a limit for size of output file generated by submitted code

Results Per Page:
In "All Submissions"
Enter 0 for no limit

Results Per Page:
In "Final Submissions"
Enter 0 for no limit

Registration: ☒ Open Public Registration.

Registration Code:
If you want to enable registration (above option), It is better to give a registration code to students in your class for validating registration. Enter 0 to disable.

Gambar 3.18: *Settings*

Halaman *settings* akan berisikan pengaturan yang ada pada *Sharif Judge*. Beberapa pengaturannya yaitu pengaturan *timezone*, direktori *assignments*, direktori *tester*, *email*, *sandboxing*, *shield* dan lain-lain.

3.2.2.10 *user.twig*

Users [Help](#) [Add Users](#) [Excel](#)

#	User ID	Username	Display Name	Email	Role	First Login	Last Login	Actions
1	1	admin	Admin	admin@admin.com	admin	2018-02-13 04:28:27	2018-03-21 17:51:24	Edit Delete Add
2	2	stillmen	Stillmen Vallian	stillmen.v@gmail.com	student	2018-03-11 11:49:57	2018-03-21 17:16:08	Edit Delete Add
3	3	dummy	Dummy Account	dummy@dummy.com	student	2018-03-11 11:53:26	2018-03-11 11:54:39	Edit Delete Add

Gambar 3.19: *User*

Halaman *user* akan berisikan list pengguna yang terdaftar pada *Sharif Judge*. Pada halaman ini, admin dapat melakukan beberapa aksi seperti menambah pengguna, menghapus pengguna, melihat hasil jawaban pengguna, menghapus jawaban pengguna dan mengubah data diri pengguna.

3.2.2.11 *add_user.twig*

+ Add Users [Help](#)

You can use this field to add multiple users at the same time.

- Usernames may contain lowercase letters or numbers and must be between 3 and 20 characters in length.
- Passwords must be between 6 and 30 characters in length.
- If you want to send passwords by email, do not add too many users at one time. This may result in mail delivery fail.

☐ Send usernames and passwords by email (Waits second(s) before sending each email, so please be patient).

```
# Lines starting with a # sign are comments.
# Each line (except comments) represents a user.
# The syntax of each line is:
#
# USERNAME,EMAIL,DISPLAY-NAME,PASSWORD,ROLE
#
# Roles: admin head_instructor instructor student
# You can use RANDOM[n] for password to generate random n-digit password.
```

Gambar 3.20: *Add User*

Halaman *add user* berfungsi untuk menambah peserta pada *Sharif Judge*. Pada halaman ini, admin dapat memasukan informasi pengguna yang ingin didaftarkan pada *Sharif Judge*. Informasi tersebut berupa *username*, *email*, *password* dan *role*.

3.2.2.12 *add_notification.twig*

+ New Notification

Title:

Text:

File Edit Insert View Format Tools

Formats B I U Text Color Background Color Bulleted List Numbered List Link Unlink Source Code

p

Gambar 3.21: *Add Notification*

Halaman *add notification* berfungsi untuk menambah pengumuman pada *Sharif Judge*. Pada halaman ini, terdapat beberapa *form* seperti judul pengumuman dan isi dari pengumuman tersebut.

3.2.2.13 *add_assignment.twig*

Add Assignment [Help](#)

Assignment Name:

Start Time:

Finish Time:

Extra Time (minutes):
Extra time for late submissions.

Participants:
Enter username of participants here (comma separated). Only these users are able to submit. You can use keyword "ALL".

Tests and Descriptions (zip file): No file chosen
[Use this structure](#)

PDF File: No file chosen
PDF File of Assignment

☐ Open Open or close this assignment

☐ Scoreboard Check this to enable scoreboard

☐ Java Exceptions Check this to show Java exceptions to users

☐ Archived Assignment Check this to make an archived assignment

Coefficient rule (2)
PHP script without <?php ?> tags

```
/*
 * Put coefficient (from 100) in variable $coefficient.
 * You can use variables $extra_time and $delay.
 * $extra_time is the total extra time given to users
 * (in seconds) and $delay is number of seconds passed
 * from finish time (can be negative).
 * In this example, $extra_time is 172800 (2 days):
 */

if ($delay <= 0)
    // no delay
    $coefficient = 100;
elseif ($delay <= 3600)
    // delay less than 1 hour
    $coefficient = ceil(100 - ((30 * $delay) / 3600));
elseif ($delay <= 86400)
    // delay more than 1 hour and less than 1 day
    $coefficient = 70;
```

Problems [+](#)

	Name	Score	Time Limit (ms)			Memory Limit (kB)	Allowed Languages (2)	Diff Command (2)	Diff Argument (2)	Upload Only (2)
			C/C++	Python	Java					
1	Problem	100	500	1500	2000	50000	C,C++,Python 2,Python 3,J	diff	-bB	<input type="checkbox"/>

Gambar 3.22: *Add Assignment*

Halaman *add assignment* berfungsi untuk menambah *assignment* pada *Sharif Judge*. Pada halaman ini, *admin* dan *head instructor* dapat mengatur beberapa pengaturan *assignment* tersebut. Beberapa pengaturan seperti nama, waktu mulai, waktu akhir, *scoreboard* dan lain-lain. *Admin* juga dapat mengunggah diskripsi serta file PDF untuk *assignment* tersebut.

3.2.2.14 *rejudge.twig*

Rejudge

Selected Assignment: Soal Kuis

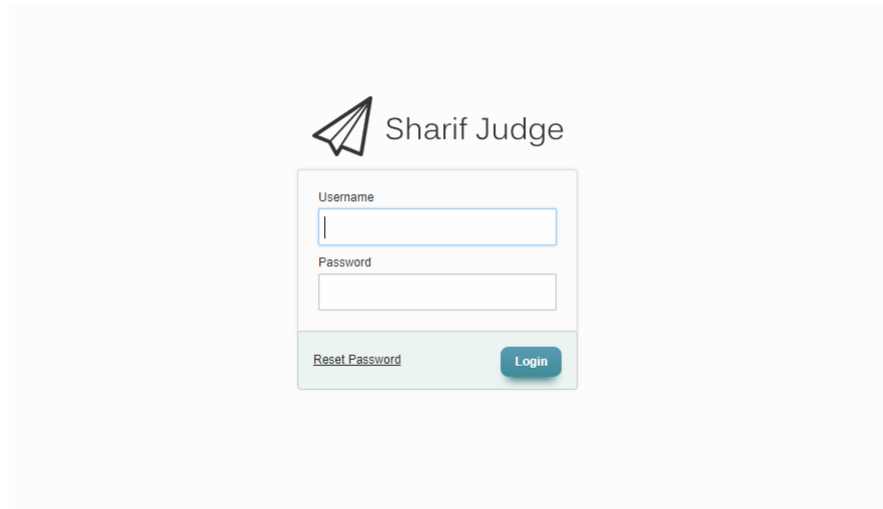
By clicking on rejudge, all submissions of selected problem will change to **PENDING** state. Then Sharif Judge rejudges them one by one.

If you want to rejudge a single submission, you can click on rejudge button in [All Submissions](#) or [Final Submissions](#) page.

Gambar 3.23: *Rejudge*

Halaman *rejudge* berfungsi untuk menilai ulang hasil pekerjaan seluruh peserta *Sharif Judge*. Pada halaman ini, *admin* dan *head instructor* dapat menilai ulang setiap masalah pada *assignment* yang dipilih.

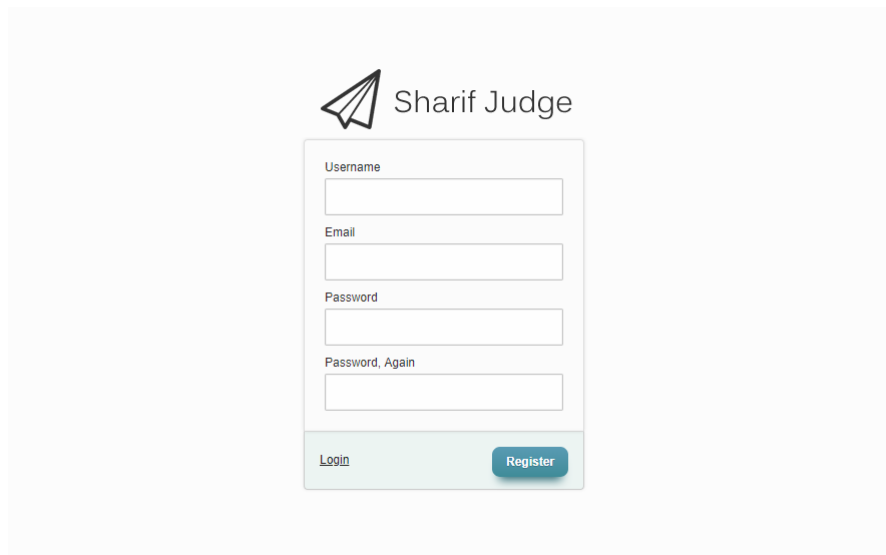
3.2.2.15 *login.twig*

The image shows a login form for 'Sharif Judge'. At the top, there is a paper airplane icon followed by the text 'Sharif Judge'. Below this, the form contains two input fields: 'Username' and 'Password'. Under the 'Password' field, there is a link that says 'Reset Password'. At the bottom right of the form is a blue button labeled 'Login'.

Gambar 3.24: *login*

Halaman login merupakan halaman pertama yang akan tampil ketika pengguna membuka *Sharif Judge*. Untuk dapat menggunakan *Sharif Judge*, para pengguna harus memasukkan kombinasi *username* dan *password* yang tepat. Jika pengguna berhasil *login*, maka akan diarahkan ke halaman *dashboard*. Jika pengguna gagal *login*, maka akan muncul pesan kesalahan "*Incorrect username or password*".

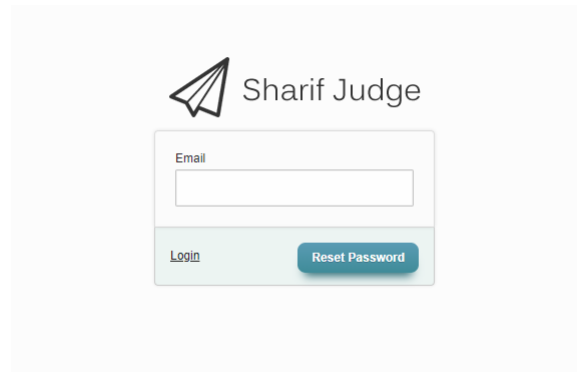
3.2.2.16 *register.twig*

The image shows a registration form for 'Sharif Judge'. At the top, there is a paper airplane icon followed by the text 'Sharif Judge'. Below this, the form contains four input fields: 'Username', 'Email', 'Password', and 'Password, Again'. At the bottom left of the form is a link that says 'Login', and at the bottom right is a blue button labeled 'Register'.

Gambar 3.25: *Register*

Halaman *register* berfungsi untuk mendaftar sebagai peserta *Sharif Judge*. Umumnya halaman ini tidak tersedia karena pengguna *Sharif Judge* telah ditentukan sebelumnya oleh *admin* dan *head instructor*. Halaman ini akan muncul jika *admin* mengaktifkan fitur *Open Public Registration* pada halaman *Settings*.

3.2.2.17 *lost.twig*



Gambar 3.26: *Lost*

Halaman *lost* berfungsi untuk para peserta yang lupa kombinasi *username* dan *password*. Peserta harus memasukan *email* yang didaftarkan pada *Sharif Judge*. *Sharif Judge* akan mengirimkan *link* untuk mereset *password* ke *email* yang telah dimasukan sebelumnya.

3.2.3 *Controller*

Direktori *controller* perangkat lunak *Sharif Judge* terdapat pada *Sharif-Judge\application\controllers*. Di dalam *folder controllers*, terdapat beberapa *file controller* yang berisikan fungsi-fungsi sebagai perantara antara *model*, *view*, dan *resource* lainnya yang dibutuhkan untuk memproses *HTTP request*.

3.2.3.1 *Assignments.php*

Pada file *Assignments.php* terdapat beberapa fungsi yaitu:

- *index*: mempersiapkan data yang dibutuhkan untuk halaman *assignments.twig*. Data yang dipersiapkan, diambil menggunakan fungsi-fungsi yang terdapat pada *Assignment_model.php*
- *select*: memilih *assignment* yang ada pada *Sharif Judge*
- *pdf*: mengunduh file pdf atau deskripsi masalah pada *assignment* tertentu
- *downloadtestsdesc*: mengompres dan mengunduh test data dan deskripsi *assignment* tertentu
- *download_submissions*: mengompres dan mengunduh jawaban akhir para peserta pada *assignment* tertentu
- *delete*: menghapus sebuah *assignment*
- *add*: mengambil input dari user untuk menambah atau mengubah *assignment*
- *_add*: menambah atau mengubah *assignment*

3.2.3.2 *Dashboard.php*

Pada file *Dashboard.php* terdapat beberapa fungsi yaitu:

- *index*: mempersiapkan data yang dibutuhkan untuk halaman *dashboard.twig*. Data yang dipersiapkan, diambil menggunakan beberapa fungsi yang terdapat pada *Assignment_model.php*, *Settings_model.php* dan *Notification_model.php*
- *widget_positions*: menyimpan posisi *widget* pada *Dashboard* pengguna

3.2.3.3 *Install.php*

Pada file *Install.php* terdapat beberapa fungsi yaitu:

- *index*: fungsi ini akan membuat table-table yang dibutuhkan oleh *Sharif Judge* ke *database* yang telah ditentukan

3.2.3.4 *Login.php*

Pada file *Login.php* terdapat beberapa fungsi yaitu:

- *__registration_code*: memeriksa apakah kode pendaftaran yang dimasukkan sudah benar atau tidak
- *index*: memvalidasi kombinasi antara *username* dan *password* yang telah dimasukan pengguna
- *register*: mempersiapkan *form* registrasi dan memproses tampilan pada *register.twig*
- *logout*: keluar dari *Sharif Judge* dan mengalihkan pengguna ke halaman *login*
- *lost*: mempersiapkan *form* lupa *password* dan memproses tampilan pada *lost.twig*

3.2.3.5 *Notification.php*

Pada file *Notification.php* terdapat beberapa fungsi yaitu:

- *index*: mempersiapkan data yang dibutuhkan untuk halaman *notification.twig*. Data yang dipersiapkan, diambil menggunakan beberapa fungsi yang terdapat pada *Assignment_model.php* dan *Notification_model.php*
- *add*: menambah pengumuman pada *Sharif Judge*
- *edit*: mengubah pengumuman yang ada pada *Sharif Judge*
- *delete*: menghapus pengumuman yang ada pada *Sharif Judge*

3.2.3.6 *Problems.php*

Pada file *Problems.php* terdapat beberapa fungsi yaitu:

- *index*: menampilkan deskripsi *problem* yang diberikan
- *edit*: mengubah diskripsi *problem* yang ada pada *Sharif Judge*

3.2.3.7 *Profile.php*

Pada file *Profile.php* terdapat beberapa fungsi yaitu:

- *index*: mempersiapkan *form profile* yang berfungsi untuk mengubah informasi dari pengguna *Sharif Judge*
- *__password_check*: mengecek apakah *password* yang dibuat oleh pengguna *Sharif Judge* memenuhi syarat. Syarat *password* tersebut yaitu minimal terdiri dari 6 karakter.
- *__password_again_check*: mengecek apakah '*password again*' yang dimasukan sama dengan *password* yang telah dimasukan sebelumnya
- *__email_check*: mengecek apakah *email* yang dimasukan pengguna telah digunakan pengguna lain
- *__role_check*: memvalidasi *user role*

3.2.3.8 *Queueprocess.php*

Pada file *Queueprocess.php* terdapat beberapa fungsi yaitu:

- *run*: fungsi utama untuk memproses antrean (*queue*) dimana fungsi ini akan menjudge antrean satu demi satu

3.2.3.9 *Rejudge.php*

Pada file *Rejudge.php* terdapat beberapa fungsi yaitu:

- *index*: mempersiapkan data yang dibutuhkan untuk halaman *rejudge.twig*. Data yang dipersiapkan, diambil menggunakan beberapa fungsi yang terdapat pada *Assignment_model.php*
- *rejudge_single*: menilai ulang jawaban peserta pada satu masalah tertentu

3.2.3.10 *Scoreboard.php*

Pada file *Scoreboard.php* terdapat beberapa fungsi yaitu:

- *index*: mempersiapkan data yang dibutuhkan untuk halaman *scoreboard.twig*. Data yang dipersiapkan, diambil menggunakan beberapa fungsi yang terdapat pada *Assignment_model.php* dan *Scoreboard_model.php*

3.2.3.11 *Server_time.php*

Pada file *Server_time.php* terdapat beberapa fungsi yaitu:

- *index*: menampilkan waktu server yang berfungsi untuk sinkronisasi waktu server

3.2.3.12 *Settings.php*

Pada file *Settings.php* terdapat beberapa fungsi yaitu:

- *index*: mempersiapkan data yang dibutuhkan untuk halaman *settings.twig*. Data yang dipersiapkan, diambil menggunakan beberapa fungsi yang terdapat *settings_model.php*
- *update*: menyimpan pengaturan yang telah diubah pada halaman *settings.twig*

3.2.3.13 *Submission.php*

Pada file *Submission.php* terdapat beberapa fungsi yaitu:

- *_download_excel*: menggunakan library *PHPExcel* untuk menghasilkan file excel dari *submission* pada *assignment* tertentu
- *final_excel*: mengunduh file excel dari *submission* yang telah ditandai sebagai jawaban akhir
- *all_excel*: mengunduh file excel dari seluruh *submission*
- *the_final*: mempersiapkan dan menampilkan data yang dibutuhkan untuk halaman *submission.twig* bagian *Final Submissions*
- *all*: mempersiapkan dan menampilkan data yang dibutuhkan untuk halaman *submission.twig* bagian *All Submissions*
- *select*: memilih *submission* tertentu untuk dijadikan jawaban akhir untuk masalah tertentu
- *download_file*: mengunduh file jawaban yang telah dikumpulkan sebelumnya

3.2.3.14 *Submit.php*

Pada file *Submit.php* terdapat beberapa fungsi yaitu:

- *__language_to_type*: mengubah bahasa pemrograman menjadi ekstensi bahasa pemrograman tersebut. Contoh: "c++" akan diubah menjadi "cpp"
- *__match*: memastikan ekstensi file jawaban yang dikumpul sesuai dengan bahasa pemrograman yang dipilih
- *__check_language*: memastikan bahasa pemrograman yang digunakan dapat ditangani oleh *Sharif Judge*
- *index*: menyiapkan data-data jawaban dan mengumpulkannya ke *Sharif Judge*
- *__upload*: menyimpan kode jawaban yang dikumpulkan dan menambahkannya ke antrian untuk dinilai

3.2.3.15 *User.php*

Pada file *User.php* terdapat beberapa fungsi yaitu:

- *index*: menyiapkan data yang dibutuhkan untuk *halaman users.twig*. Data yang dipersiapkan, diambil menggunakan beberapa fungsi yang terdapat *assignment_model.php* dan *user_model.php*
- *add: controller* untuk menambahkan pengguna baru
- *delete: controller*: untuk menghapus pengguna yang ada
- *delete_submissions: controller* untuk menghapus *submission* pengguna tertentu
- *list_excel*: mengunduh file *excel* dari *list user*

BAB 4

PERANCANGAN

Bab ini membahas tentang perancangan setiap fitur yang akan diimplementasi pada perangkat lunak *Sharif Judge*.

4.1 Mengganti method *shell_exec("rm ...")* menjadi *unlink()*

Method *shell_exec("rm ...")* yang memiliki fungsi untuk menghapus sebuah file terdapat pada file *controller Assignment.php* tepatnya di baris kode 425 dan 473

Assignments.php

```
...
423    // Upload Tests (zip file)
424
425    shell_exec('rm -f ' . $assignments_root . '/*.zip');
426    $config = array(
...
472    foreach($old_pdf_files as $old_name)
473        shell_exec("rm -f $old_name");
474    $this->messages[] = array(
...

```

Fungsi *shell_exec("rm ...")* pada baris 425 dan 473 akan diubah menggunakan fungsi *unlink()* menjadi seperti berikut

Assignments.php

```
...
423    // Upload Tests (zip file)
424
425    unlink($assignments_root . '/*.zip');
426    $config = array(
...
472    foreach($old_pdf_files as $old_name)
473        unlink($old_name);
474    $this->messages[] = array(
...

```

4.2 Menambahkan method koneksi ke *database*

Method koneksi ke *database* akan ditambahkan pada file *controller Queueprocess.php*.

Queueprocess.php

```

...
133
134     // Save the result
135     $this->queue_model->save_judge_result_in_db($submission, $type);
...

```

Method rekoneksi yang digunakan yaitu *\$this->db->reconnect()*. *Method* ini diletakan pada baris 134 tepat sebelum *Sharif Judge* menyimpan hasil *judge*. Hal tersebut dilakukan untuk menghindari *connection times out* akibat pengujian yang memakan waktu lama.

Queueprocess.php

```

...
133
134     //reconnect to database incase we have run test for a long time.
135     $this->db->reconnect();
136
137     // Save the result
138     $this->queue_model->save_judge_result_in_db($submission, $type);
...

```

4.3 Membatasi soal (deskripsi & PDF) hanya bisa diunduh saat *assignment "open"* dan setelah waktu mulai

Fungsi untuk mengunduh soal (deskripsi & PDF) terdapat pada *controller Assignment.php* tepatnya di baris kode 100.

Assignments.php

```

...
97     /**
98     * Download pdf file of an assignment (or problem) to browser
99     */
100    public function pdf($assignment_id, $problem_id = NULL)
101    {
102        // Find pdf file
103        if ($problem_id === NULL)
104            $pattern = rtrim($this->settings_model->get_setting('
assignments_root'), '/') . "/assignment_{ $assignment_id }/*.pdf";
105        else
106            $pattern = rtrim($this->settings_model->get_setting('
assignments_root'), '/') . "/assignment_{ $assignment_id }/p{ $problem_id }/*.pdf";
107        $pdf_files = glob($pattern);
108        if ( ! $pdf_files )
109            show_error("File not found");
110
111        // Download the file to browser
112        $this->load->helper('download')->helper('file');
113        $filename = shj_basename($pdf_files[0]);
114        force_download($filename, file_get_contents($pdf_files[0]), TRUE)
;
115    }
...

```

Selain membatasi soal (deskripsi & PDF) hanya dapat diunduh saat *assignment "open"* dan setelah waktu mulai, pada fungsi ini juga ditambahkan fitur lain. Fitur lain tersebut yaitu membatasi soal hanya dapat diunduh oleh peserta yang terdaftar sebagai "*participant*" dan soal tidak dapat diunduh setelah melewati batas waktu pengumpulan. Rancangan algoritma kode yang akan digunakan yaitu

- Membuat atribut tambahan untuk menyimpan informasi waktu selesai, waktu mulai dan waktu tambahan sebuah assignment.
- Jika atribut "*open*" pada *assignment* tidak memiliki nilai, maka munculkan pesan *error "Selected assignment has been closed."*
- Jika pengguna tidak terdaftar sebagai "*participant*" dalam *assignment* yang dipilih, maka munculkan pesan error "*You are not registered for submitting.*"
- Jika waktu sekarang telah melewati batas waktu selesai + waktu tambahan, maka munculkan pesan *error "Selected assignment has finished."*
- Jika waktu sekarang belum melewati waktu mulai, maka munculkan pesan *error "Selected assignment has not started."*

Berikut hasil pengimplementasian rancangan algoritma di atas ke dalam kode program *Assignments.php*

```
...
97      /**
98      * Download pdf file of an assignment (or problem) to browser
99      */
100     public function pdf($assignment_id, $problem_id = NULL)
101     {
102         $finishtime = strtotime($this->assignment_model->assignment_info(
103             $assignment_id)['finish_time']);
104         $starttime = strtotime($this->assignment_model->assignment_info(
105             $assignment_id)['start_time']);
106         $extratime = $this->assignment_model->assignment_info(
107             $assignment_id)['extra_time'];
108
109         // Find pdf file
110         if ($problem_id === NULL)
111             $pattern = rtrim($this->settings_model->get_setting('
112                 assignments_root'), '/') . "/assignment_{ $assignment_id }/*.pdf";
113         else
114             $pattern = rtrim($this->settings_model->get_setting('
115                 assignments_root'), '/') . "/assignment_{ $assignment_id }/p{ $problem_id }/*.pdf";
116         $pdf_files = glob($pattern);
117         if ( ! $pdf_files )
118             show_error("File not found");
119         elseif ( !$this->assignment_model->assignment_info($assignment_id)
120             ['open'])
121             show_error('Selected assignment has been closed.');
```

```

118         elseif ( shj_now() > $finishtime + $extratime)
119             show_error('Selected assignment has finished.');
```

```

120         elseif ( shj_now() < $starttime)
121             show_error('Selected assignment has not started.');
```

```

122
123         // Download the file to browser
124         $this->load->helper('download')->helper('file');
```

```

125         $filename = shj_basename($pdf_files[0]);
126         force_download($filename, file_get_contents($pdf_files[0]), TRUE)
127     ;
128 }
129 ...

```

4.4 Mensupport *file* dengan ekstensi TXT

Untuk dapat mensupport *file* dengan ekstensi TXT pada perangkat lunak *Sharif Judge*, diperlukan penambahan dan perubahan kode pada beberapa *file*. Beberapa *file* tersebut antara lain *controller Submit.php*, *model Assignment_model.php*, *view submissions.twig* dan *file* bantuan *shj_helper.php* yang terdapat pada direktori *Sharif-Judge\application\helper*. Berikut beberapa baris potongan kode program

Submit.php

```

...
58         case 'java': return 'java';
59         case 'zip': return 'zip';
60         case 'pdf': return 'pdf';
61         default: return FALSE;
62     }
...
76         case 'java': return ($extension==='java'?TRUE:FALSE);
77         case 'zip': return ($extension==='zip'?TRUE:FALSE);
78         case 'pdf': return ($extension==='pdf'?TRUE:FALSE);
79     }
...
88     if ($str=='0')
89         return FALSE;
90     if (in_array( strtolower($str),array('c', 'c++', 'python 2', 'python 3', '
java', 'zip', 'pdf'))))
91         return TRUE;
92     return FALSE;
...

```

Assignment_model.php

```

...
100     $item2 = strtolower($item);
101     if ( ! in_array($item2, array('c','c++','python 2','python 3','java','zip
','pdf'))))
102         continue;
...

```

shj_helper.php


```

...
81     case 'java': return 'java';
82     case 'zip': return 'zip';
83     case 'pdf': return 'pdf';
84     default: return FALSE;
...
104    case 'java': return 'Java';
105    case 'zip': return 'Zip';
106    case 'pdf': return 'PDF';
107    default: return FALSE;
...

```

submissions.twig

```

...
160     <td>
161         {% if submission.file_type == 'zip' or submission.file_type == '
pdf' %}
162             <div class="btn shj-orange" data-type="download">Download
</div>
163         {% else %}
164             <div class="btn shj-orange" data-type="code" >Code</div>
165         {% endif %}
166     </td>
...

```

Penambahan dan perubahan kode dilakukan setelah baris 60, 78 dan 90 pada *controller Submit.php*. Berikut hasil penambahan dan perubahan kode

```

...
58         case 'java': return 'java';
59         case 'zip': return 'zip';
60         case 'pdf': return 'pdf';
61         case 'txt': return 'txt';
62         default: return FALSE;
63     }
...
77         case 'java': return ($extension==='java'?TRUE:FALSE);
78         case 'zip': return ($extension==='zip'?TRUE:FALSE);
79         case 'pdf': return ($extension==='pdf'?TRUE:FALSE);
80         case 'txt': return ($extension==='txt'?TRUE:FALSE);
81     }
...
90     if ($str=='0')
91         return FALSE;
92     if (in_array( strtolower($str),array('c', 'c++', 'python 2', 'python 3', '
java', 'zip', 'pdf', 'txt')))
93         return TRUE;
94     return FALSE;
...

```

Perubahan kode dilakukan di baris 101 pada model *Assignment_model.php*. Berikut hasil perubahan kode

Assignment_model.php

```
...
100     $item2 = strtolower($item);
101     if ( ! in_array($item2, array('c','c++','python 2','python 3','java','zip',
    ', 'pdf','txt'))))
102         continue;
...
```

Penambahan kode dilakukan setelah baris 83 pada file bantuan *shj_helper.php*. Berikut hasil penambahan kode

shj_helper.php

```
...
81     case 'java': return 'java';
82     case 'zip': return 'zip';
83     case 'pdf': return 'pdf';
84     case 'txt': return 'txt';
85     default: return FALSE;
...
105    case 'java': return 'Java';
106    case 'zip': return 'Zip';
107    case 'pdf': return 'PDF';
108    case 'txt': return 'TXT';
109    default: return FALSE;
...
```

Perubahan kode dilakukan pada baris 161 pada *view submissions.twig*. Berikut hasil perubahan kode

submissions.twig

```
...
160     <td>
161         {% if submission.file_type == 'zip' or submission.file_type == '
    pdf' or submission.file_type == 'txt' %}
162             <div class="btn shj-orange" data-type="download">Download
    </div>
163         {% else %}
164             <div class="btn shj-orange" data-type="code" >Code</div>
165         {% endif %}
166     </td>
...
```

4.5 Membuat halaman *Logs* yang mencatat aktivitas *login* pengguna

Agar halaman *Logs* dapat berjalan dengan baik, perlu ditambahkan tabel baru pada *database Sharif Judge*. Tabel baru tersebut akan bernama *shj_logins*.

Tabel 4.1: Perancangan Tabel *shj_logins*

Atribut	Tipe Data	Ukuran	Default
<i>login_id</i> (PK*)	int	11	None
<i>username</i>	varchar	20	None
<i>ip_address</i>	varchar	15	None
<i>timestamp</i>	timestamp	11	current_timestamp
<i>last_24h_login_id</i>	int	11	null

*PK = *Primary Key*.

Keterangan atribut:

1. *login_id*: sebagai penanda yang membedakan setiap *login* peserta satu dengan yang lain. Memiliki *length default* int dari *phpMyAdmin* yaitu 11. Atribut *login_id* merupakan *primary key* karena id harus unik agar setiap *login* peserta dapat dibedakan. Atribut ini juga bersifat *auto increment*.
2. *username*: *username* peserta yang berhasil *login* pada *Sharif Judge*. Memiliki *length varchar* 20 karena *length username* pada tabel *shj_users* adalah 20.
3. *ip_address*: *ip address* peserta yang berhasil *login* pada *Sharif Judge*. Memiliki *length varchar* 15 karena *length* maksimal dari *ip address protocol version 4 (IPv4)* adalah 15. Contoh: 202.100.123.255
4. *timestamp*: waktu peserta saat berhasil *login* pada *Sharif Judge*. Menggunakan tipe data *timestamp* yang akan mencatat waktu *login* dengan format YYYY-MM-DD HH:MM:SS. Contoh: 2018-04-06 18:15:43
5. *last_24h_login_id*: id *login* peserta yang berhasil *login* pada *Sharif Judge* namun menggunakan *ip address* berbeda dalam waktu 24 jam terakhir.

Selain tabel diatas, halaman logs juga akan ditambahkan *model*, *view* dan *controller*.

1. Model

Model untuk halaman *logs* akan bernama *Logs_model.php*. Berikut adalah perincian fungsi yang terdapat dalam rancangan *model Logs_model.php*.

Tabel 4.2: Perincian fungsi *insert_to_logs*

Nama Method	<i>insert_to_logs</i>
Parameter Input	<i>\$username</i> dan <i>\$ip_address</i>
Parameter Output	-
Tabel yang berhubungan	<i>shj_logins</i>
Deskripsi	Proses untuk memasukan <i>logs</i> pengguna <i>Sharif Judge</i>
Algoritma	<ul style="list-style-type: none"> • mengecek dan menghapus <i>logs</i> pada tabel <i>shj_logins</i> yang <i>timestampnya</i> lebih dari 24 jam • mengecek entri <i>login</i> terakhir untuk <i>\$username</i> yang menggunakan <i>IP address</i> tidak sama dengan <i>\$ip_address</i> • jika tidak memiliki hasil, maka tambahkan entri baru menggunakan <i>\$username</i> dan <i>\$ip_address</i> tersebut • jika memiliki hasil, maka tambahkan entri baru menggunakan <i>\$username</i> dan <i>\$ip_address</i> serta <i>last_24h_login_id</i> diisi dengan <i>login_id</i> sebelumnya

Tabel 4.3: Perincian fungsi *get_all_logs*

Nama Method	<i>get_all_logs</i>
Parameter Input	-
Parameter Output	semua entri logs dari tabel <i>shj_logins</i>
Tabel yang berhubungan	<i>shj_logins</i>
Deskripsi	Proses untuk mengembalikan entri <i>logs</i> yang terdapat pada tabel <i>shj_logins</i>
Algoritma	<ul style="list-style-type: none"> • mengembalikan seluruh entri logs yang terdapat pada tabel <i>shj_logins</i> dalam bentuk <i>array</i>

2. View

View untuk halaman *logs* akan bernama *logs.twig*. Menu halaman *logs* akan terletak di paling bawah menu lainnya dan akan bernama '*24-hour log*'. Berikut adalah rancangan tampilan halaman *logs*.

#	▲ Login ID	Username	IP Address	Login Time	Log from different IP (< 24 hours)
1	180	admin	202.10.10.1	2018-04-07 14:58:22	179
2	179	admin	202.10.10.10	2018-04-07 14:57:14	
3	178	student1	202.10.10.9	2018-04-07 14:55:0	
4	177	student2	202.10.10.11	2018-04-07 14:50:11	

Gambar 4.1: Rancangan tampilan halaman *logs*

3. Controller

Controller untuk halaman *logs* akan bernama *Logs.php*. Berikut adalah perincian fungsi yang terdapat dalam rancangan *controller Logs.php*.

Tabel 4.4: Perincian fungsi *consturct__*

Nama <i>Method</i>	<i>consturct__</i>
Parameter <i>Input</i>	-
Parameter <i>Output</i>	-
Tabel yang berhubungan	-
Deskripsi	membatasi pengguna yang dapat mengakses halaman <i>logs</i>
Algoritma	<ul style="list-style-type: none"> • mengecek <i>session</i> pengguna yang akan mengakses halaman <i>logs</i> • jika <i>session</i> tidak berstatus '<i>logged_in</i>', maka pengguna akan dialihkan ke halaman <i>login</i> • mengecek <i>role</i> pengguna yang akan mengakses halaman <i>logs</i> • jika <i>role</i> pengguna bukan <i>admin</i>, maka pengguna akan dialihkan ke halaman '<i>404 Not Found</i>'

Tabel 4.5: Perincian fungsi *index*

Nama <i>Method</i>	<i>index</i>
Parameter <i>Input</i>	-
Parameter <i>Output</i>	-
Tabel yang berhubungan	<i>shj_logins</i>
Deskripsi	Proses untuk memuat seluruh entri <i>logs</i> pada halaman <i>logs.twig</i>
Algoritma	<ul style="list-style-type: none"> • memuat data <i>logs</i> menggunakan fungsi <i>get_all_logs</i> dari <i>model Logs_model.php</i> • memproses data untuk tampilan <i>logs.twig</i>

4.6 Menambahkan parameter "*Display Name*" pada pendaftaran peserta *Sharif Judge*

Untuk dapat menambahkan parameter "*Display Name*" pada pendaftaran peserta *Sharif Judge*, diperlukan beberapa perubahan dan penambahan kode. Berikut rancangan algoritma yang akan dilakukan

1. Menambahkan parameter "*Display Name*" pada fungsi *add_user* yang terdapat di *model User_model.php*.
2. Mengubah pemisah (*separator*) antar parameter pada fungsi *add_user*. Pemisah antar parameter yang awalnya menggunakan spasi akan diubah menggunakan tanda koma.
3. Menambahkan keterangan parameter "*Display Name*" pada halaman *add_user.twig* dan *add_user_result.twig*.
4. Menambahkan *Display Name* untuk *admin* pada proses *install Sharif Judge*.
5. Mengubah urutan data yang digunakan pada fungsi pendaftaran peserta melalui *email*.
6. Menambahkan *text field Display Name* pada halaman *Register (Open Public Registration)*.

Dari rancangan algoritma yang akan diterapkan, terdapat perubahan dan penambahan kode pada beberapa *file*. Beberapa *file* tersebut antara lain *controller Install.php*, *controller Login.php*, *model User_model.php*, *view add_user.twig*, *view add_user_result.twig* dan *view register.twig*. Berikut beberapa baris potongan kode program

Install.php

```
...
252     // add admin user
253     $this->user_model->add_user(
254         $this->input->post('username'),
255         $this->input->post('email'),
256         $this->input->post('password'),
257         'admin'
258     );
...
```

Login.php

```
...
92     $this->user_model->add_user(
93         $this->input->post('username'),
94         $this->input->post('email'),
95         $this->input->post('password'),
96     );
...
```

User_model.php

```
...
120     */
121     public function add_user($username, $email, $password, $role)
122     {
123     ...
138         'username' => $username,
139         'email' => $email,
140         'password' => $this->password_hash->HashPassword($password),
141     ...
176         $parts = preg_split('/\s+/', $line);
177         if (count($parts) != 4)
178             continue; //ignore lines that not contain 4 parts
179     ...
180         if (strtolower(substr($parts[2], 0, 6)) == 'random')
181         {
182             // generate random password
183             $len = trim(substr($parts[2], 6), '[]');
184             if (is_numeric($len)){
185                 $this->load->helper('string');
186                 $parts[2] = shj_random_password($len);
187             }
188         }
189     ...
190     $result = $this->add_user($parts[0], $parts[1], $parts[2], $parts[3]);
191 }
```

```

192     if ($result === TRUE)
193         array_push($users_ok, array($parts[0], $parts[1], $parts[2],
    $parts[3]));
194     else
195         array_push($users_error, array($parts[0], $parts[1], $parts[2],
    $parts[3], $result));
...
230     $text = str_replace('{SITE_URL}', base_url(), $text);
231     $text = str_replace('{ROLE}', $user[3], $text);
232     $text = str_replace('{USERNAME}', $user[0], $text);
233     $text = str_replace('{PASSWORD}', htmlspecialchars($user[2]), $text);
234     $text = str_replace('{LOGIN_URL}', base_url(), $text);
...

```

add_user.twig

```

...
66     #
67     # USERNAME EMAIL PASSWORD ROLE
68     #
...

```

add_user_result.twig

```

...
9         {% for item in ok %}
10         <li>Username: {{ item[0] }} Email: {{ item[1] }} Password: <code>{{ item
    [2] }}</code> Role: {{ item[3] }}</li>
11         {% endfor %}
...
17         {% for item in error %}
18         <li>Username: {{ item[0] }} Email: {{ item[1] }} Password: <code>{{ item
    [2] }}</code> Role: {{ item[3] }} ({{ item[4] }})</li>
19         {% endfor %}
...

```

register.twig

```

...
33         {{ form_error('email', '<div class="shj_error">', '</div>') }}
34         </p>
35         <p>
36         <label for="form_password">Password</label><br/>
...

```

Perubahan dan penambahan kode di *User_model.php* untuk menambahkan parameter "*Display Name*". Penambahan kode tersebut terjadi pada baris 121, 139, 180, 183, 190, 193 dan 195. Mengubah pemisah antar parameter menggunakan tanda koma terjadi pada baris 176. Mengubah urutan data pada pendaftaran peserta melalui *email* terjadi pada baris 231 dan 233. Berikut hasil penambahan dan perubahan kode program yang terjadi di *User_model.php*

User_model.php

```

...
120    */
121    public function add_user($username, $email, $display_name, $password,
    $role)
122    {
...
138    'username' => $username,
139    'email' => $email,
140    'display_name' => $display_name,
141    'password' => $this->password_hash->HashPassword($password),
...
177    $parts = preg_split('/,+/', $line);
178    if (count($parts) != 5)
179        continue; //ignore lines that not contain 5 parts
...
181    if (strtolower(substr($parts[3], 0, 6)) == 'random')
182    {
183        // generate random password
184        $len = trim(substr($parts[3], 6), '[]');
185        if (is_numeric($len)){
186            $this->load->helper('string');
187            $parts[3] = shj_random_password($len);
188        }
189    }
190
191    $result = $this->add_user($parts[0], $parts[1], $parts[2], $parts[3]);
192
193    if ($result === TRUE)
194        array_push($users_ok, array($parts[0], $parts[1], $parts[2],
    $parts[3], $parts[4]));
195    else
196        array_push($users_error, array($parts[0], $parts[1], $parts[2],
    $parts[3], $parts[4], $result));
...
231    $text = str_replace('{SITE_URL}', base_url(), $text);
232    $text = str_replace('{ROLE}', $user[4], $text);
233    $text = str_replace('{USERNAME}', $user[0], $text);
234    $text = str_replace('{PASSWORD}', htmlspecialchars($user[3]), $text);
235    $text = str_replace('{LOGIN_URL}', base_url(), $text);
...

```

Penambahan kode di *Install.php* untuk menambahkan *Display Name admin* terjadi setelah baris 255. Berikut hasil penambahan kode program yang terjadi di *Install.php*

Install.php

```

...
252    // add admin user
253    $this->user_model->add_user(
254        $this->input->post('username'),
255        $this->input->post('email'),
256        'Admin',

```



```

257         $this->input->post('password'),
258         'admin'
);
...

```

Perubahan dan penambahan kode di halaman *add_user.twig* untuk menambahkan keterangan parameter "*Display Name*". Penambahan kode tersebut terjadi pada baris 67, sedangkan halaman *add_user_result.twig* terjadi pada baris 10 dan 18. Berikut hasil penambahan kode program yang terjadi di halaman *add_user.twig* dan halaman *add_user_result.twig*

add_user.twig

```

...
66     #
67     # USERNAME,EMAIL,DISPLAY-NAME,PASSWORD,ROLE
68     #
...

```

add_user_result.twig

```

...
9      {% for item in ok %}
10     <li>Username: {{ item[0] }} Email: {{ item[1] }} Diplay Name: {{ item[2] }}
      Password: <code>{{ item[3] }}</code> Role: {{ item[4] }} </li>
11     {% endfor %}
...
17     {% for item in error %}
18     <li>Username: {{ item[0] }} Email: {{ item[1] }} Diplay Name: {{ item[2] }}
      Password: <code>{{ item[3] }}</code> Role: {{ item[4] }} ({{ item[5] }})</li>
19     {% endfor %}
...

```

Penambahan kode di halaman *register.twig* untuk menambahkan *text field Display Name* terjadi setelah baris 34 dan pada *controller Login.php* setelah baris 94. Berikut hasil penambahan kode program yang terjadi di halaman *register.twig* dan *controller Login.php*.

register.twig

```

...
33     {{ form_error('email','<div class="shj_error">','</div>')}}
34     </p>
35     <p>
36         <label for="form_displayname">Display Name</label><br/>
37         <input id="form_displayname" type="text" name="displayname"
      required="required" pattern="[A-Za-z\s]+" title="The Display Name field must
      be contain only alphabetical letters" class="sharif_input" value="{{ set_value
      ('displayname') }}" />
38         {{ form_error('form_displayname','<div class="shj_error">','</
      div>')}}
39     </p>
40     <p>
41     <label for="form_password">Password</label><br/>
...

```

Login.php

```

...
92     $this->user_model->add_user(
93         $this->input->post('username'),
94         $this->input->post('email'),
95         $this->input->post('displayname'),
96         $this->input->post('password'),
...

```

4.7 Menambahkan fitur "*Lock Student's Display Name*"

Fitur "*Lock Student's Display Name*" membutuhkan sebuah "*key*" pada *database*, dimana "*key*" tersebut berfungsi untuk menyimpan sebuah nilai. Nilai yang disimpan akan menentukan apakah para peserta dapat mengubah *Display Name* atau tidak. "*Key*" akan disimpan pada tabel *shj_settings* pada kolom *shj_key* dengan nama *lock_student_display_name*. *lock_student_display_name* akan memiliki nilai *default shj_value* = 0. Jika nilai dari *lock_student_display_name* = 1, maka para peserta tidak dapat mengubah *Display Name*, sebaliknya jika bernilai 0, maka para peserta dapat mengubah *Display Name*.

Rancangan algoritma yang digunakan untuk menambahkan fitur "*Lock Student's Display Name*" yaitu

1. Menambahkan *shj_key* dengan nama *lock_student_display_name* yang memiliki nilai *shj_value* = 0 pada tabel *shj_settings*.
2. Menambahkan *check box* pada halaman *settings.twig* untuk mengaktifkan atau menonaktifkan fitur "*Lock Student's Display Name*".
3. Jika fitur "*Lock Student's Display Name*" diaktifkan, maka *text field Display Name* pada halaman *profile.twig* akan dinonaktifkan (*disabled*).
4. Jika fitur "*Lock Student's Display Name*" dinonaktifkan, maka *text field Display Name* pada halaman *profile.twig* akan kembali aktif.
5. Menambahkan fungsi untuk mengecek kembali nilai dari *lock_student_display_name* pada saat peserta menyimpan perubahan yang terjadi di halaman *profile.twig*. Hal tersebut dilakukan untuk menangani para peserta yang "memaksa" agar dapat mengubah *Display Name* dengan cara *inspect element* lalu menghapus kode "*disabled*" pada *text field Display Name*.

Dari rancangan algoritma yang akan diterapkan, terdapat penambahan kode pada beberapa *file*. Beberapa *file* tersebut antara lain *controller Profile.php*, *controller Settings.php*, *model User_model.php*, *view settings.twig* dan *view profile.twig*. Berikut beberapa baris potongan kode program

Profile.php

```

...
65         'role' => $user->role,
66         'form_status' => $this->form_status,
67     );
...

```

Settings.php

```
...
113     'results_per_page_all' => $this->input->post('rpp_all'),
114     'results_per_page_final' => $this->input->post('rpp_final'),
115     'week_start' => $this->input->post('week_start'),
...
```

User_model.php

```
...
405     $username = $the_user->username;
406
407     $user=array(
408         'display_name' => $this->input->post('display_name'),
409         'email' => $this->input->post('email')
410     );
...
```

settings.twig

```
...
115         <span class="form_comment">Enable Log</span>
116     </p>
117     <p class="input_p">
...
```

profile.twig

```
...
31         <input id="form_name" type="text" name="display_name" class="
    sharif_input medium" value="{{ display_name }}" />
32         {{ form_error('display_name', '<div class="shj_error">', '</div
    >') }}
33     </p>
...
```

Penambahan kode di halaman *settings.twig* untuk menambahkan check box fitur "Lock Student's Display Name". Penambahan kode tersebut terjadi setelah baris 116. Berikut hasil penambahan kode program yang terjadi di halaman *settings.twig*

settings.twig

```
...
115         <span class="form_comment">Enable Log</span>
116     </p>
117     <p class="input_p">
118         <input id="form_lock_student_display_name" type="checkbox" name="
    lock_student_display_name" value="1" {{ lock_student_display_name ? 'checked'
    }} />
119         <label for="form_lock_student_display_name">Lock Student's
    Display Name</label><br>
120         <span class="form_comment">Student's can't change their display
    name</span>
```

```

121     </p>
122     <p class="input_p">
...

```

Penambahan kode di *controller Settings.php* untuk menyimpan nilai dari *check box* fitur "*Lock Student's Display Name*" ke *database*. Penambahan kode tersebut terjadi setelah baris 115. Berikut hasil penambahan kode program yang terjadi di *controller Settings.php*

Settings.php

```

...
113     'results_per_page_all' => $this->input->post('rpp_all'),
114     'results_per_page_final' => $this->input->post('rpp_final'),
115     'week_start' => $this->input->post('week_start'),
116     'lock_student_display_name' => $this->input->post('
    lock_student_display_name')===NULL?0:1,
...

```

Penambahan kode di *controller Profile.php* untuk mengecek apakah fitur "*Lock Student's Display Name*" aktif atau tidak aktif. Penambahan kode tersebut terjadi setelah baris 66. Hasil pengecekan tersebut akan dikirimkan untuk halaman *profile.twig*. Berikut hasil penambahan kode program yang terjadi di *controller Profile.php*

Profile.php

```

...
65         'role' => $user->role,
66         'form_status' => $this->form_status,
67         'lock_student_display_name' => $this->settings_model->get_setting
    (lock_student_display_name),
68     );
...

```

Penambahan kode di halaman *profile.twig* untuk menentukan apakah *text field Display Name* akan diaktifkan atau tidak. Penambahan kode tersebut terjadi setelah baris 31. Berikut hasil penambahan kode program yang terjadi di tampilan *profile.twig*

profile.twig

```

...
31
32     {% if lock_student_display_name == 1 %}
33     <input id="form_name" type="text" name="display_name" class="sharif_input
    medium" value="{{ display_name }}" disabled/>
34     {% else %}
35     <input id="form_name" type="text" name="display_name" class="sharif_input
    medium" value="{{ display_name }}" />
36     {% endif %}
37
...

```

Penambahan kode di *model User_model.php* untuk mengecek apakah fitur "*Lock Student's Display Name*" aktif atau tidak aktif. Penambahan kode tersebut terjadi setelah baris 405. Berikut hasil penambahan kode program yang terjadi di *model User_model.php*

User_model.php

```
...
405     $username = $the_user->username;
406
407     $display_name = $this->input->post('display_name');
408     $locked = $this->settings_model->get_setting(lock_student_display_name);
409     if ($locked == 1) {
410         $display_name = $the_user->display_name;
411     }
412
413     $user=array(
414         'display_name' => $display_name,
415         'email' => $this->input->post('email')
416     );
...
```

4.8 Menambahkan fitur "*Archived Assignment*"

Fitur "*Archived Assignment*" membutuhkan sebuah atribut baru pada *database*, dimana atribut tersebut berfungsi untuk menyimpan sebuah nilai. Nilai yang disimpan akan menentukan apakah *assignment* tersebut bersifat *Archived Assignment* atau tidak. Atribut baru tersebut akan ditambahkan pada tabel *shj_assignments* dengan nama *archived_assignment* yang menggunakan tipe data *tinyint*. *archived_assignment* akan memiliki nilai *default* = 0. Jika nilai dari *archived_assignment* = 1, maka *assignment* tersebut merupakan sebuah *archived_assignment*, sebaliknya jika bernilai 0, maka *assignment* tersebut merupakan *assignment* biasa.

Rancangan algoritma yang digunakan untuk menambahkan fitur "*Archived Assignment*" yaitu

1. Menambahkan atribut baru dengan nama *archived_assignment* yang menggunakan tipe data *tinyint*. Atribut baru tersebut akan ditambahkan pada tabel *shj_assignments*
2. Menambahkan *check box* pada halaman *assignments.twig* untuk mengaktifkan atau menonaktifkan fitur "*Archived Assignment*".
3. Jika fitur "*Archived Assignment*" diaktifkan, maka secara otomatis *text field Start time* akan bernilai 1970-01-02 00:00:00, *text field Finish Time* akan bernilai 2038-01-18 00:00:00 dan *text field Extra Time* akan bernilai 0.
4. *Assignment* yang bersifat *Archived Assignment* tidak akan muncul pada kalender halaman *dashboard*.

Dari rancangan algoritma yang akan diterapkan, terdapat penambahan kode pada beberapa *file*. Beberapa file tersebut antara lain *model Assignment_model.php*, *view add_assignment.twig* dan *view dashboard.twig*. Berikut beberapa baris potongan kode program *Assignment_model.php*

```
...
37     foreach($extra_items as $extra_item)
38     {
39         $extra_time *= $extra_item;
40     }
41     $assignment = array(
```

```

42         'id' => $id,
43         'name' => $this->input->post('assignment_name'),
44         'problems' => $this->input->post('number_of_problems'),
45         'total_submits' => 0,
46         'open' => ($this->input->post('open')===NULL?0:1),
47         'scoreboard' => ($this->input->post('scoreboard')===NULL?0:1),
48         'javaexceptions' => ($this->input->post('javaexceptions')===NULL
?0:1),
49         'description' => '', /* todo */
50         'start_time' => date('Y-m-d H:i:s', strtotime($this->input->post('
start_time'))),
51         'finish_time' => date('Y-m-d H:i:s', strtotime($this->input->post
('finish_time'))),
52         'extra_time' => $extra_time*60,
53         'late_rule' => $this->input->post('late_rule'),
54         'participants' => $this->input->post('participants')
55     );
...

```

add_assignment.twig

```

...
37     $("#add").click(function(){
38         $('#problems_table>tbody').append(shj.row.replace(/PID/g,
(shj.num_of_problems+1)));
39         shj.num_of_problems++;
40         $('#nop').attr('value', shj.num_of_problems);
41     });
42     $(document).on('click', '.delete_problem', function(){
43         if (shj.num_of_problems==1) return;
44         var row = $(this).parents('tr');
...
154         {{ form_error('javaexceptions', '<div class="shj_error">', '</div
>')}}
155     </p>
156     <p class="input_p">
...

```

dashboard.twig

```

...
28     {% set colors = ['#812C8C', '#FF750D', '#2C578C', '#013440', '#A6222C
', '#42758C', '#02A300', '#BA6900'] %}
29     {% for assignment in all_assignments %}
30         {id:{{ assignment.id }},title:'{{ assignment.name|e('js') }}',
start:'{{ assignment.start_time }}', end:' {{ assignment.finish_time }}',
31         allDay:false,color:'{{ colors[(loop.index0)%colors|length] }}'}
32     {% if not loop.last %},{% endif %}
33     {% endfor %}
...

```

Penambahan kode di *model Assignment_model.php* untuk menyimpan nilai dari *check box* fitur "Archived Assignment" ke *database*. Penambahan kode tersebut terjadi setelah baris 54. Penambahan kode setelah baris 40 berfungsi untuk mengambil nilai dari *archived_assignment* dari sebuah *assignment*. Berikut hasil penambahan kode program yang terjadi di *textitmodel Assignment_model.php*

Assignment_model.php

```
...
37     foreach($extra_items as $extra_item)
38     {
39         $extra_time += $extra_item;
40     }
41
42     $archived_assignment = $archived_assignment = $this->input->post('
archived_assignment')!==NULL ? 1 : 0;
43
44     $assignment = array(
45         'id' => $id,
46         'name' => $this->input->post('assignment_name'),
47         'problems' => $this->input->post('number_of_problems'),
48         'total_submits' => 0,
49         'open' => ($this->input->post('open')===NULL?0:1),
50         'scoreboard' => ($this->input->post('scoreboard')===NULL?0:1),
51         'javaexceptions' => ($this->input->post('javaexceptions')===NULL
?0:1),
52         'description' => '', /* todo */
53         'start_time' => date('Y-m-d H:i:s', strtotime($this->input->post('
start_time'))),
54         'finish_time' => date('Y-m-d H:i:s', strtotime($this->input->post
('finish_time'))),
55         'extra_time' => $extra_time*60,
56         'late_rule' => $this->input->post('late_rule'),
57         'participants' => $this->input->post('participants')
58         'archived_assignment' => $archived_assignment
59     );
...
```

Penambahan kode di halaman *add_assignment.twig* untuk menambahkan *check box* fitur "Archived Assignment". Penambahan kode tersebut terjadi setelah baris 155. Penambahan kode setelah baris 41 untuk mengisi nilai *text field Start time* menjadi 1970-01-02 00:00:00, *text field Finish Time* menjadi 2038-01-18 00:00:00 dan *text field Extra Time* menjadi 0. Berikut hasil penambahan kode program yang terjadi di halaman *add_assignment.twig*

add_assignment.twig

```
...
37     $("#add").click(function(){
38         $('#problems_table>tbody').append(shj.row.replace(/PID/g,
(shj.num_of_problems+1)));
39         shj.num_of_problems++;
40         $('#nop').attr('value', shj.num_of_problems);
41     });
42     $("#form_a_archived_assignment").click(function(){
```

```

43         if ($("#form_a_archived_assignment").is(':checked')) {
44             $("#start_time").val('1970-01-02 00:00:00');
45             $("#finish_time").val('2038-01-18 00:00:00');
46             $("#form_extra_time").val('0');
47         }
48         else{
49             $("#start_time").val('');
50             $("#finish_time").val('');
51             $("#form_extra_time").val('');
52         }
53     });
54     $(document).on('click', '.delete_problem', function(){
55         if (shj.num_of_problems==1) return;
56         var row = $(this).parents('tr');
57         ...
166         {{ form_error('javaexceptions', '<div class="shj_error">', '</div
>') }}
167     </p>
168     <p class="input_p">
169         <input id="form_a_archived_assignment" type="checkbox" name="
archived_assignment" value="1" {{ edit ? (edit_assignment.archived_assignment
? 'checked') : set_checkbox('archived_assignment', '1')|raw }} />
170         <label for="form_a_archived_assignment" class="default">Archived
Assignment</label>
171         <span class="form_comment space-left">Check this to make an
archived assignment</span>
172         {{ form_error('archived_assignment', '<div class="shj_error">',
'</div>') }}
173     </p>
174     <p class="input_p">
175     ...

```

Penambahan kode di halaman *dashboard.twig* untuk mengatur *assignment* yang bersifat *Archived Assignment* agar tidak muncul pada kalender. Penambahan kode tersebut terjadi setelah baris 29. Berikut hasil penambahan kode program yang terjadi di halaman *dashboard.twig*

```

...
28     {% set colors = ['#812C8C', '#FF750D', '#2C578C', '#013440', '#A6222C
', '#42758C', '#02A300', '#BA6900'] %}
29     {% for assignment in all_assignments %}
30         {% if assignment.archived_assignment == '0' %}
31             {id:{{ assignment.id }},title:'{{ assignment.name|e('js')
}}', start:'{{ assignment.start_time }}', end:' {{ assignment.finish_time
}}', allDay:false,color:'{{ colors[(loop.index0)%colors|length] }}'}
32         {% endif %}
33         {% if assignment.archived_assignment == '1' %}
34         {}
35         {% endif %}
36         {% if not loop.last %},{% endif %}
37     {% endfor %}

```


...

4.9 Menambahkan halaman *Hall of Fame*

Halaman *Hall of Fame* tidak membutuhkan atribut atau tabel baru pada *database* namun perlu ditambahkan *model*, *view* dan *controller*.

1. Model

Model untuk halaman *Hall of Fame* akan bernama *Hof_model.php*. Berikut adalah perincian fungsi yang terdapat dalam rancangan *model Hof_model.php*.

Tabel 4.6: Perincian fungsi *get_all_final_submission*

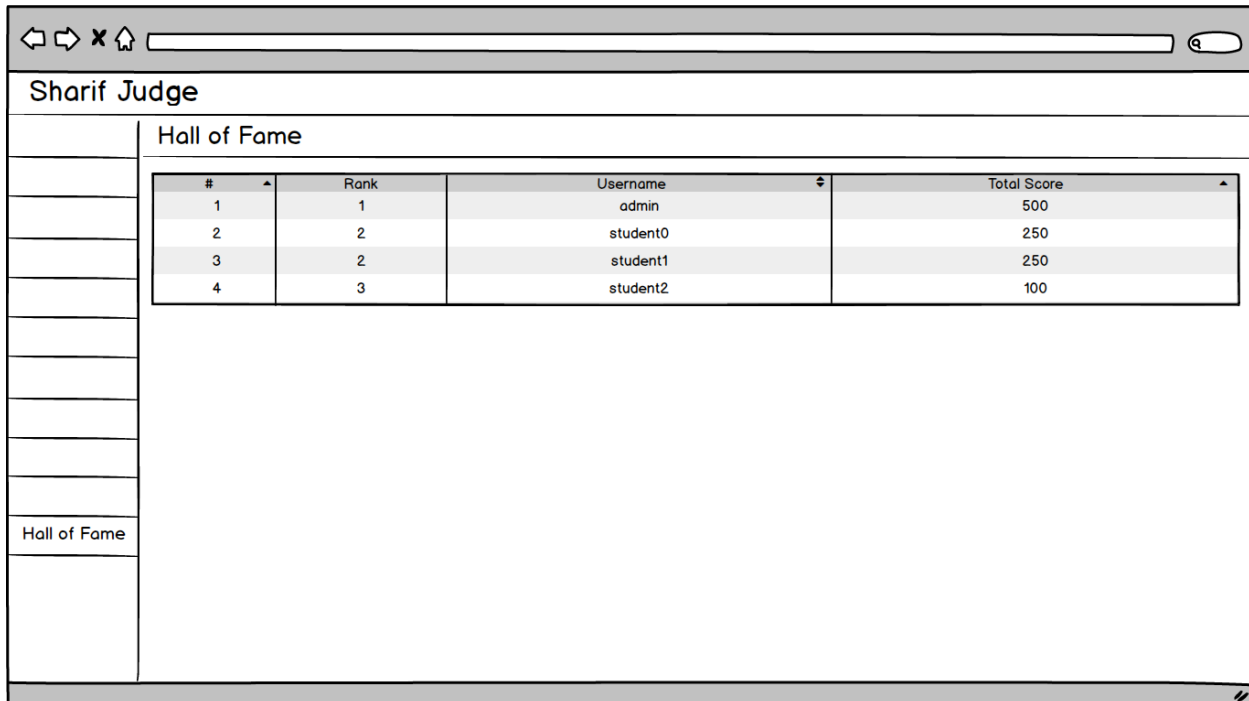
Nama Method	<i>get_all_final_submission</i>
Parameter Input	-
Parameter Output	semua entri nilai submissions yang telah dijumlahkan
Tabel yang berhubungan	<i>shj_submissions</i>
Deskripsi	Proses untuk mengembalikan semua entri nilai submission yang telah dijumlahkan pada tabel <i>shj_submissions</i>
Algoritma	<ul style="list-style-type: none"> • menjumlahkan seluruh nilai submission setiap peserta yang tidak bersifat "<i>Upload Only</i>". Nilai yang telah dijumlahkan akan disimpan sebagai total skor. • mengurutkan total skor dari yang paling besar • mengembalikan seluruh entri yang telah dijumlahkan dalam bentuk <i>array</i>

Tabel 4.7: Perincian fungsi *get_all_user_assignments*

Nama Method	<i>get_all_user_assignments</i>
Parameter Input	<i>\$username</i>
Parameter Output	mengembalikan seluruh <i>details</i> dari <i>assignment</i> pengguna tertentu
Tabel yang berhubungan	<i>shj_submissions</i>
Deskripsi	Proses untuk mengembalikan <i>details assignment</i> pengguna tertentu. <i>Details</i> akan berisikan nama <i>assignment</i> , nama <i>problem</i> dan skor
Algoritma	<ul style="list-style-type: none"> • menyimpan nama <i>assignment</i>, nama <i>problem</i> dan skor setiap <i>problem</i> dari sebuah <i>assignment</i> pengguna tertentu • mengembalikan <i>details</i> di atas dalam bentuk <i>array</i>

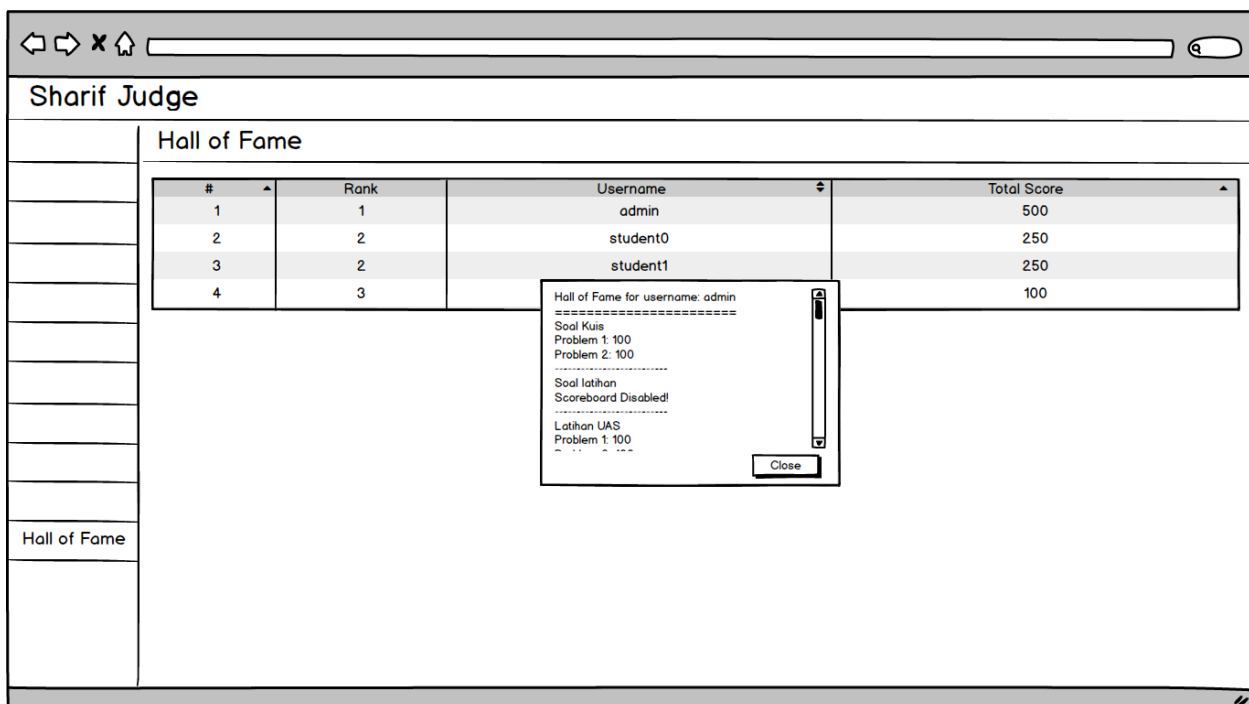
2. View

View untuk halaman *Hall of Fame* akan bernama *halloffame.twig*. Menu halaman *Hall of Fame* akan terletak di bawah menu *Scoreboard*. Pada halaman ini juga akan berlaku sistem ranking dimana para peserta akan diurutkan berdasarkan total skor. Jika total skor yang dimiliki peserta memiliki nilai yang sama dengan peserta lainnya, maka peserta tersebut memiliki ranking yang sama dengan peserta lainnya. Berikut adalah rancangan tampilan halaman *Hall of Fame*



Gambar 4.2: Rancangan tampilan halaman *Hall of Fame*

Berikut adalah rancangan tampilan detail dari *Hall of Fame* peserta tertentu



Gambar 4.3: Rancangan tampilan *details Hall of Fame* peserta tertentu

3. Controller

Controller untuk halaman *Hall of Fame* akan bernama *Halloffame.php*. Berikut adalah perincian fungsi yang terdapat dalam rancangan *controller Logs.php*.

Tabel 4.8: Perincian fungsi *consturct__*

Nama <i>Method</i>	<i>consturct__</i>
Parameter <i>Input</i>	-
Parameter <i>Output</i>	-
Tabel yang berhubungan	-
Deskripsi	membatasi pengguna yang dapat mengakses halaman <i>Hall of Fame</i>
Algoritma	<ul style="list-style-type: none"> • mengecek <i>session</i> pengguna yang akan mengakses halaman <i>Hall of Fame</i> • jika <i>session</i> tidak berstatus '<i>logged_in</i>', maka pengguna akan dialihkan ke halaman <i>login</i> • memuat <i>model Hof_model.php</i>

Tabel 4.9: Perincian fungsi *index*

Nama <i>Method</i>	<i>index</i>
Parameter <i>Input</i>	-
Parameter <i>Output</i>	-
Tabel yang berhubungan	<i>shj_submissions</i>
Deskripsi	Proses untuk memuat seluruh entri <i>submissions</i> yang telah dijumlahkan pada halaman <i>halloffame.twig</i>
Algoritma	<ul style="list-style-type: none"> • memuat data <i>Hall of Fame</i> menggunakan fungsi <i>get_all_final_submission</i> dari <i>model Hof_model.php</i> • memproses data untuk tampilan <i>halloffame.twig</i>

Tabel 4.10: Perincian fungsi *hof_details*

Nama <i>Method</i>	<i>hof_details</i>
Parameter <i>Input</i>	-
Parameter <i>Output</i>	-
Tabel yang berhubungan	<i>shj_submissions</i>
Deskripsi	Proses untuk memuat details <i>submissions</i> pada halaman <i>halloffame.twig</i>
Algoritma	<ul style="list-style-type: none"> • memuat details <i>Hall of Fame</i> peserta tertentu menggunakan fungsi <i>get_all_user_assignments</i> dari <i>model Hof_model.php</i> • memproses data untuk tampilan details <i>Hall of Fame</i> dari peserta tertentu pada halaman <i>halloffame.twig</i>

Selain menambahkan *file model, view* dan *controller*, terdapat penambahan fungsi pada *file shj_functions.js* yang terletak di *Sharif-Judge/assets/js*. Penambahan fungsi tersebut berguna untuk meminta details dari Hall of Fame peserta tertentu menggunakan fungsi *hof_details* pada *controller Halloffame.php* lalu menampilkannya. Berikut beberapa baris potongan kode program *shj_functions.js*

```

...
478     });
479
480
481
482     /**
483     * Set dir="auto" for all input elements
484     */
485     $(document).ready(function(){
486         $('input').attr('dir', 'auto');
487     });
...

```

Fungsi tersebut akan ditambahkan setelah baris 479. Berikut hasil penambahan kode program yang terjadi di *shj_functions.js*

4.10 Integrasi *login* ke *server RADIUS*

Sharif Judge memerlukan *library* baru agar dapat mengintegrasikan *login* ke *server RADIUS*. *Library* yang akan digunakan adalah *Dapphp/Radius*. *Dapphp/Radius* adalah klien *RADIUS PHP* untuk mengautentikasi pengguna terhadap *server RADIUS* REFRENSI. Cara yang disarankan untuk menginstal *Dapphp/Radius* adalah menggunakan *Composer*. Jika *Composer* telah terinstall, maka jalankan perintah "*composer require dapphpradius*" atau tambahkan "*dapphp/radius*" pada file *composer.json* bagian "*section*" [7].

Rancangan algoritma yang digunakan untuk menambahkan mengintegrasikan *login* ke *server RADIUS* yaitu

1. Menginstall *Composer* pada perangkat lunak *Sharif Judge*.
2. Menambahkan *library DapphpRadius* menggunakan *Composer*.
3. Membuat *file secrets.php* pada direktori *Sharif-Judge\application\config* untuk menyimpan konfigurasi alamat *server RADIUS*.
4. Mengintegrasikan *login* pengguna *Sharif Judge* ke *server RADIUS*.

Dari rancangan algoritma yang akan diterapkan, terdapat penambahan kode pada *file model User_model.php*. Berikut beberapa baris potongan kode program

User_model.php

```

...
7     defined('BASEPATH') OR exit('No direct script access allowed');
8
9     class User_model extends CI_Model
...
323     /**
324     * Validate User
325     *
326     * Returns TRUE if given username and password is valid for login
327     *
328     * @param $username
329     * @param $password
330     * @return bool

```

```

331     */
332     public function validate_user($username, $password)
333     {
334         $this->load->library('password_hash', array(8, FALSE));
335         $query = $this->db->get_where('users', array('username' =>
$username));
336         if ($query->num_rows() != 1)
337             return FALSE;
338         if ($query->row()->username != $username) // needed (because of
utf8_general_ci [ci=case insensitive])
339             return FALSE;
340         if ($this->password_hash->CheckPassword($password, $query->row()
->password))
341             return TRUE;
342         return FALSE;
343     }
...

```

Penambahan kode di model *User_model* untuk memanggil *library Dapphp/Radius* dan mengintegrasikan *login* pengguna *Sharif Judge* ke *server RADIUS*. Penambahan kode tersebut terjadi setelah baris 7 dan 341. Berikut hasil penambahan kode program yang terjadi di model *User_model* *User_model.php*

```

...
7     defined('BASEPATH') OR exit('No direct script access allowed');
8     +use Dapphp\Radius\Radius;
9
10    class User_model extends CI_Model
...
324    /**
325     * Validate User
326     *
327     * Returns TRUE if given username and password is valid for login
328     *
329     * @param $username
330     * @param $password
331     * @return bool
332     */
333    public function validate_user($username, $password)
334    {
335        $this->load->library('password_hash', array(8, FALSE));
336        $query = $this->db->get_where('users', array('username' =>
$username));
337        if ($query->num_rows() != 1)
338            return FALSE;
339        if ($query->row()->username != $username) // needed (because of
utf8_general_ci [ci=case insensitive])
340            return FALSE;
341        if ($this->password_hash->CheckPassword($password, $query->row()
->password))
342            return TRUE;
343    }

```

```
344         $this->load->config('secrets');
345         if($this->config->item('shj_authenticate') == 'radius') {
346             $client = new Radius();
347             $client->setServer($this->config->item('shj_radius')['
server']) // RADIUS server address
348             ->setSecret($this->config->item('shj_radius')['
secret']);
349             if($client->accessRequest($username, $password))
350                 return TRUE;
351         }
352
353         return FALSE;
354     }
...

```

DAFTAR REFERENSI

- [1] Naderi, M. J. (2014) Sharif judge. <https://github.com/mjnaderi/Sharif-Judge/>. 6 Oktober 2017.
- [2] of Technology, B. C. I. (2017) Codeigniter documentation. https://codeigniter.com/user_guide/overview/index.html. 6 Oktober 2017.
- [3] Naderi, M. J. (2014) Sharif judge documentation. <https://github.com/mjnaderi/Sharif-Judge/tree/docs/v1.4>. 6 Oktober 2017.
- [4] Aiken, A. (2010) A system for detecting software similarity. <http://theory.stanford.edu/~aiken/moss/>. 25 Februari 2018.
- [5] Smith, B. (2009) The gnu operating system and the free software movement. <https://www.gnu.org/>. 22 Februari 2018.
- [6] Potencier, F. (2009) The flexible, fast, and secure template engine for php. <https://twig.symfony.com/>. 12 Maret 2018.
- [7] Phillips, D. (2016) Dapphp - radius - a pure php radius client based on the sysco/al implementation. <https://github.com/dapphp/radius>. 10 April 2018.

LAMPIRAN A

KODE PROGRAM

Listing A.1: MyCode.c

```

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

```

Listing A.2: MyCode.java

```

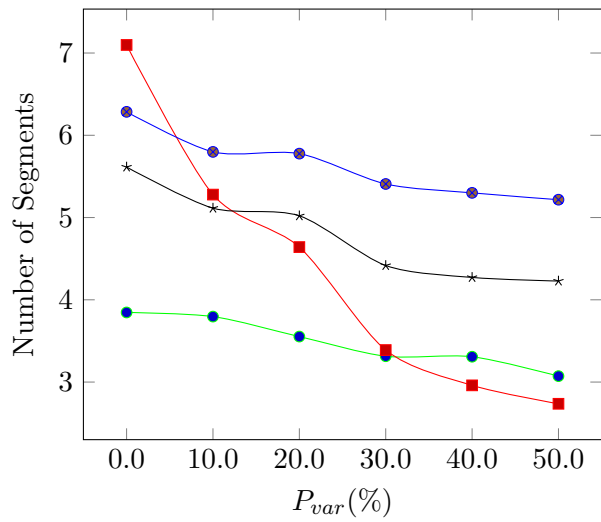
1 import java.util.ArrayList;
2 import java.util.Collections;
3 import java.util.HashSet;
4
5 //class for set of vertices close to furthest edge
6 public class MyFurSet {
7     protected int id; //id of the set
8     protected MyEdge FurthestEdge; //the furthest edge
9     protected HashSet<MyVertex>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

```

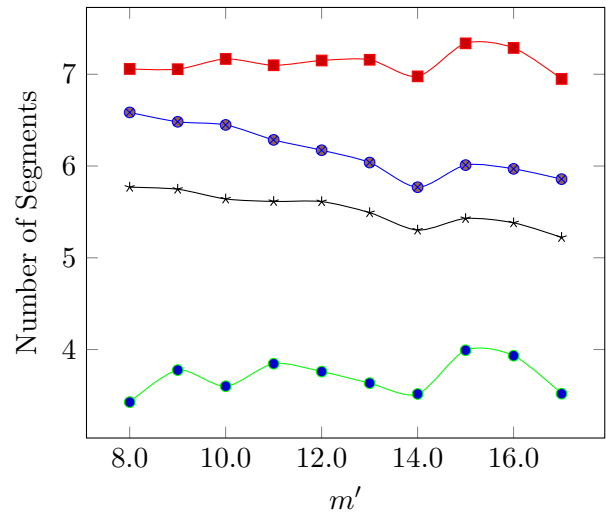

LAMPIRAN B

HASIL EKSPERIMEN

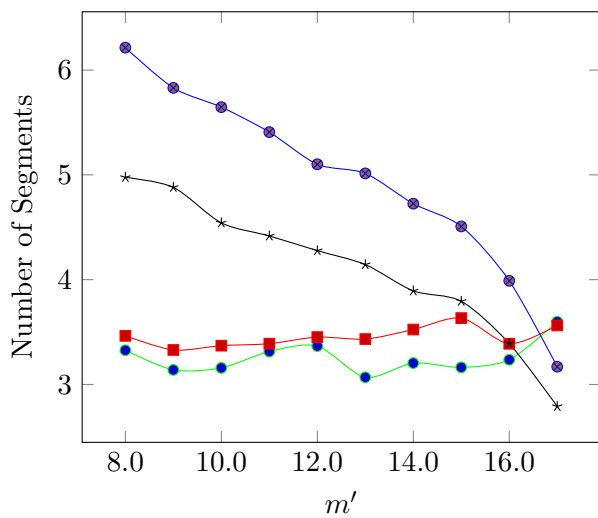
Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



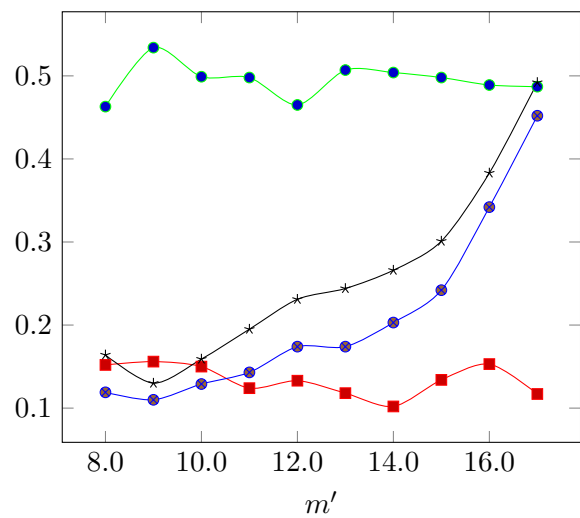
Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4