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, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer 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, consectetuer 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 xxi

Daftar Tabel xxiii

1 Pendahuluan 1
- 1.1 Latar Belakang . . . 1
- 1.2 Rumusan Masalah . . . 1
- 1.3 Tujuan . . . 2
- 1.4 Batasan Masalah . . . 2
- 1.5 Metodologi . . . 2
- 1.6 Sistematika Pembahasan . . . 2

2 Landasan Teori 3
- 2.1 *CodeIgniter* . . . 3
  - 2.1.1 Fitur-fitur CodeIgniter . . . 4
  - 2.1.2 *Flow Chart* Aplikasi . . . 5
  - 2.1.3 *Model-View-Controller* . . . 5
  - 2.1.4 Desain dan Tujuan Arsitektur . . . 9
- 2.2 *Sharif Judge* . . . 10
  - 2.2.1 Instalasi . . . 10
  - 2.2.2 *Clean URLs* . . . 11
  - 2.2.3 *Users* . . . 12
  - 2.2.4 Menambah *Assignment* . . . 13
  - 2.2.5 Struktur Pengujian . . . 20
  - 2.2.6 Deteksi Kecurangan . . . 23

3 Analisis 25
- 3.1 Analisis Kebutuhan Perangkat Lunak *Sharif Judge* . . . 25
  - 3.1.1 *Security with PHP* . . . 27
  - 3.1.2 *Securing Assignment* . . . 27
  - 3.1.3 *New Function* . . . 28
  - 3.1.4 *Solve Problem Indicator* . . . 28
  - 3.1.5 *Some Problem & Sugestion* . . . 28
  - 3.1.6 *Queue failed to process if submission take too long to complete?* . . . 28
  - 3.1.7 *Compilation Error on All Language* . . . 29
  - 3.1.8 Membatasi Soal (Deskripsi & PDF) Hanya Bisa Diakses saat *Assignment* "*Open*" 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 *Assignment* "*Open*" 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 *Assignment* "*Open*" dan Setelah Waktu Mulai . . . 30
3.1.14 Mengumpulkan *File* dengan Format TXT . . . 31
3.1.15 Perlu Ditambah Petunjuk Penamaan *File Input* dan *Output* . . . 31
3.1.16 Membatasi Soal (Deskripsi & PDF) Hanya Bisa Diakses saat *Assignment* "*Open*" dan Setelah Waktu Mulai . . . 31
3.1.17 Pendaftaran Peserta Disertai dengan *Display Name* . . . 31
3.1.18 Nama Pengguna *Sharif Judge* Seharusnya Tidak Bisa Diubah . . . 31
3.1.19 *Sharif Judge* Diharapkan Memiliki Fungsi Dimana *Assignment* Dapat Dikumpulkan Tanpa Adanya Batasan Waktu . . . 32
3.1.20 *Sharif Judge* Diharapkan Memiliki *Scoreboard Global* untuk Semua *Assignment* 32
3.1.21 Membatasi Soal (Deskripsi & PDF) Hanya Bisa Diakses saat *Assignment* "*Open*" dan Setelah Waktu Mulai . . . 33
3.1.22 Mengumpulkan *File* dengan Format TXT . . . 33
3.1.23 UI Masih Merepotkan . . . 33
3.1.24 UI Ada yang Tidak Berguna . . . 34
3.1.25 *Sharif Judge* Diharapkan Memiliki Fungsi *Reminder* . . . 34
3.1.26 Membatasi Soal (Deskripsi & PDF) Hanya Bisa Diakses saat *Assignment* "*Open*" dan Setelah Waktu Mulai . . . 34
3.1.27 Integrasi *Login* ke *Server RADIUS* . . . 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 *Submission* . . . 36
3.2 Analisis Sistem Kini . . . 38
3.2.1 *Model* . . . 38
3.2.2 *View* . . . 41
3.2.3 *Controller* . . . 51

4 Perancangan **55**
4.1 Mengganti Method *shell_exec("rm ...")* Menjadi *unlink()* . . . 55
4.2 Menambahkan Method Rekoneksi ke *Database* . . . 55
4.3 Membatasi Pengaksesan Soal (deskripsi & PDF) . . . 56
4.4 Mensupport *File* dengan Ekstensi TXT . . . 58
4.5 Menambahkan Halaman *Logs* . . . 60
4.6 Menambahkan Parameter "*Display Name*" pada Pendaftaran Peserta *Sharif Judge* 63
4.7 Menambahkan Fitur "*Lock Student's Display Name*" . . . 68
4.8 Menambahakan Fitur "*Archived Assignment*" . . . 71
4.9 Menambahkan Halaman *Hall of Fame* . . . 74
4.10 Integrasi *Login* ke *Server RADIUS* . . . 78
4.11 Branding Teknik Informatika . . . 80
4.12 Mengganti *Library PHPExcel* Menjadi *PhpSpreadsheet* . . . 81

5 Implementasi dan Pengujian **87**
5.1 Lingkungan untuk Implementasi dan Pengujian . . . 87
5.2 Implementasi . . . 87
5.3 Pengujian Fungsional . . . 91
5.3.1 Mengunduh Soal (deskripsi & PDF) yang Telah Dibatasi . . . 92
5.3.2 Membuat *Assignment* yang Menerima *File* dengan Ekstensi TXT . . . 93

5.3.3 Mengakses Halaman *24-hour Logs* 94
5.3.4 Mendaftarkan Peserta Menggunakan Tambahan Parameter "*Display Name*" 96
5.3.5 *Disable Display Name* Peserta Menggunakan Fitur "*Lock Student's Display Name*" 97
5.3.6 Menambahkan *Assignment* yang Mengaktifkan Fitur "*Archived Assignment*" 98
5.3.7 Mengakses Halaman *Hall of Fame* 99
5.3.8 *Login* Menggunakan *Username* yang Terdaftar pada *Server RADIUS* 100
5.3.9 Mengunduh *File Excel* dari Halaman *All Submission* dan *Users* 101
5.4 Pengujian Eksperimental 102
5.4.1 *Remove the Assignments Folder* 102
5.4.2 *Add User Email are not formatted correctly* 104
5.4.3 *Cannot Create Assignments* 105
5.4.4 *Link* ke *New Home of Sharif-Judge* 105
5.4.5 Hapus Perbedaan antara *Admin* dan *Student* pada PDF *Download* 106

Daftar Referensi **109**

A Kode Program Halaman *24-hour Logs* **111**

B Kode Program Halaman *Hall of Fame* **113**

C Kode Program *shj_functions.js* **117**

DAFTAR GAMBAR

2.1 *Flow Chart* Aplikasi 5
2.2 Tampilan Halaman *Assignments* 13

3.1 *Mockup* Halaman *Logs* 30
3.2 *Mockup* Halaman *Hall of Fame* 33
3.3 Detail Skor 33
3.4 Logo dan Ikon 35
3.5 *Banner Sharif Judge* 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 *Dashboard* 41
3.10 *Notifications* 42
3.11 *Assignments* 43
3.12 *Problems* 43
3.13 *Submit* 44
3.14 *Profile* 44
3.15 *Scoreboard* 45
3.16 *All Submission* 46
3.17 *Final Submission* 46
3.18 *Settings* 47
3.19 *User* 47
3.20 *Add User* 48
3.21 *Add Notification* 48
3.22 *Add Assignment* 49
3.23 *Rejudge* 49
3.24 *Login* 50
3.25 *Register* 50
3.26 *Lost* 51

4.1 Rancangan tampilan halaman *logs* 62
4.2 Rancangan tampilan halaman *Hall of Fame* 76
4.3 Rancangan tampilan *details Hall of Fame* peserta tertentu 76
4.4 Logo dan Ikon 80
4.5 *Banner Sharif Judge* 80
4.6 Halaman *Login Sharif Judge* 81
4.7 Ikon *Sharif Judge* pada *Title Bar* 81
4.8 Logo *Sharif Judge* pada *Top Bar* 81

5.1 Tampilan Halaman Login 89
5.2 Tampilan Halaman Hall of Fame 90
5.3 Tampilan Detail dari Hall of Fame 90
5.4 Tampilan Halaman Logs 91

5.5 Tampilan Side Menu 91
5.6 Empat Buah *Assignment* yang Dibuat 92
5.7 Pesan *Error "You are not registered for submitting."* 92
5.8 Pesan *Error "Selected assignment has not started."* 92
5.9 Pesan *Error "Selected assignment has finished."* 92
5.10 Pesan *Error "Selected assignment has been closed."* 93
5.11 Pembuatan *Assignment* Upload TXT 93
5.12 *Submit File* TXT 93
5.13 Halaman *All Submission* setelah Mengumpulkan *File* TXT 94
5.14 *File* TXT Hasil Unduh 94
5.15 File TXT Utama 94
5.16 Halaman *24-hour Logs* yang Tampil 95
5.17 Halaman *24-hour Logs* Mencatat Aktivitas *Login* Pengguna 95
5.18 Halaman *Add User* 96
5.19 Pengguna Berhasil Didaftarkan 96
5.20 *Display Name* yang Tampil Sesuai dengan Parameter yang Dimasukan 97
5.21 Mengaktifkan Fitur *Lock Student's Display Name* 97
5.22 *Text Field Display Name* Menjadi *Disable* 98
5.23 Mengaktifkan Fitur *Archived Assignment* 98
5.24 *Finish Time* dengan Nilai "2038-01-18 00:00:00" 99
5.25 *Archived Assignment* Tidak Muncul pada Kalendar 99
5.26 Halaman *Hall of Fame* 100
5.27 *Details* Nilai yang Diperoleh 100
5.28 Berhasil *Login* ke *SharIF Judge* 101
5.29 Letak Tombol Unduh *Excel* pada Halaman *Users* 101
5.30 Isi *Excel judge_all_submissions.xlsx* 101
5.31 Isi *Excel sharifjudge_users.xlsx* 102
5.32 Perubahan yang Terjadi pada Halaman *Settings* 103
5.33 Format *Email* Tidak Beraturan 104

DAFTAR TABEL

2.1 *User Roles Table* 12
2.2 *Permission Table* 12
2.3 *Problem* 1 (Penjumlahan) 16
2.4 *Problem* 2 (*Max*) 17

3.1 Tabel Analisis Kebutuhan *Sharif Judge* 26

4.1 Perancangan Tabel *shj_logins* 61
4.2 Perincian fungsi *insert_to_logs* 61
4.3 Perincian fungsi *get_all_logs* 62
4.4 Perincian fungsi *consturct___* 63
4.5 Perincian fungsi *index* 63
4.6 Perincian fungsi *get_all_final_submission* 75
4.7 Perincian fungsi *get_all_user_assignments* 75
4.8 Perincian fungsi *consturct___* 77
4.9 Perincian fungsi *index* 77
4.10 Perincian fungsi *hof_details* 77

5.1 Lingkungan Perangkat Keras 87
5.2 Lingkungan Perangkat Lunak 87
5.3 Struktur Tabel *shj_logins* 88
5.4 Struktur Tabel *shj_settings* 88
5.5 Struktur Tabel *shj_assignments* 89

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*. Antarmuka *Sharif Judge* ditulis menggunakan bahasa pemrograman PHP (*framework CodeIgniter*) dan *backend* menggunakan *BASH* [1]. Selain berfungsi sebagai *grader*, *Sharif Judge* memiliki beberapa fungsi seperti deteksi plagiarisme jawaban para peserta. Cara kerja *grader* pada *Sharif Judge* dimulai dari dosen membuat *assignment*. Untuk membuat *assignment* dibutuhkan data-data seperti nama *assignment*, waktu mulai, waktu berhenti, waktu tambahan, daftar peserta, diskripsi soal dan kunci jawaban. Para peserta dapat mengunduh diskripsi soal lalu mengerjakan *assignment* tersebut. Peserta yang telah selesai mengerjakan *assignment*, dapat mengumpulkan jawaban dalam bentuk kode program. *Sharif Judge* akan menjalankan kode program dan menyesuaikan dengan kunci jawaban, lalu *grader* akan langsung menilai jawaban para peserta.

Sharif Judge digunakan oleh Jurusan Teknik Informatika Universitas Katolik Parahyangan pada mata kuliah seperti Algoritma dan Struktur Data serta Desain dan Analisis Algoritma. Perangkat lunak *Sharif Judge* 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 dan ujian. Mahasiswa juga dapat melihat nilai secara langsung setelah jawaban dikumpulkan. Para mahasiswa dapat memperbaiki jawaban yang telah dikumpulkan jika assignment yang dikerjakan tidak melewati batas waktu pengumpulan.

Pada prakteknya, perangkat lunak *Sharif Judge* terkini masih butuh pengembangan. Pengembangan tersebut dibutuhkan karena Jurusan Teknik Informatika memiliki kebutuhan yang lebih spesifik seperti *login* yang terintegrasi dengan server RADIUS Teknik Informatika, membatasi pengaksesan diskripsi soal pada assignment dan kebutuhan spesifik lainnya. *Sharif Judge* terakhir dicommit pada *Github* pada bulan Juli 2015 dan meninggalkan beberapa bug yang belum diperbaiki. Hal-hal di atas 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 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. Menentukan dan merancang fitur yang akan diimplementasi.
4. Mengimplementasikan fitur terhadap perangkat lunak.
5. Menguji 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, metodologi dan sistematika pembahasan.
2. Bab 2 : Landasan 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 analisis kebutuhan *Sharif Judge*.
4. Bab 4 : Perancangan
   Bab 4 membahas mengenai perancangan yang dilakukan sebelum masuk ke tahap 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 digunakan dalam skripsi ini. Landasan teori yang digunakan, 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 untuk mengembangkan proyek-proyek menjadi lebih cepat dibandingkan menulis kode dari awal. *Framework* ini memiliki banyak libary untuk fungsi-fungsi yang biasa diperlukan, serta antarmuka dan struktur logis yang sederhana untuk mengakses library ini. *CodeIgniter* membuat para pengguna lebih fokus pada proyek dengan cara meminimalkan jumlah kode yang dibutuhkan [2].

Beberapa keunggulan dari *CodeIgniter* yaitu:

- *Framework* yang Ringan
  Inti dari sistem *CodeIgniter* membutuhkan *library* yang kecil. Hal ini sangat berbeda dengan *framework* lain yang membutuhkan *resource* 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 Clean URLs* dan *search-engine friendly*. Clean URLs akan mempermudah pengguna dalam membaca URL. Contoh perbandingan URL biasa dengan Clean URLs: URL biasa: `\\example.com/index.php?page=news`, Clean URLs: `\\example.com/news`.

- *Packs a Punch*
  CodeIgniter dilengkapi dengan *library* yang umumnya diperlukan untuk mengembangkan web, seperti mengakses database, mengirim email, memvalidasi data *form*, menjaga *session*, memanipulasi gambar 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 rapih, 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.

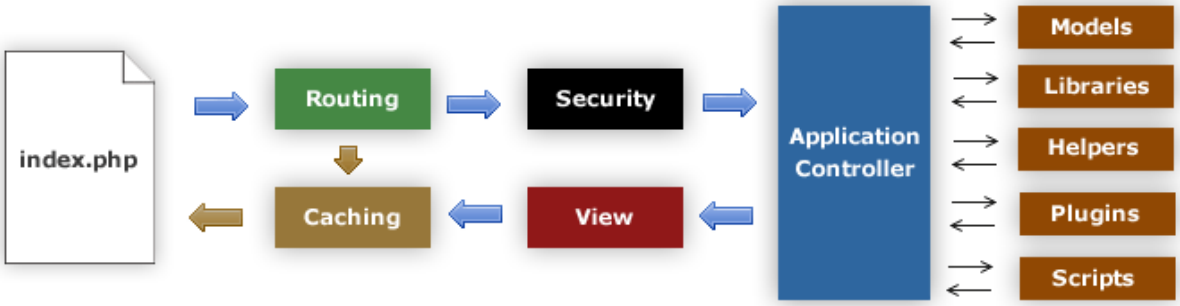
2.1.1 Fitur-fitur CodeIgniter

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 menunjukan 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 menggunaakan 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 sedikit karena presentasi terpisah dari *scripting* PHP [2].

2.1.3.1 *Model*

Model merepresentasikan bagian struktur data pengguna. Biasanya kelas *Model* 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 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 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`. Kode yang ditambahkan untuk menghubungkan *model* secara otomatis selama sistem berjalan adalah *$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 metode standar 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 berikut:

```
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 *cont*roller 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*, pengguna 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 dimasukan 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 yang 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 direktori tersebut. 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 tersebut, URI pengguna yang tampil akan seperti `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 hanya menggunakan *Controller* dan *View*. Hal tersebut dapat dilakukan jika pengguna tidak memerlukan adanya pemisahan tambahan atau pengguna merasa bahwa menggunakan 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 dieksekusi jika diminta. Tidak ada asumsi yang dibuat oleh sistem tentang apa yang mungkin diperlukan di luar *resource* utama, sehingga sistem 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 lebih fleksibel. Tujuan dari *framework* ini adalah sistem yang sangat longgar (*very loosely coupled system*).

- *Component Singularity*
  Singularity adalah sejauh mana komponen memiliki tujuan. 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*. Antarmuka *Sharif Judge* ditulis menggunakan bahasa pemrograman PHP (*framework CodeIgniter*) dan *backend* menggunakan *BASH* [1]. Selain sebagai *grader* otomatis, *Sharif Judge* juga memiliki beberapa fitur lainnya seperti:

- Beberapa peran (*role*) *users* (*admin, head instructor, instructor, student*)
- *Sandboxing* (belum diterpakan untuk *phyton*)
- 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.
- Menambahakan beberapa pengguna sekaligus
- Diskripsi *Problem* (*PDF/Markdown/HTML*)
- Penilaian ulang (*rejudge*)
- Papan nilai
- Notifikasi

2.2.1 Instalasi

Untuk menjalankan *Sharif Judge*, dibutuhkan 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 meginstal 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 adalah (*gcc, g++, javac, java, python2, python3 commands*)
- *Perl* lebih baik diinstal untuk alasan 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 direktori *application/cache/Twig* agar 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 *assigments* di luar *public directory* lalu simpan *path* lengkap di halaman *Settings*. Dua folder tersebut harus dapat ditulis oleh PHP. *File-file* yang diunggah akan disimpan di folder *assigments* 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* dibagi menjadi 4 *role*. Keempat *role* tersebut adalah *Admins, Head Instructor, Instructor,* dan *Students* Tabel 2.1 menunjukan *level users* [3].

Tabel 2.1: *User Roles Table*

User Role	***User Level***
Admin	3
Head Instructor	2
Instructor	1
Student	0

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

Tabel 2.2: *Permission Table*

Aksi	*Admin*	*Head Instructor*	*Instructor*	*Student*
Mengubah *Settings*	✓	✗	✗	✗
Menambah/Menghapus *users*	✓	✗	✗	✗
Mengubah Peran *users*	✓	✗	✗	✗
Menambah/Menghapus/Mengubah *Assignment*	✓	✓	✗	✗
Mengunduh *Test*	✓	✓	✗	✗
Menambah/Menghapus/Mengubah Notifikasi	✓	✓	✗	✗
Rejudge	✓	✓	✗	✗
Melihat/*Pause*/Melanjutkan/*Submission Queue*	✓	✓	✗	✗
Mendeteksi Kode yang Mirip	✓	✓	✗	✗
Melihat Semua Kode	✓	✓	✓	✗
Mengunduh Kode Final	✓	✓	✓	✗
Memilih *Assignment*	✓	✓	✓	✓
Submit	✓	✓	✓	✓

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

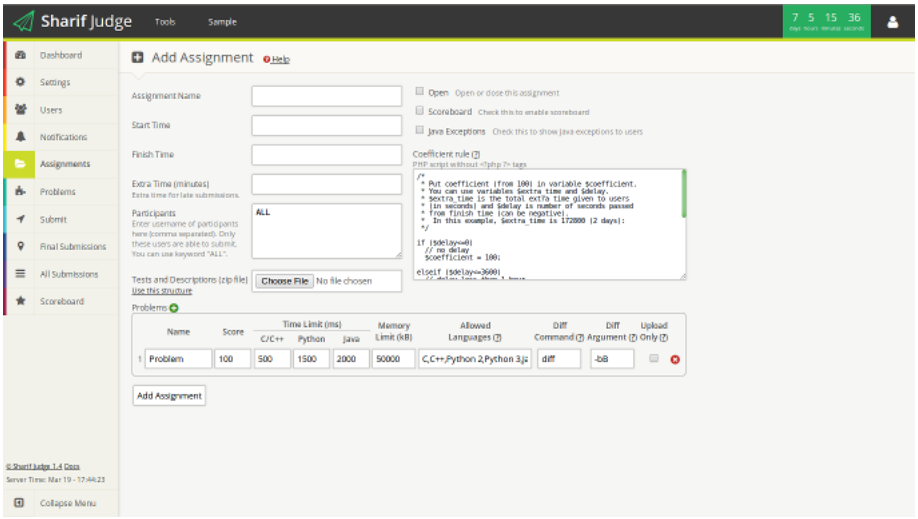
```
USERNAME EMAIL PASSWORD ROLE

* 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 *Assignment*

Pengguna dapat menambahkan *assignment* dengan cara mengklik *Add* di halaman *assigments* [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*
  Assignment akan ditampilkan dengan nama ini dalam daftar *assignment*.

- *Start Time*
  Users tidak dapat mengumpulkan *assignment* 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 *ssignment* setelah *Finish Time + Extra Time*. *Assignment* yang telat akan dikalikan dengan koefisien tertentu. Pengguna harus menulis *script* PHP untuk menghitung koefisien pada bidang *"Coefficient Rule"*. Gunakan format berikut 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 peserta. Hanya *users* ini yang dapat mengumpulkan *assignment*. Pengguna dapat menggunakan kata kunci *ALL* untuk mengijinkan semua *users* agar dapat mengumpulkan *assignment*. Contoh: *admin, instructor1 , instructor2 ,student1 , student2,student3 , student4*.

- *Open*
  Pengguna dapat membuka atau menutup *assignment* menggunakan pilihan ini. Jika pengguna menutup *assignment*, *non-student users* masih dapat mengumpulkan *assignment*.

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

- *Java Exceptions* Pengguna dapat mengaktifkan dan mematikan *java exceptions* yang ditunjukan 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 *Python* dan *Java* 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 *problem* (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 *problem* sebagai *Upload-Only*, maka *Sharif Judge* tidak akan menilai *assignment* pada *problem* tersebut. Pengguna dapat menggunakan *zip* dan *PDF* pada *allowed languages* jika mengaktifkan pilihan ini.

2.2.4.1 Contoh *Assignment*

Berikut contoh *assignment* untuk mencoba *Sharif Judge*. Menambah *assignment* dengan mengklik *Add* di halaman *Assignment*. *Assignment* dibagi menjadi 3 *problem*:

1. *Problem* 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: *Problem* 1 (Penjumlahan)

Sample Input	Sample Output
5 54 78 0 4 9	145

2. *Problem* 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: *Problem* 2 (*Max*)

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

3. *Problem* 2 (*Upload!*)
   Pengguna diharuskan mengunggah sebuah *file C* atau *zip*. *Problem* 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
```

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

2.2.4.2 Contoh Solusi

Problem diatas dapat diselesaikan menggunakan contoh solusi berikut ini:

- Solusi *Problem* 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 *Problem* 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 *assignment*. *File zip* ini dapat berisikan folder-folder untuk setiap *problem*. Pengguna harus memberikan nama pada folder sesuai aturan seperti *p1, p2, p3, dst*. *Assignment* yang menggunakan pilihan *Upload-Only* tidak membutuhkan *folder* [3].

2.2.5.1 Metode Pengecekan

Sharif Judge memiliki dua metode pengecekan untuk setiap *problem* 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 problem*. *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.text, 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 jika salah 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* penguji 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
```

Problem 1 menggunakan metode *"Tester* untuk mengecek hasil keluaran, sehingga memiliki *file tester.cpp*. Berikut isi dari *file tester.cpp* untuk *problem* 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;

}
```

Problem 2 menggunakan metode *"Input/Output Comparison"* untuk mengecek hasil keluaran, sehingga memiliki dua folder *in* dan *out* yang berisikan *test cases*. *Problem* 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 lalukan gunakan pada *Sharif Judge* untuk mendetksi kecurangan. Pengguna dapat menyimpan *user id* di *Sharif Judge* pada halaman *Moss*. *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 *assignment* berakhir, karena para peserta masih dapat mengubah *Final Submissions* masing-masing sebelum waktu habis. Dengan cara tersebut *Sharif Judge* dapat mengirimkan *Final submissions* para peserta ke *server 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 *repository 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 kini pada perangkat lunak *Sharif Judge*.

3.1 Analisis Kebutuhan Perangkat Lunak *Sharif Judge*

Analisis dilakukan dengan menganalisis setiap isu terbuka yang ada pada *repository* di *GitHub*. Setiap isu di *GitHub* terdapat kode unik yang dimulai dengan tanda '#' dan diikuti dengan angka. Kode unik tersebut menunjukan 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	*Security with PHP*	*GitHub*	#61	kathiedart	Diimplemntasi
2	*Securing Assignment*	*GitHub*	#55	wojcik13	Tidak diimplementasi
3	*New Function*	*GitHub*	#53	wojcik13	Tidak diimplementasi
4	*Solved Problem Indicator*	*GitHub*	#46	atiabjobayer	Tidak diimplementasi
5	*Some Problem & Sugestion*	*GitHub*	#45	atiabjobayer	Tidak diimplementasi
6	*Queue failed to process if submission take too long to complete?*	*GitHub*	#32	truongan	Diimplemntasi
7	*Compilation Error on all language*	GitHub	#34	Eririn07	Tidak diimplementasi
8	Membatasi soal (deskripsi dan PDF) hanya bisa diakses saat *assignment* "*open*" dan setelah waktu mulai	Dosen	AIF102	Husnul Hakim	Diimplementasi
9	Menguji kemiripan kode antar mahasiswa (Contek)	Dosen	AIF102	Husnul Hakim	Tidak diimplementasi
10	1 Akun hanya dapat *login* 1 waktu (Jika suatu akun sedang *login*, tidak ada lagi yang bisa *login* akun tersebut)	Dosen	AIF102	Husnul Hakim	Diimplementasi
11	Membatasi soal (deskripsi dan PDF) hanya bisa diakses saat *assignment* "*open*" dan setelah waktu mulai	Dosen	AIF102	Vania Natali	Diimplementasi
12	*Sharif Judge* tidak dapat menerima *file* dengan ekstensi *.txt untuk *Pre-Test*	Dosen	AIF102	Vania Natali	Diimplementasi
13	Membatasi soal (deskripsi dan PDF) hanya bisa diakses saat *assignment* "*open*" dan setelah waktu mulai	Dosen	AIF202	Luciana Abednego	Diimplementasi
14	*Sharif Judge* tidak dapat menerima *file* dengan ekstensi *.txt untuk *Pre-Test*	Dosen	AIF202	Luciana Abednego	Diimplementasi
15	Perlu ditambah petunjuk penamaan *file input* & *output* yg langsung bisa dilihat ketika hendak *upload*	Dosen	AIF202	Luciana Abednego	Tidak diimplementasi
16	Membatasi soal (deskripsi dan PDF) hanya bisa diakses saat *assignment* "*open*" dan setelah waktu mulai	Dosen	AIF202	Joanna Helga	Diimplementasi
17	*Register* peserta yg *mode batch*, *Sharif Judge* tidak minta nama orangnya (lebih baik ada nama orangnya)	Dosen	AIF202	Joanna Helga	Diimplementasi
18	Nama peserta seharusnya tidak bisa diganti (Bisa menjadi "mainan" dan tindak kecurangan karena dapat memberikan *hint*)	Dosen	AIF202	Joanna Helga	Diimplementasi
19	Ingin memiliki fungsi dimana *Assignment* tidak memiliki batasan waktu (arsip soal tahun lalu dapat dikerjakan kapan saja)	Dosen	AIF202	Joanna Helga	Diimplementasi
20	Ingin memiliki *scoreboard global* untuk semua *assignment*.	Dosen	AIF202	Joanna Helga	Diimplementasi
21	Membatasi soal (deskripsi dan PDF) hanya bisa diakses saat *assignment* "*open*" dan setelah waktu mulai	Dosen	AIF102 & AIF202	Claudio Fransiscus	Diimplementasi
22	*Sharif Judge* tidak dapat menerima *file* dengan ekstensi *.txt untuk *Pre-Test*	Dosen	AIF102 & AIF202	Claudio Fransiscus	Diimplementasi
23	UI masih merepotkan	Dosen	AIF102 & AIF202	Claudio Fransiscus	Tidak diimplementasi
24	UI ada yang tidak berguna (yang lebih banyak digunakan *assignment, submit, scoreboard,* dan hasil *submit*	Dosen	AIF102 & AIF202	Claudio Fransiscus	Tidak diimplementasi

25	Ingin memiliki fungsi *reminder*. Banyak mahasiswa lupa mengerjakan tugas dan tidak bisa mengumpulkan. Fungsi *reminder* akan mengirimkan *reminder* ke *email* mahasiswa	Dosen	AIF102 & AIF202	Claudio Fransiscus	Tidak diimplementasi
26	Membatasi soal (deskripsi dan PDF) hanya bisa diakses saat *assignment* "*open*" dan setelah waktu mulai	Dosen	AIF102 & AIF202	Pascal Alfadian	Diimplementasi
27	Integrasi login ke *RADIUS* (*password* sama dengan *login Windows*)	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 *Excel* tidak berfungsi pada halaman *Submission*	Asisten Dosen	AIF202	Kresna Dwi Cahyo	Diimplementasi

Pada tabel di atas teradapat beberapa kolom, yaitu:

- Deskripsi
  Jika sumber kebutuhan berasal dari *GitHub*, maka pada kolom deskripsi akan ditulis judul dari isu yang ada pada *repository*. 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. Keterangan kode mata kuliah AIF102 = Algoritma & Struktur Data, AIF202 = Desain & Analisis Algoritma dan AIF401 = Skripsi 1.

- 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 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 diperbaiki. Solusi yang ditawarkan untuk mengatasi hal di atas adalah menambahakan *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 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.

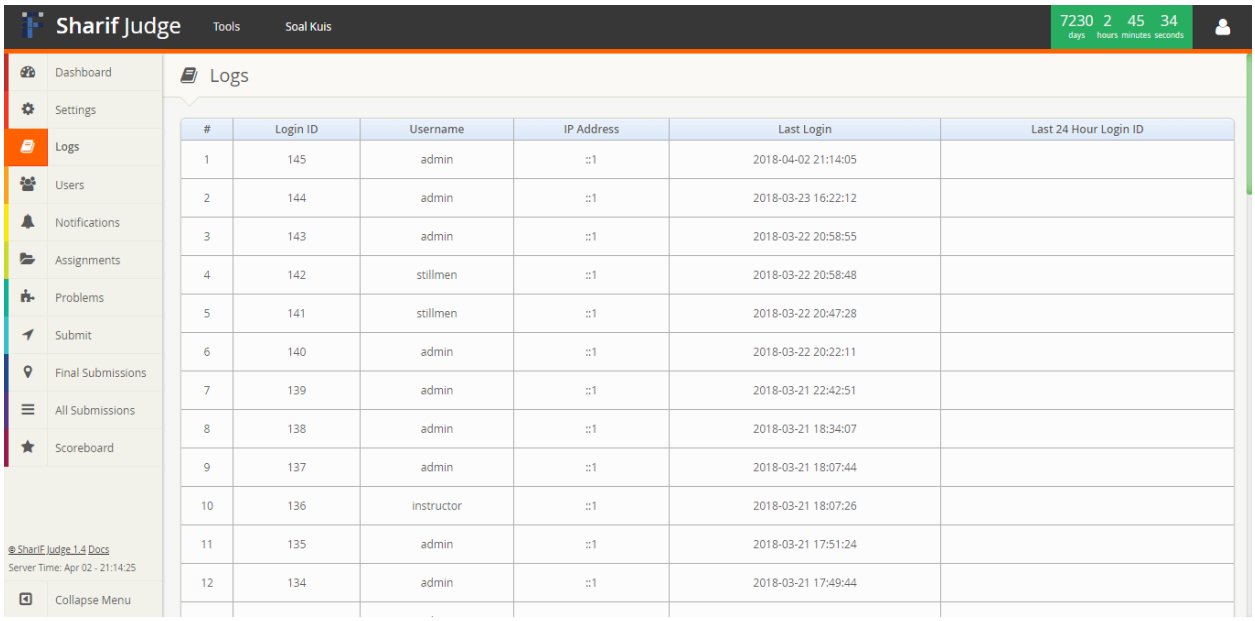
3.1.10 Satu Akun Hanya Dapat Login Satu Waktu

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 diimplementasikan ke dalam *Sharif Judge*.



Gambar 3.1: *Mockup* Halaman *Logs*

Halaman ini terletak di paling bawah menu lainnya 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 mengumupulkan jawaban *Pre-test*, harus terlebih dahulu mengubah ekstensi *file* menjadi Java atau mengompres *file* ke dalam Zip.

Pada skripsi ini, kebutuhan di atas diimplementasi. Solusi yang ditawarkan untuk kebutuhan di atas yaitu menambahkan *file* dengan format TXT agar dapat dikumpul ke *Sharif Judge*. *Assignment* yang digunakan merupakan *assignmnent* 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 diimplementasi. Solusi yang ditawarkan untuk memenuhi kebutuhan di atas yaitu menambahkan parameter *Display Name* pada saat pendaftaran peserta. Pendaftaran peserta menggunakan 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 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 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.

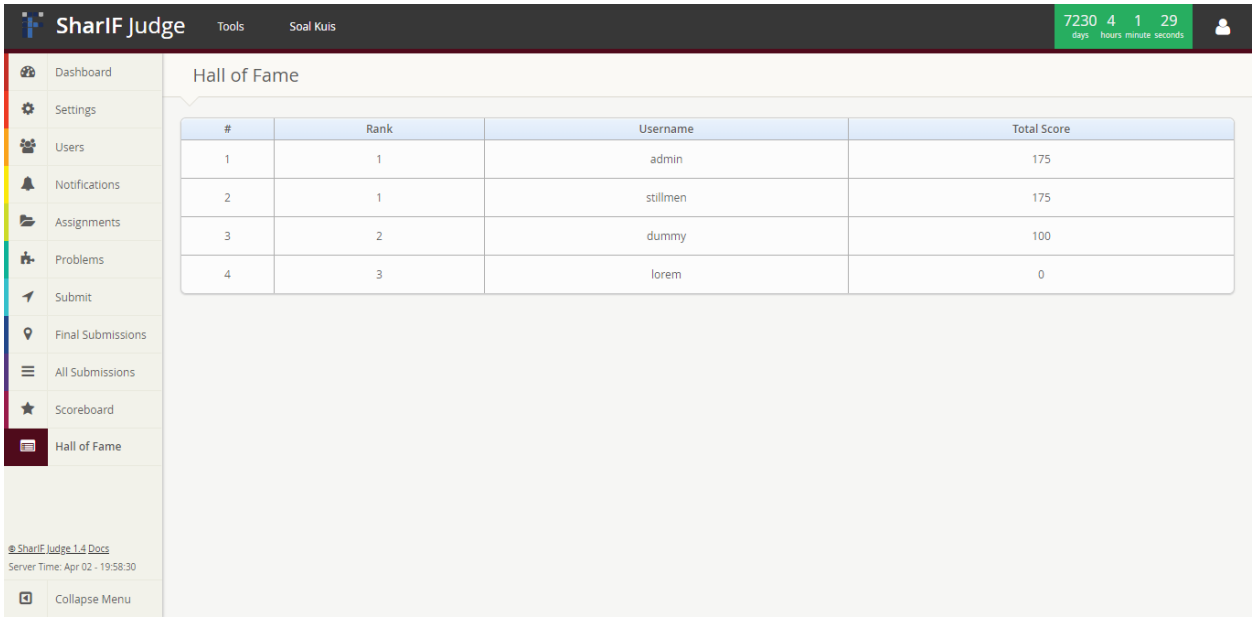
Pada skripsi ini, kebutuhan di atas diimplementasi. Solusi yang ditawarkan untuk memenuhi kebutuhan di atas yaitu membuat sebuah fitur tambahan pada *assignment*. *Assignment* yang mengaktifkan fitur ini tidak akan muncul pada kalendar yang terdapat di halaman *Dashboard*. Fitur tersebut akan membuat 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*. Tanggal 18 Januari 2038 waktu UTC+7 merupakan 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 menampilkan total skor beserta detail skor setiap *problem* yang telah dikerjakan para peserta diseluruh *assignment* yang ada.

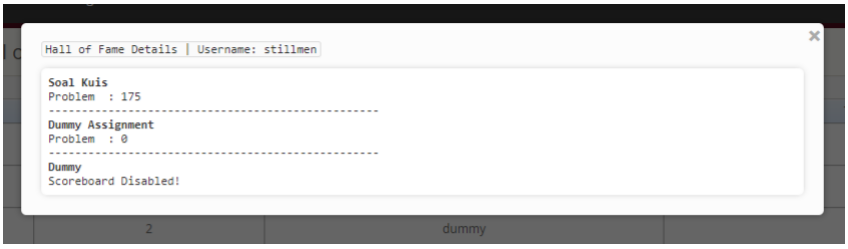
Pada skripsi ini, kebutuhan di atas diimplementasi. Solusi yang ditawarkan untuk memenuhi kebutuhan di atas yaitu membuat halaman baru yang diberi nama *Hall of Fame*. Halaman *Hall of Fame* menampilkan berapa *problem* yang telah dikerjakan oleh para peserta diseluruh *assignment* yang ada. Nama peserta 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 diimplementasikan ke dalam perangkat lunak *Sharif Judge*.



Gambar 3.2: *Mockup* Halaman *Hall of Fame*

Halaman ini terletak di bawah menu halaman *Scoreboard* dan dapat diakses oleh seluruh *role*. Pada halaman Hall of Fame diberlakukan sistem rangking bedasarkan 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.



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 diselsaikan.

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 *Server 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 lainnya [1]. 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.

Pada skripsi ini, kebutuhan di atas 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*.

[1] Cisco, "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)

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. Fitur ini tidak diimplementasi karena belum selesai setelah UTS Skripsi 2. Fitur ini telah dicommit ke *repository* pathhttps://github.com/ayenz/Sharif-Judge/ di *GitHub* pada *branch* Skripsi_29.

3.1.30 Branding Teknik Informatika

Branding Teknik Informatika dilakukan dengan cara mengubah logo dan ikon *Sharif Judge* menjadi logo Teknik Informatika. Selain mengubah logo dan ikon, perubahan juga 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 penggunanya [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

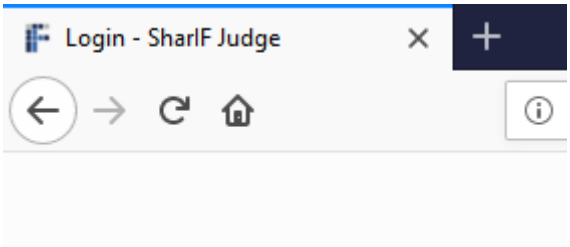
Pada skripsi ini, kebutuhan di atas diimplementasi. Beberapa logo yang digunakan seperti:

Gambar 3.4: Logo dan Ikon

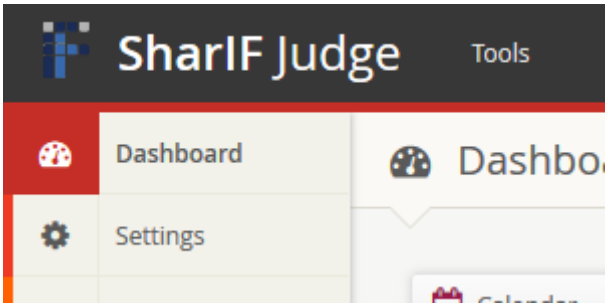
Gambar 3.5: *Banner Sharif Judge*

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].

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

Pada skripsi ini, kebutuhan di atas 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.

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 kelas 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 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 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 dipanggil oleh *update_scoreboard*
- *update_scoreboards*: mengupdate *cache scoreboard* dari semua *assignment*. Fungsi ini 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 *u*ser 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*: memperbaru *First Login Time* dan *Last Login Time username* tertentu

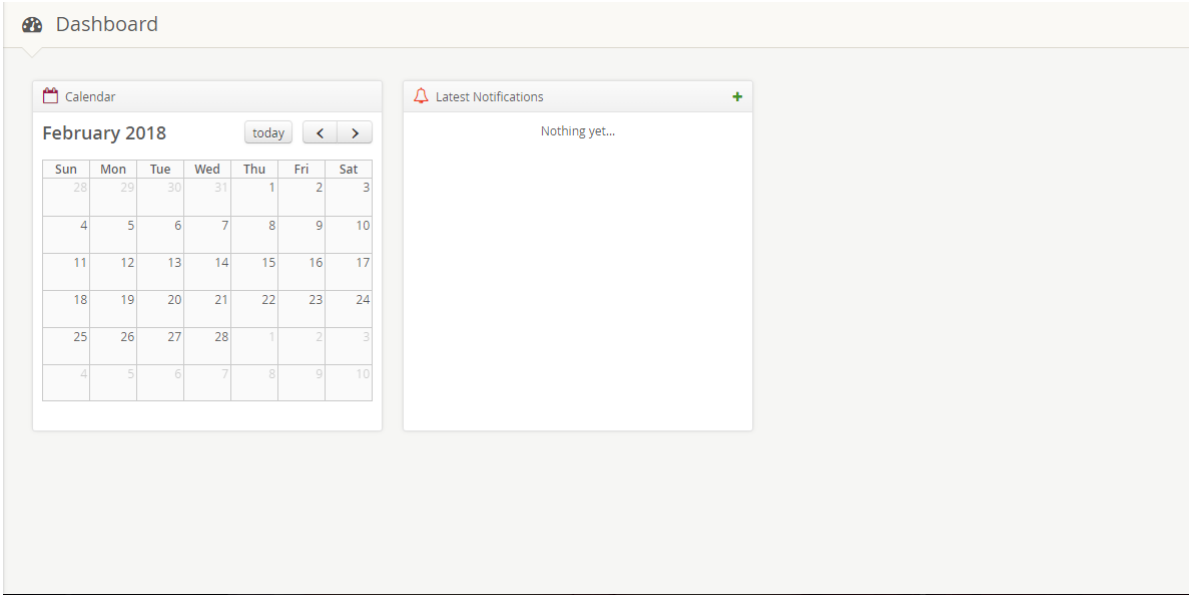
3.2.2 *View*

Pada perangkat lunak *Sharif Judge*, kelas view menggunakan *framework* aplikasi yaitu *Twig*. *Twig* merupakan sebuah *template engine modern* untuk *PHP*. Beberapa kelebihannya 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 kelas *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 authtentication. *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*.

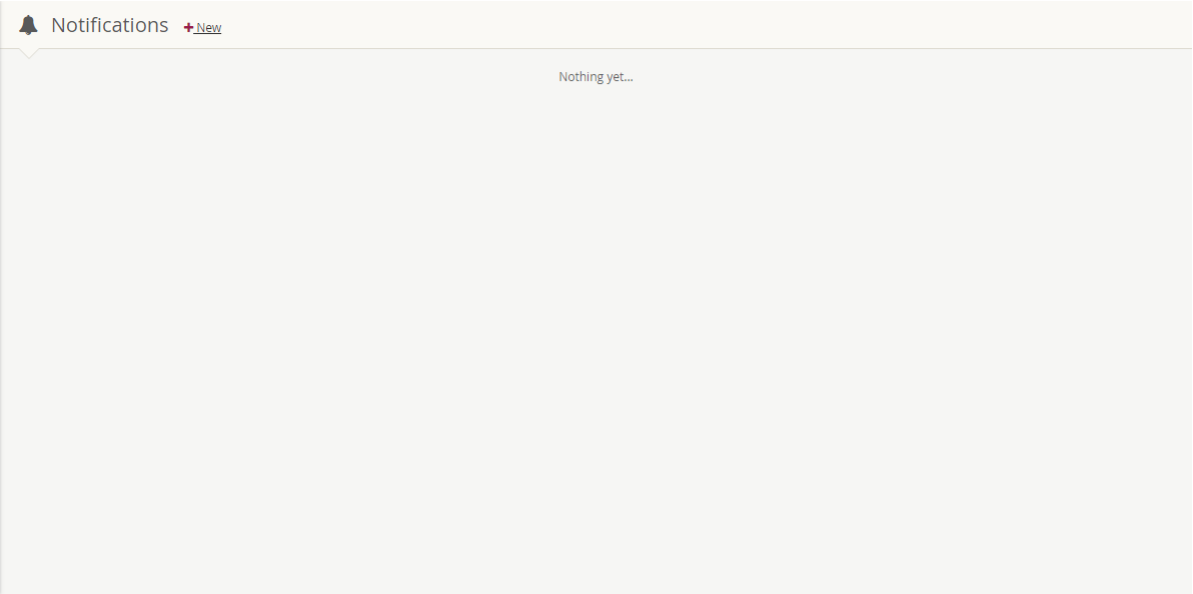
3.2.2.1 *dashboard.twig*



Gambar 3.9: *Dashboard*

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

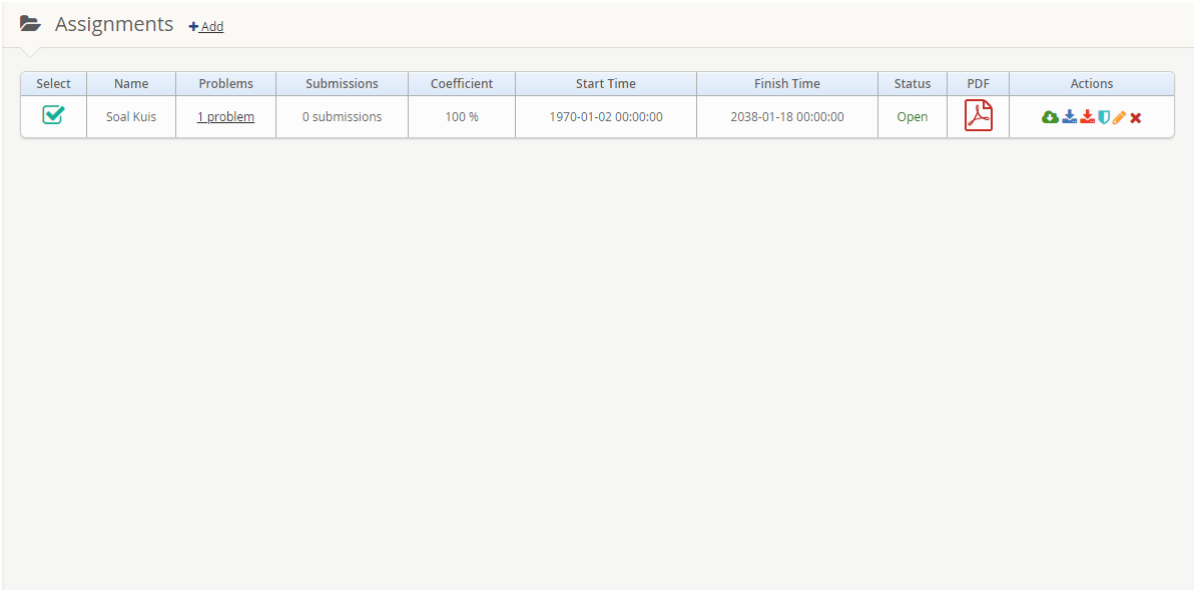
3.2.2.2 *notifications.twig*



Gambar 3.10: *Notifications*

Halaman notifikasi 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 munucl pada tabel notifikasi di *dashboard*.

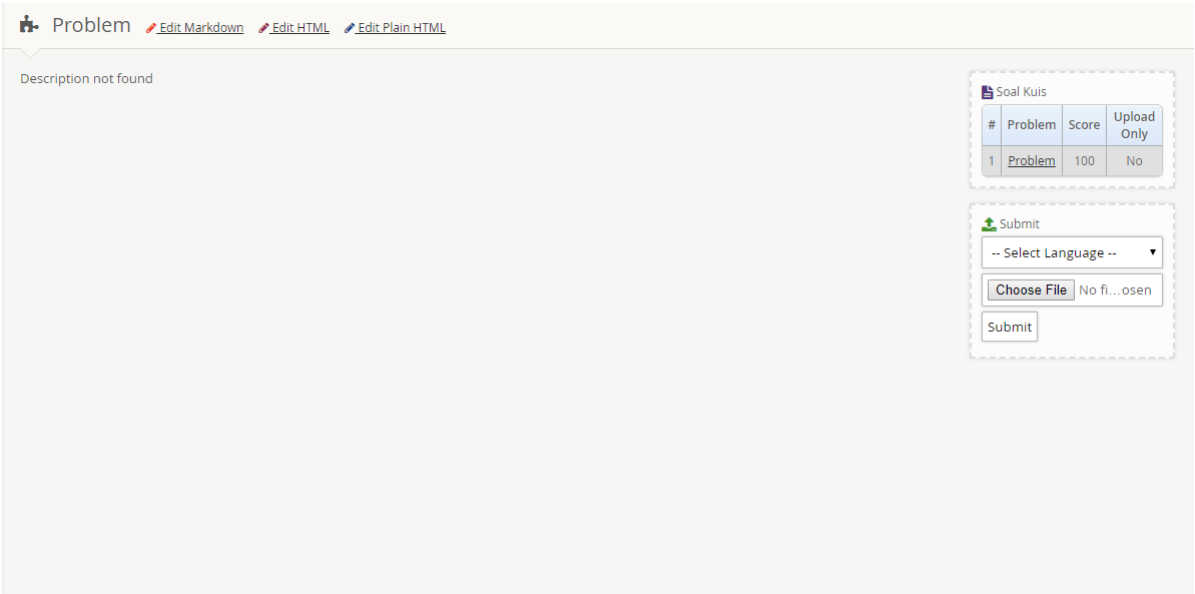
3.2.2.3 *assignments.twig*



Gambar 3.11: *Assignments*

Halaman *assignments* 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*

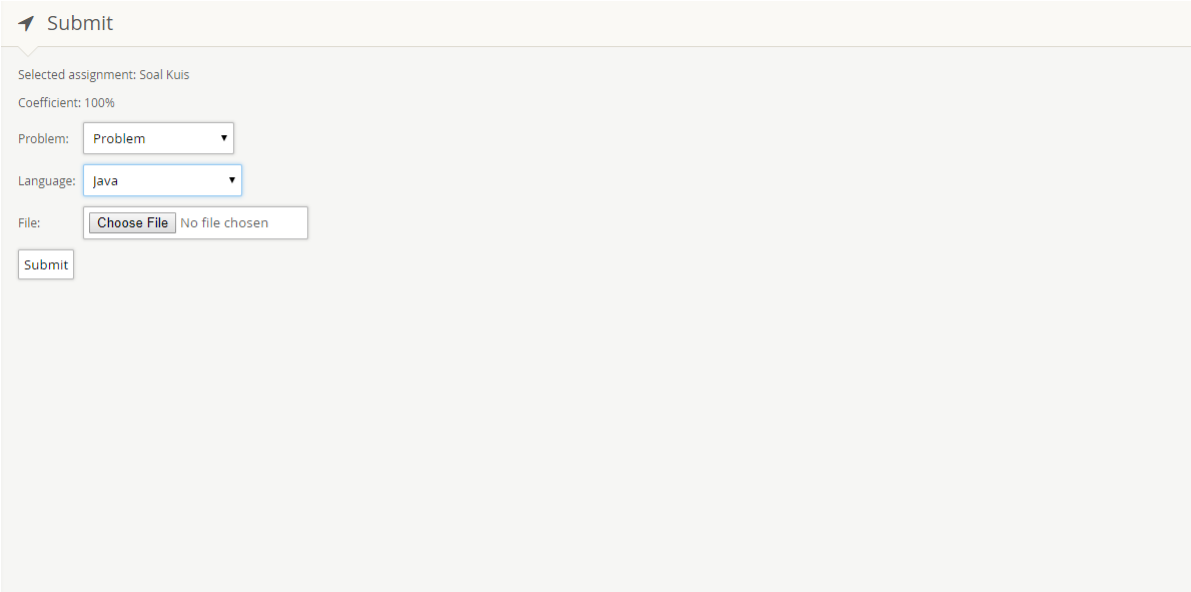


Gambar 3.12: *Problems*

Halaman *problems* 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.

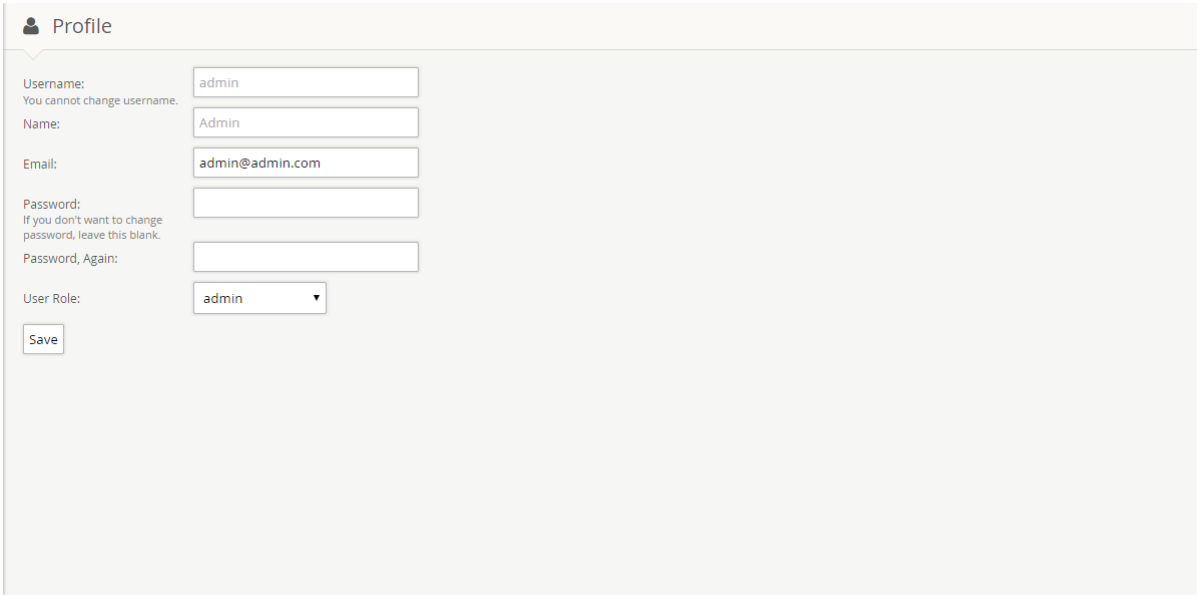
3.2.2.5 *submit.twig*



Gambar 3.13: *Submit*

Halaman *submit* berfungsi untuk mengumpulkan jawaban dari *assignment* yang telah dipilih pengguna sebelumnya. Pengguna dapat memilih masalah telah diselsaikan 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.

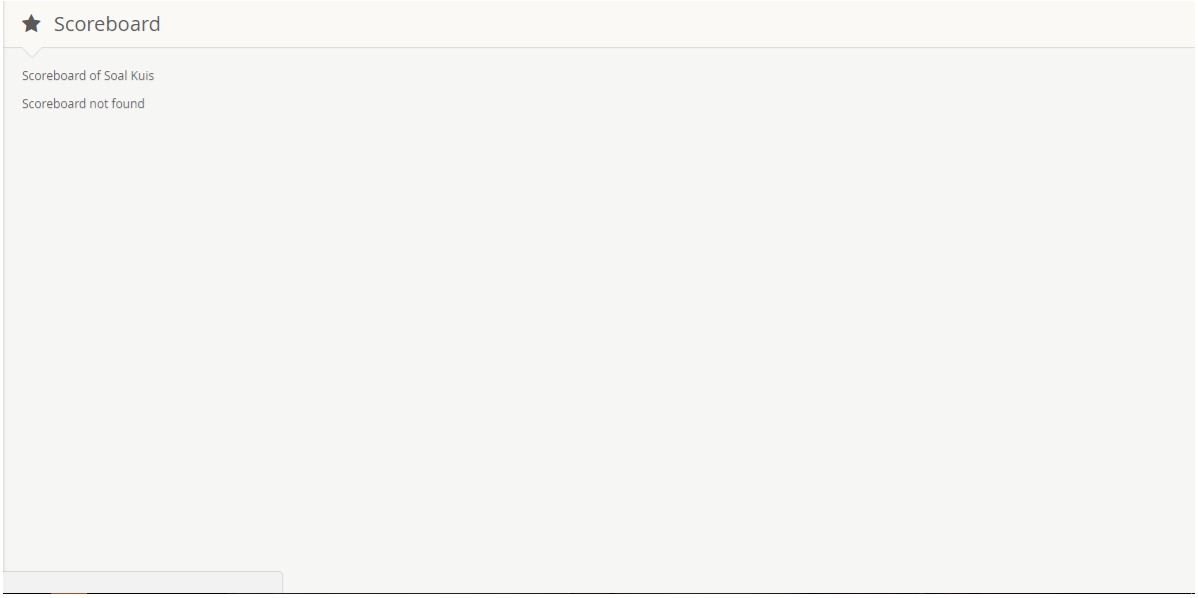
3.2.2.6 *profile.twig*



Gambar 3.14: *Profile*

Halaman *profile* 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*.

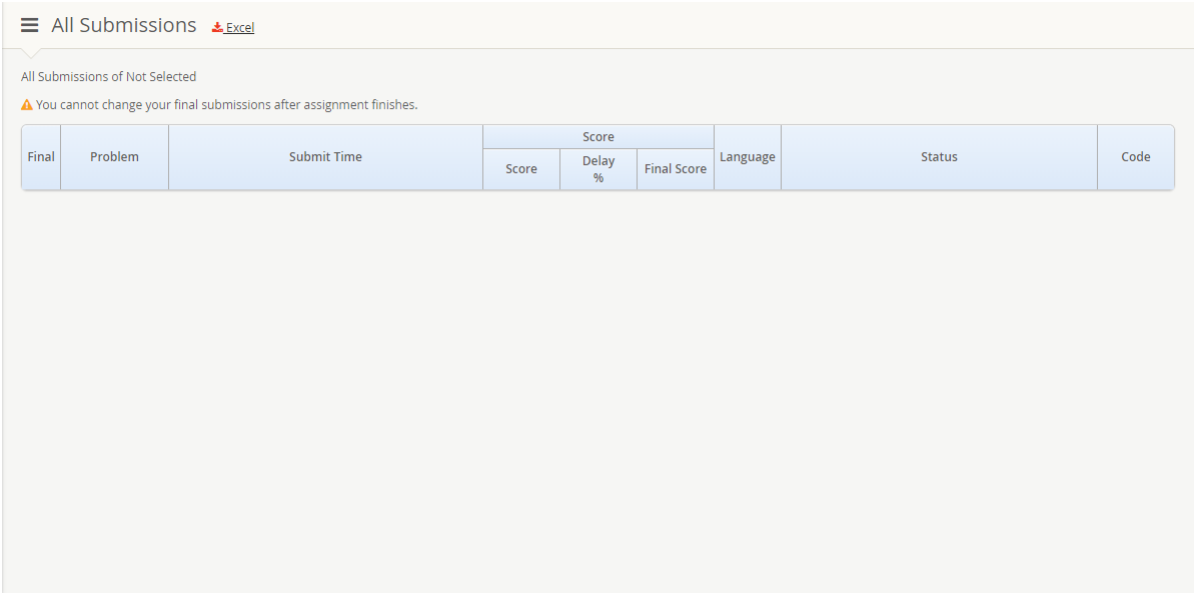
3.2.2.7 *scoreboard.twig*



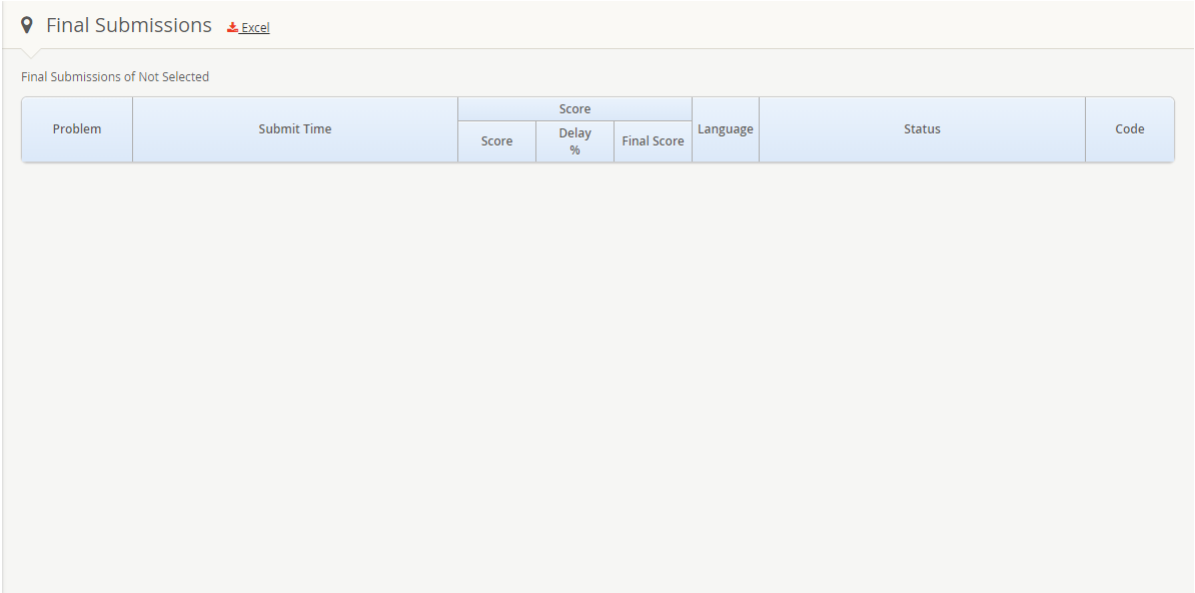
Gambar 3.15: *Scoreboard*

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

3.2.2.8 *submission.twig*



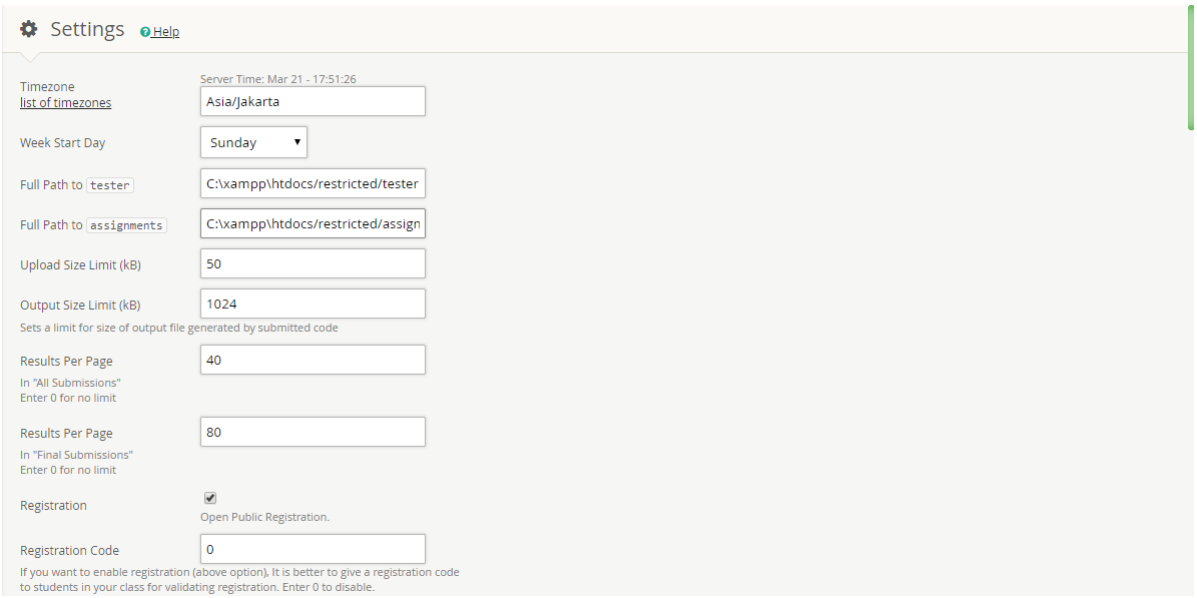
Gambar 3.16: *All Submission*



Gambar 3.17: *Final Submission*

File submission.twig 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*.

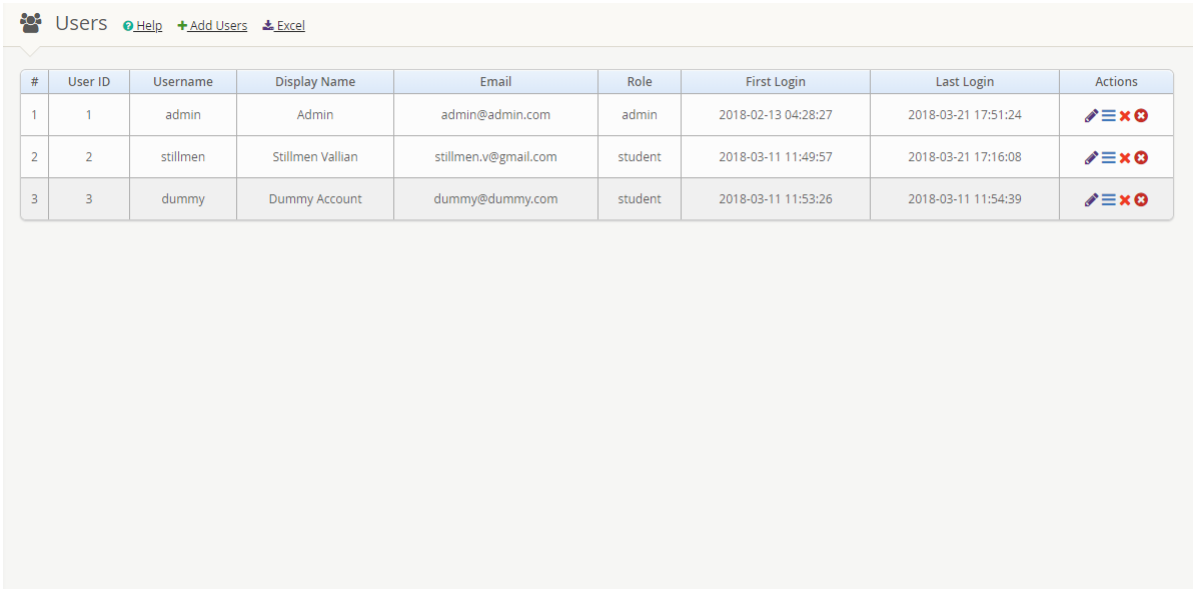
3.2.2.9 *settings.twig*



Gambar 3.18: *Settings*

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

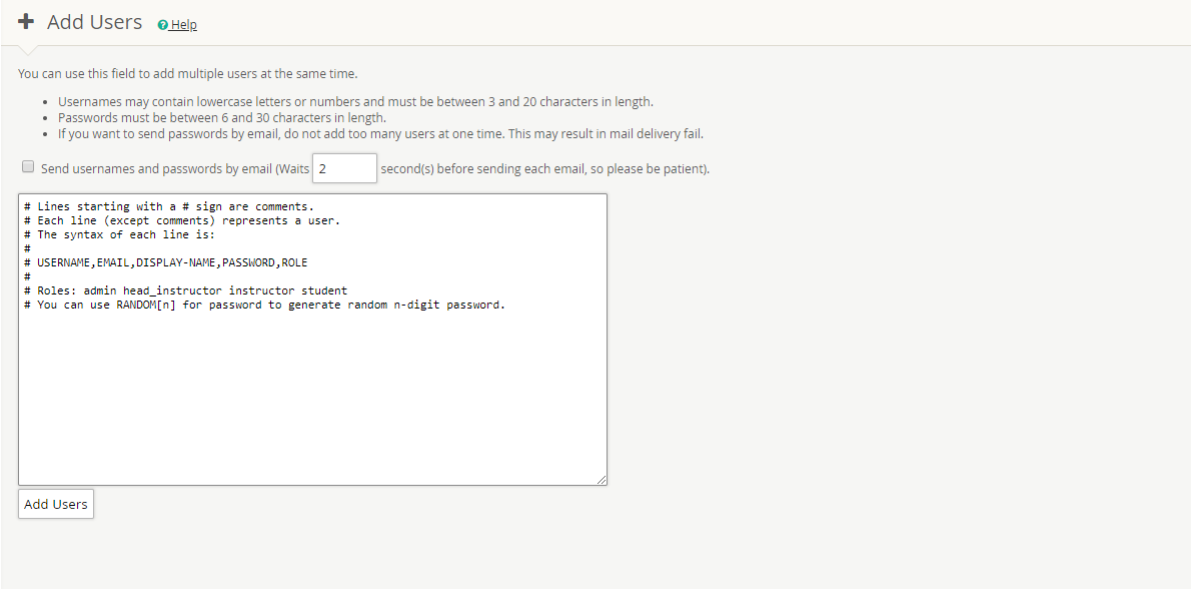
3.2.2.10 *user.twig*



Gambar 3.19: *User*

Halaman *user* 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.

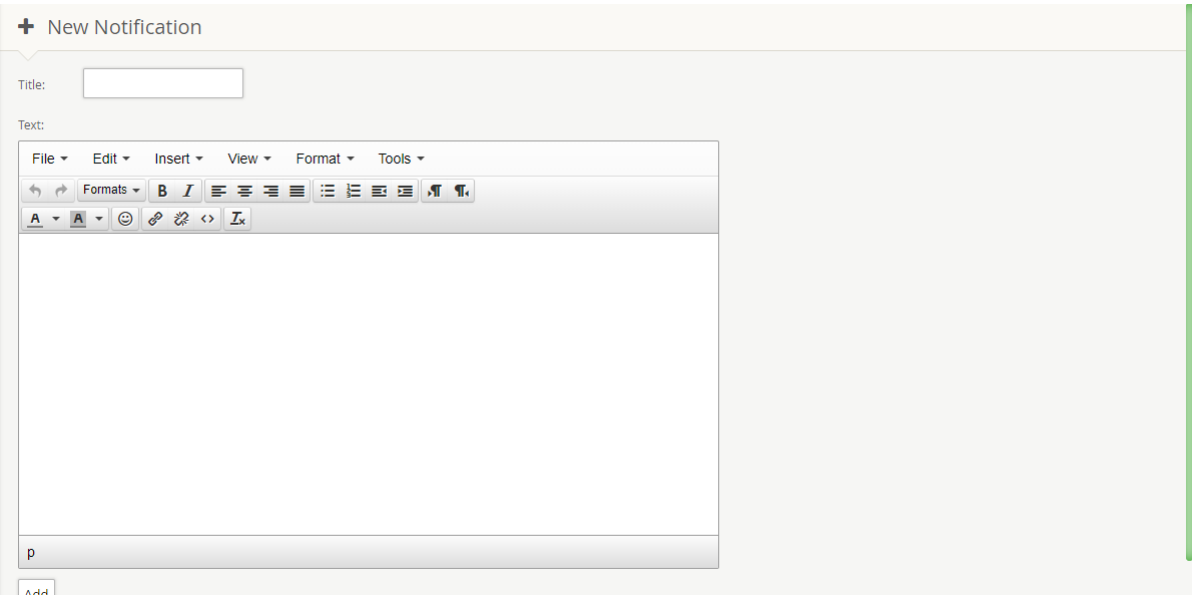
3.2.2.11 *add_user.twig*



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*.

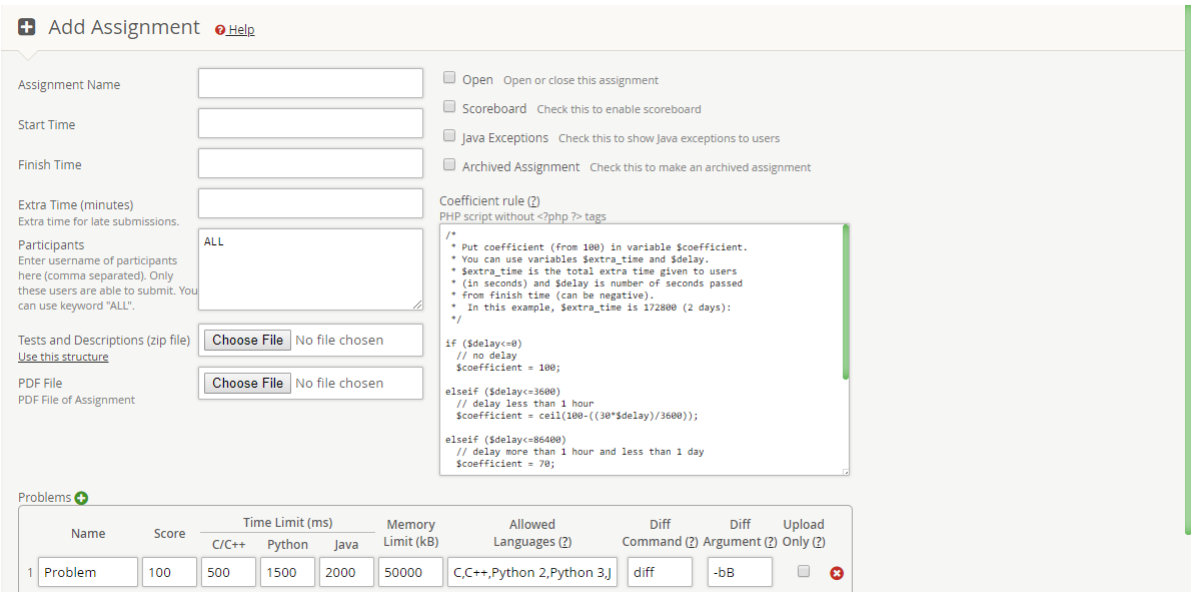
3.2.2.12 *add_notification.twig*



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.

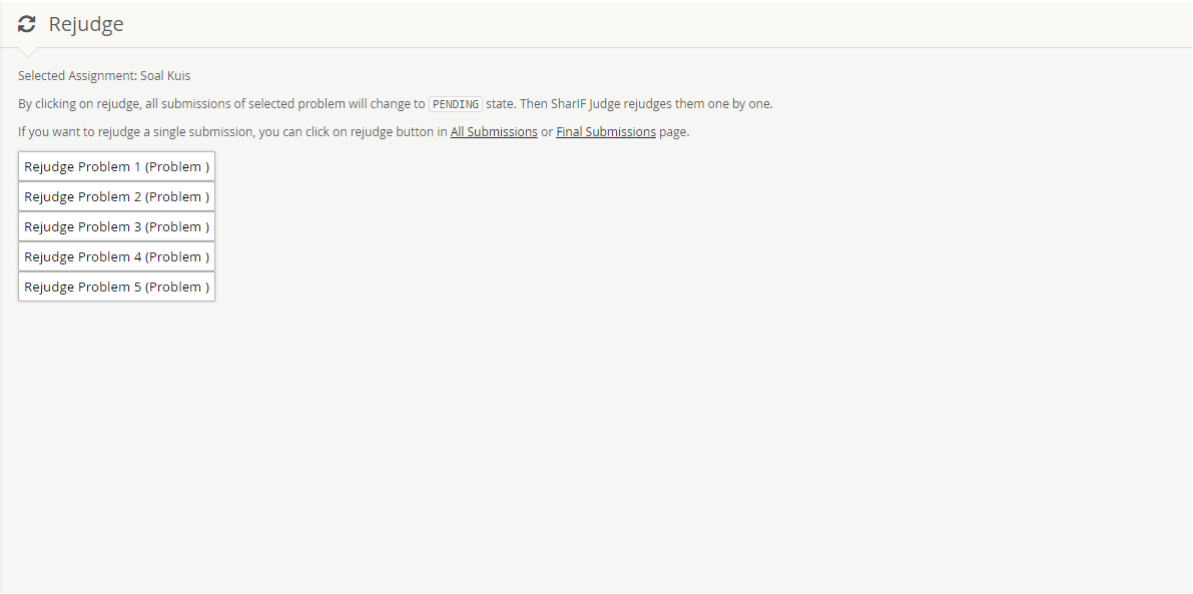
3.2.2.13 *add_assignment.twig*



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.

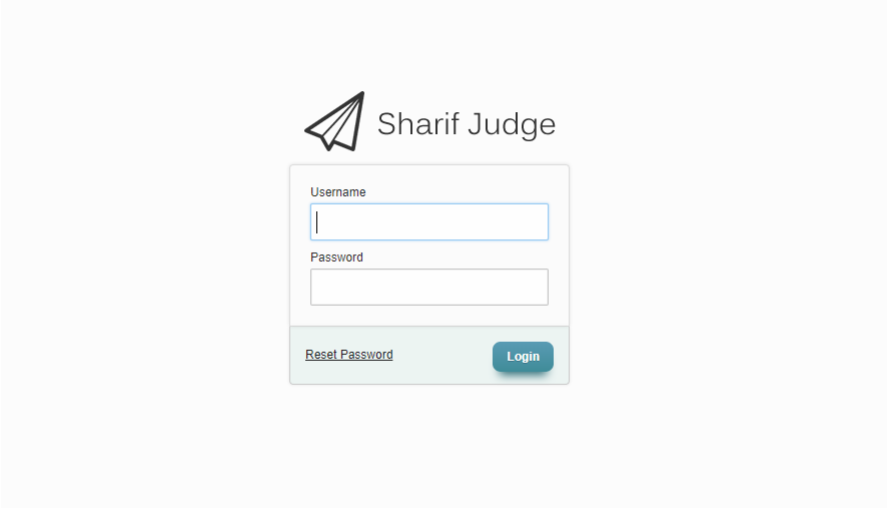
3.2.2.14 *rejudge.twig*



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.

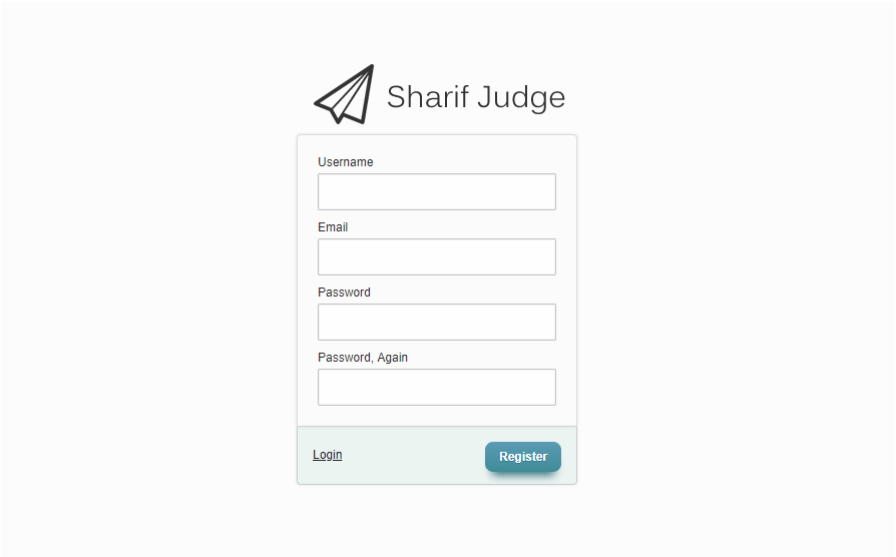
3.2.2.15 *login.twig*



Gambar 3.24: *login*

Halaman login merupakan halaman pertama yang tampil ketika pengguna membuka *Sharif Judge*. Untuk dapat menggunakan *Sharif Judge*, para pengguna harus memasukan 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*



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 kelas *controller* yang berisikan fungsi-fungsi sebagai perantara *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 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*: mempersiapakan *form* registrasi dan memproses tampilan pada *register.twig*
- *logout*: keluar dari *Sharif Judge* dan mengalihkan pengguna ke halanan *login*
- *lost*: mempersiapakan *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 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 masalalah 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* terentu
- *final_excel*: mengunuduh *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 antrean 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 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 kelas *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 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 Rekoneksi ke *Database*

Method rekoneksi ke *database* ditambahkan pada kelas *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 Pengaksesan Soal (deskripsi & PDF)

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 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(
    $assignment_id)['finish_time']);
103             $starttime = strtotime($this->assignment_model->assignment_info(
    $assignment_id)['start_time']);
104             $extratime = $this->assignment_model->assignment_info(
    $assignment_id)['extra_time'];
105
106             // Find pdf file
107             if ($problem_id === NULL)
108                     $pattern = rtrim($this->settings_model->get_setting('
    assignments_root'),'/')."/assignment_{$assignment_id}/*.pdf";
109             else
110                     $pattern = rtrim($this->settings_model->get_setting('
    assignments_root'),'/')."/assignment_{$assignment_id}/p{$problem_id}/*.pdf";
111             $pdf_files = glob($pattern);
112             if ( ! $pdf_files )
113                     show_error("File not found");
114             elseif (!$this->assignment_model->assignment_info($assignment_id)
    ['open'])
115                     show_error('Selected assignment has been closed.');
116             elseif  ( ! $this->assignment_model->is_participant($this->
    assignment_model->assignment_info($assignment_id)['participants'],$this->user
    ->username) )
117                     show_error('You are not registered for submitting.');
```

```
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     }
...
```

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 kelas. Beberapa kelas tersebut antara lain *controller Submit.php, model Assignment_model.php, view submissions.twig* dan kelas 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

Submit.php

```
...
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 kelas 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 Menambahkan Halaman *Logs*

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

Tabel 4.1: Perancangan Tabel *shj_logins*

Atribut	**Tipe Data**	**Ukuran**	**Default**
login_id (primary key)	int	11	None
username	varchar	20	None
ip_address	varchar	15	None
timestamp	timestamp	11	current_timestamp
last_24h_login_id	int	11	null

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 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 ditambahkan kelas *model, view* dan *controller*.

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

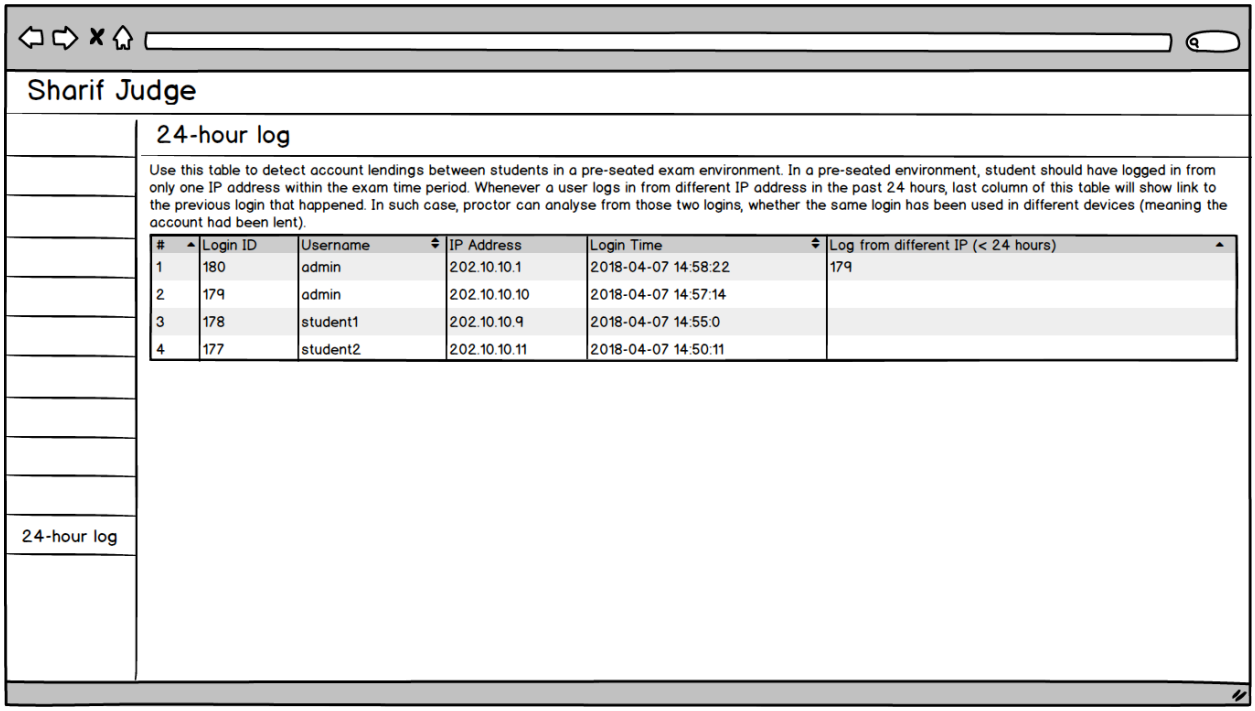
Tabel 4.2: Perincian fungsi *insert_to_logs*

Nama *Method*	*insert_to_logs*
Parameter *Input*	*$username* dan *$ip_address*
Parameter *Output*	-
Tabel yang berhubungan	*shj_logins*
Deskripsi	Proses untuk memasukan *logs* pengguna *Sharif Judge*
Algoritma	• Mengecek dan menghapus *logs* pada tabel *shj_logins* yang *timestampnya* lebih dari 24 jam. • Mengecek entri *login* terakhir untuk *$username* yang menggunakan *IP address* tidak sama dengan *$ip_address* • Jika tidak memiliki hasil, maka tambahkan entri baru menggunakan *$username* dan *$ip_address* tersebut. • Jika memiliki hasil, maka tambahkan entri baru menggunakan *$username* dan *$ip_address* serta. last_*24h_login_id* diisi dengan *login_id* sebelumnya

Tabel 4.3: Perincian fungsi *get_all_logs*

Nama *Method*	*get_all_logs*
Parameter *Input*	-
Parameter *Output*	semua entri logs dari tabel *shj_logins*
Tabel yang berhubungan	*shj_logins*
Deskripsi	Proses untuk mengembalikan entri *logs* yang terdapat pada tabel *shj_logins*
Algoritma	• Mengembalikan seluruh entri logs yang terdapat pada tabel *shj_logins* dalam bentuk *array*.

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



Gambar 4.1: Rancangan tampilan halaman *logs*

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

Tabel 4.4: Perincian fungsi *consturct___*

Nama *Method*	*consturct___*
Parameter *Input*	-
Parameter *Output*	-
Tabel yang berhubungan	-
Deskripsi	membatasi pengguna yang dapat mengakses halaman *logs*
Algoritma	• Mengecek *session* pengguna yang akan mengakses halaman *logs*. • Jika *session* tidak berstatus '*logged_in*, maka pengguna akan dialihkan ke halaman *login*. • Mengecek *role* pengguna yang akan mengakses halaman *logs*. • Jika role pengguna bukan *admin*, maka pengguna akan dialihkan ke halaman *'404 Not Found'*.

Tabel 4.5: Perincian fungsi *index*

Nama *Method*	*index*
Parameter *Input*	-
Parameter *Output*	-
Tabel yang berhubungan	*shj_logins*
Deskripsi	Proses untuk memuat seluruh entri *logs* pada halaman *logs.twig*
Algoritma	• Memuat data *logs* menggunakan fungsi *get_all_logs* dari *model Logs_model.php*. • Memproses data untuk tampilan *logs.twig*.

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 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 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. Menambahkan *text field Display Name* pada halaman *Register (Open Public Registration)*.

Dari rancangan algoritma yang diterapkan, terdapat perubahan dan penambahan kode pada beberapa kelas. Beberapa kelas 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'
        );
...
```

Login.php

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

User_model.php

```
...
120     */
121     public function add_user($username, $email, $password, $role)
122     {
...
138     'username' => $username,
139     'email' => $email,
140     'password' => $this->password_hash->HashPassword($password),
...
176     $parts = preg_split('/\s+/', $line);
177     if (count($parts) != 4)
178             continue; //ignore lines that not contain 4 parts
...
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));
...
```

add_user.twig

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

add_user_result.twig

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

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>Usename: {{ 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>Usename: {{ 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*" disimpan pada tabel *shj_settings* pada kolom *shj_key* dengan nama *lock_student_display_name*. *lock_student_display_name* 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 diterapkan, terdapat penambahan kode pada beberapa kelas. Beberapa kelas 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 Menambahakan 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 ditambahkan pada tabel *shj_assignments* dengan nama *archived_assignment* yang menggunakan tipe data *tinyint*. *archived_assignment* 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 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* bernilai 1970-01-02 00:00:00, *text field Finish Time* bernilai 2038-01-18 00:00:00 dan *text field Extra Time* bernilai 0.
4. *Assignment* yang bersifat *Archived Assignment* tidak muncul pada kalendar halaman *dashboard*.

Dari rancangan algoritma yang diterapkan, terdapat penambahan kode pada beberapa kelas. Beberapa kelas 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');
...
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">
...
```

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

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* 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*

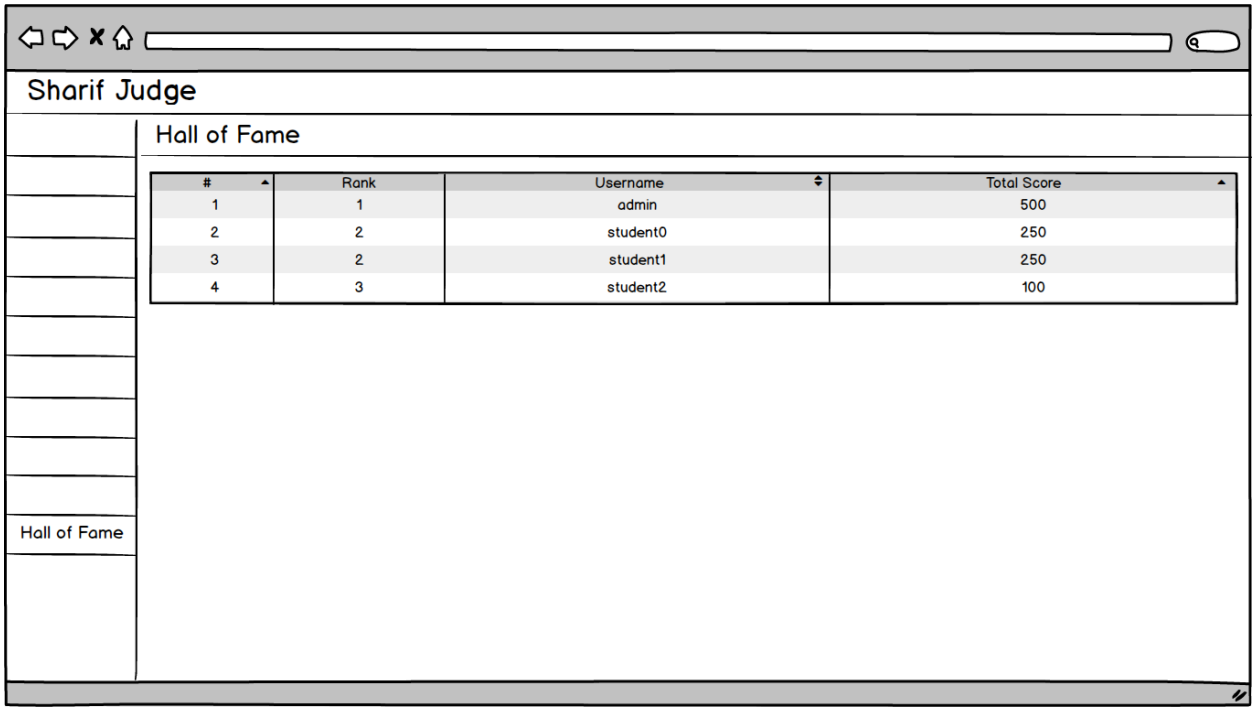
Nama *Method*	*get_all_final_submission*
Parameter *Input*	-
Parameter *Output*	semua entri nilai submissions yang telah dijumlahkan
Tabel yang berhubungan	*shj_submissions*
Deskripsi	Proses untuk mengembalikan semua entri nilai submission yang telah djumlahkan pada tabel *shj_submissions*
Algoritma	• Menjumlahkan seluruh nilai submission setiap peserta yang tidak bersifat "*Upload Only*". Nilai yang telah dijumlahkan akan disimpan sebagai total skor. • Mengurutkan total skor dari yang paling besar. • Mengembalikan seluruh entri yang telah dijumlahkan dalam bentuk *array*.

Tabel 4.7: Perincian fungsi *get_all_user_assignments*

Nama *Method*	*get_all_user_assignments*
Parameter *Input*	*$username*
Parameter *Output*	mengembalikan seluruh *details* dari *assignment* pengguna tertentu
Tabel yang berhubungan	*shj_submissions*
Deskripsi	Proses untuk mengembalikan *details assignment* pengguna tertentu. *Details* berisikan nama *assignment*, nama *problem* dan skor
Algoritma	• Menyimpan nama *assignment*, nama *problem* dan skor setiap *problem* dari sebuah *assignment* pengguna tertentu. • Mengembalikan *details* di atas dalam bentuk *array*.

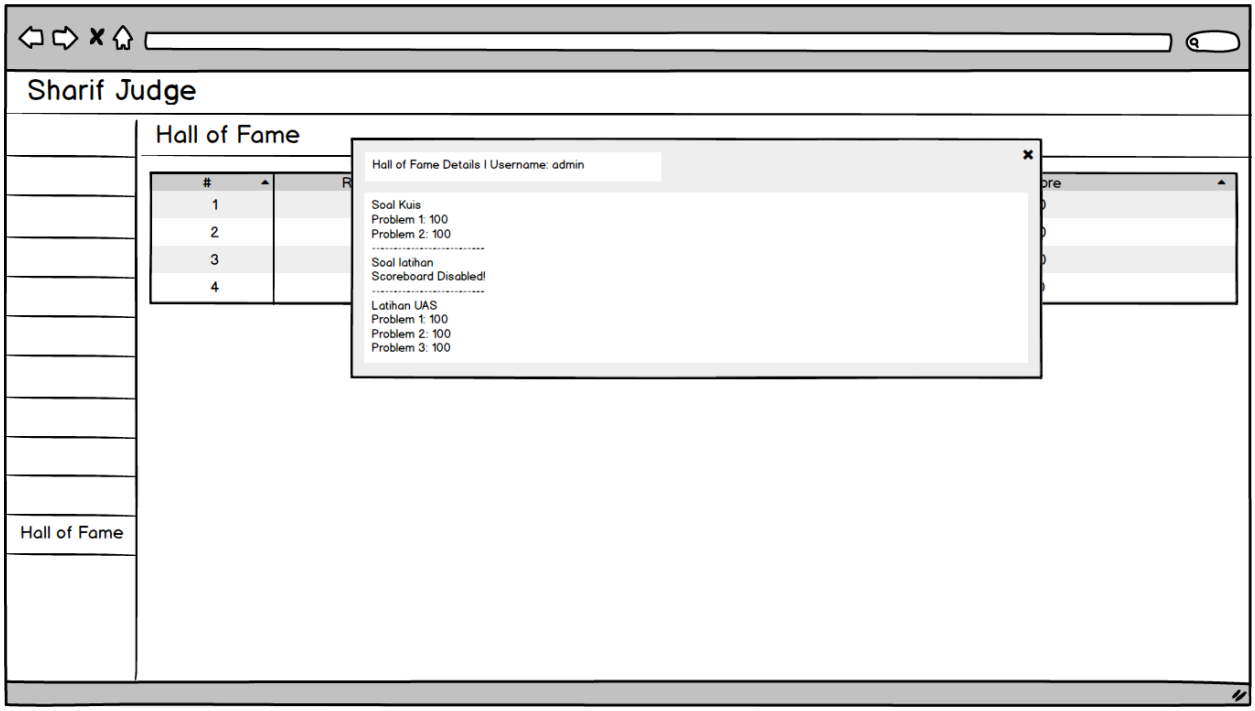
2. *View*

   View untuk halaman *Hall of Fame* bernama *halloffame.twig*. Menu halaman *Hall of Fame* terletak di bawah menu *Scoreboard*. Pada halaman ini juga berlaku sistem rangking dimana para peserta diurutkan bedasarkan total skor. Jika total skor yang dimiliki peserta memiliki nilai yang sama dengan peserta lainnya, maka peserta tersebut memiliki rangking 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* bernama *Halloffame.php*. Berikut adalah perincian fungsi yang terdapat dalam rancangan *controller Logs.php*.

Tabel 4.8: Perincian fungsi *consturct___*

Nama *Method*	*consturct___*
Parameter *Input*	-
Parameter *Output*	-
Tabel yang berhubungan	-
Deskripsi	membatasi pengguna yang dapat mengakses halaman *Hall of Fame*
Algoritma	• mengecek *session* pengguna yang akan mengakses halaman *Hall of Fame.* • Jika *session* tidak berstatus '*logged_in*, maka pengguna akan dialihkan ke halaman *login.* • Memuat *model Hof_model.php.*

Tabel 4.9: Perincian fungsi *index*

Nama *Method*	*index*
Parameter *Input*	-
Parameter *Output*	-
Tabel yang berhubungan	*shj_submissions*
Deskripsi	Proses untuk memuat seluruh entri *submissions* yang telah dijumlahkan pada halaman *halloffame.twig*
Algoritma	• Memuat data *Hall of Fame* menggunakan fungsi *get_all_final_submission* dari *model Hof_model.php.* • Memproses data untuk tampilan *halloffame.twig.*

Tabel 4.10: Perincian fungsi *hof_details*

Nama *Method*	*hof_details*
Parameter *Input*	-
Parameter *Output*	-
Tabel yang berhubungan	*shj_submissions*
Deskripsi	Proses untuk memuat details *submissions* pada halaman *halloffame.twig*
Algoritma	• Memuat *details Hall of Fame* peserta tertentu menggunakan fungsi *get_all_user_assignments* dari *model Hof_model.php.* • Memproses data untuk tampilan *details Hall of Fame* dari peserta tertentu pada halaman *halloffame.twig.*

Selain menambahkan kelas *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 ditambahkan setelah baris 479. Hasil penamabahan kode program yang terjadi di *shj_functions.js* dapat dilihat di Lampiran C

4.10 Integrasi *Login* ke *Server RADIUS*

Sharif Judge memerlukan *library* baru agar dapat mengintegrasikan *login* ke *server RADIUS*. *Library* yang 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 dapphp/radius*" 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 Dapphp/Radius* menggunakan *Composer*.
3. Membuat *file secrets.php* pada direktori `Sharif-Judge\application\config` untuk menyimpan configurasi alamat *server RADIUS*.
4. Mengintegrasikan *login* pengguna *Sharif Judge* ke *server RADIUS*.

Dari rancangan algoritma yang diterapkan, terdapat penambahan kode pada kelas *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     }
...
```

4.11 Branding Teknik Informatika

Branding Teknik Informatika dilakukan dengan cara mengubah seluruh logo dan ikon *Sharif Judge* menggunakan logo dan ikon Teknik Informatika. Berikut logo dan ikon Teknik Informatika yang digunakan

Gambar 4.4: Logo dan Ikon

Gambar 4.5: *Banner Sharif Judge*

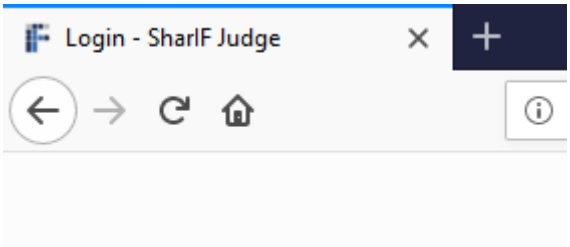
Agar dapat menggunakan logo dan ikon Teknik Informatika dibutuhkan penggantian beberapa file. Beberapa *file* tersebut antara lain *banner.png, favicon.ico* dan *logo_small.png* yang terdapat pada `Sharif-Judge\assets\images`. Penggantian beberapa *file* tersebut mencakup *banner* pada halaman *login*, icon pada *title bar* dan *top bar*.

Langkah-langkah yang dilakukan untuk proses branding Teknik Informatika adalah

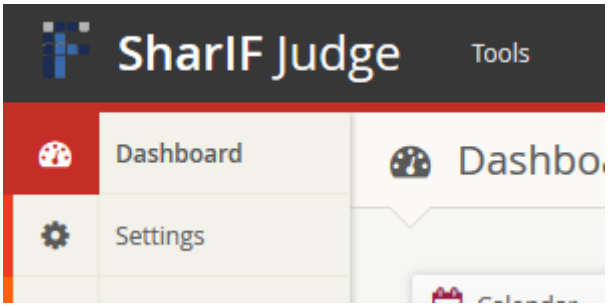
1. Menggunakan logo dan ikon Teknik Informatika pada *Sharif Judge*.
2. Mengubah nama *Sharif Judge* menjadi SharIF Judge.
3. Mengubah link dokumentasi *Github Sharif Judge* menjadi `https://github.com/ftisunpar/Sharif-Judge`.

Berikut tampilan *Sharif Judge* yang baru

Gambar 4.6: Halaman *Login Sharif Judge*



Gambar 4.7: Ikon *Sharif Judge* pada *Title Bar*



Gambar 4.8: Logo *Sharif Judge* pada *Top Bar*

4.12 Mengganti *Library PHPExcel* Menjadi *PhpSpreadsheet*

PhpSpreadsheet adalah *library* yang ditulis dalam PHP. *Library PhpSpreadsheet* menyediakan sekumpulan kelas yang memungkinkan pengguna untuk membaca dan menulis ke berbagai format *file spreadsheet*, seperti *Excel* dan *LibreOffice Calc* [8]. Menginstall *library PhpSpreadsheet* dapat dilakukan dengan menggunakan *Composer* dan menjalankan perintah "*composer require phpoffice/phpspreadsheet*".

Rancangan algoritma yang digunakan untuk mengganti *library PHPExcel* menjadi *PhpSpreadsheet* yaitu

1. Menginstall *Composer* pada perangkat lunak *Sharif Judge*.

2. Menambahkan *library PhpSpreadsheet* menggunakan *Composer*.
3. Mengubah fungsi yang menggunakan kelas *PHPExcel* menjadi menggunakan kelas *PhpSpreadsheet*

Dari rancangan algoritma yang diterapkan, terdapat perubahan kode pada kelas *controller Submissions.php* dan *controller Users.php*. Berikut beberapa baris potongan kode program *Submissions.php*

```
...
7       defined('BASEPATH') OR exit('No direct script access allowed');
8
9       class Users extends CI_Controller
...
144     // Load PHPExcel library
145     $this->load->library('phpexcel');
146
147     // Set document properties
148     $this->phpexcel->getProperties()->setCreator('Sharif Judge')
149             ->setLastModifiedBy('Sharif Judge')
150             ->setTitle('Sharif Judge Users')
151             ->setSubject('Sharif Judge Users')
152             ->setDescription('List of Sharif Judge users ('.$now.')');
153
154     // Name of the file sent to browser
155     $output_filename = 'sharifjudge_users';
156
157     // Set active sheet
158     $this->phpexcel->setActiveSheetIndex(0);
159     $sheet = $this->phpexcel->getActiveSheet();
...
172             'fill' => array(
173                     'type' => PHPExcel_Style_Fill::FILL_SOLID,
174                     'color' => array('rgb' => '173C45')
175     ),
...
207             'fill' => array(
208                     'type' => PHPExcel_Style_Fill::FILL_SOLID,
209                     'color' => array('rgb' => (($i%2)?'F0F0F0':'FAFAFA'))
210     )
...
215     // Set text align to center
216     $sheet->getStyle( $sheet->calculateWorksheetDimension() )
217             ->getAlignment()
218             ->setHorizontal(PHPExcel_Style_Alignment::HORIZONTAL_CENTER);
...
228     'outline' => array(
229             'style' => PHPExcel_Style_Border::BORDER_THIN,
230             'color' => array('rgb' => '444444'),
231     ),
...
249     header('Cache-Control: max-age=0');
```

```
250     $objWriter = PHPExcel_IOFactory::createWriter($this->phpexcel, ($ext===’
    xlsx’?’Excel2007’:’Excel5’));
251     $objWriter->save(’php://output’);
...
```

Users.php

```
...
7       defined(’BASEPATH’) OR exit(’No direct script access allowed’);
8
9       class Submissions extends CI_Controller
...
58      // Load PHPExcel library
59      $this->load->library(’phpexcel’);
60
61      // Set document properties
62      $this->phpexcel->getProperties()->setCreator(’Sharif Judge’)
63              ->setLastModifiedBy(’Sharif Judge’)
64              ->setTitle(’Sharif Judge Users’)
65              ->setSubject(’Sharif Judge Users’)
66              ->setDescription(’List of Sharif Judge users (’.$now.’)’);
67
68      // Name of the file sent to browser
69      $output_filename = ’judge_’.$view.’_submissions’;
70
71      // Set active sheet
72      $this->phpexcel->setActiveSheetIndex(0);
73      $sheet = $this->phpexcel->getActiveSheet();
...
99      ’fill’ => array(
100             ’type’ => PHPExcel_Style_Fill::FILL_SOLID,
101             ’color’ => array(’rgb’ => ’173C45’)
102     ),
...
184     ’fill’ => array(
185             ’type’ => PHPExcel_Style_Fill::FILL_SOLID,
186             ’color’ => array(’rgb’ => (($i%2)?’F0F0F0’:’FAFAFA’))
187     )
..
192     // Set text align to center
193     $sheet->getStyle( $sheet->calculateWorksheetDimension() )
194             ->getAlignment()
195             ->setHorizontal(PHPExcel_Style_Alignment::HORIZONTAL_CENTER);
...
205     ’outline’ => array(
206             ’style’ => PHPExcel_Style_Border::BORDER_THIN,
207             ’color’ => array(’rgb’ => ’444444’),
208     ),
...
211     header(’Cache-Control: max-age=0’);
212     $objWriter = PHPExcel_IOFactory::createWriter($this->phpexcel, ($ext===’
```

```
   xlsx'?'Excel2007':'Excel5'));
213     $objWriter->save('php://output');
...
```

Perubahan kode di controller *Submissions.php* untuk mengubah fungsi yang menggunakan kelas *PHPExcel* menjadi menggunakan kelas *PhpSpreadsheet*. Penambahan kode tersebut terjadi setelah baris 145, 148, 158, 159, 173, 208, 218, 229 dan 250. Berikut hasil penambahan kode program yang terjadi di controller *Submissions.php*

Submissions.php

```
...
7       defined('BASEPATH') OR exit('No direct script access allowed');
8
9       use PhpOffice\PhpSpreadsheet\Spreadsheet;
10      use PhpOffice\PhpSpreadsheet\IOFactory;
11      use PhpOffice\PhpSpreadsheet\Writer\Xlsx;
12      use PhpOffice\PhpSpreadsheet\Style\Fill;
13      use PhpOffice\PhpSpreadsheet\Style\Border;
14      use PhpOffice\PhpSpreadsheet\Style\Alignment;
15
16      class Users extends CI_Controller
...
151     // Load PHPExcel library
152     $phpspreedsheet = new Spreadsheet();
153
154     // Set document properties
155     $phpspreedsheet->getProperties()->setCreator('SharIF Judge')
156             ->setLastModifiedBy('Sharif Judge')
157             ->setTitle('Sharif Judge Users')
158             ->setSubject('Sharif Judge Users')
159             ->setDescription('List of Sharif Judge users ('.$now.')');
160
161     // Name of the file sent to browser
162     $output_filename = 'sharifjudge_users';
163
164     // Set active sheet
165     $phpspreedsheet->setActiveSheetIndex(0);
166     $sheet = $phpspreedsheet->getActiveSheet();
...
179             'fill' => array(
180                     'fillType' => Fill::FILL_SOLID,
181                     'color' => array('rgb' => '173C45')
182     ),
...
214             'fill' => array(
215                     'fillType' => Fill::FILL_SOLID,
216                     'color' => array('rgb' => (($i%2)?'F0F0F0':'FAFAFA'))
217     )
...
222     // Set text align to center
223     $sheet->getStyle( $sheet->calculateWorksheetDimension() )
```

```
224             ->getAlignment()
225             ->setHorizontal(Alignment::HORIZONTAL_CENTER);
...
235     'outline' => array(
236             'borderStyle' => Border::BORDER_THIN,
237             'color' => array('rgb' => '444444'),
238     ),
...
256     header('Cache-Control: max-age=0');
257     $objWriter = IOFactory::createWriter($phpspreedsheet, ucfirst($ext));
258     $objWriter->save('php://output');
...
```

Perubahan kode di *controller Users.php* untuk mengubah fungsi yang menggunakan kelas *PHPExcel* menjadi menggunakan kelas *PhpSpreadsheet*. Penambahan kode tersebut terjadi setelah baris 59, 72, 73, 100, 185, 195, 206 dan. Berikut hasil penambahan kode program yang terjadi di *controller Users.php*

Users.php

```
...
7       defined('BASEPATH') OR exit('No direct script access allowed');
8
9       use PhpOffice\PhpSpreadsheet\Spreadsheet;
10      use PhpOffice\PhpSpreadsheet\IOFactory;
11      use PhpOffice\PhpSpreadsheet\Writer\Xlsx;
12      use PhpOffice\PhpSpreadsheet\Style\Fill;
13      use PhpOffice\PhpSpreadsheet\Style\Border;
14      use PhpOffice\PhpSpreadsheet\Style\Alignment;
15      class Submissions extends CI_Controller
...
65      // Load PHPExcel library
66      $phpspreedsheet = new Spreadsheet();
67
68      // Set document properties
69      $phpspreedsheet->getProperties()->setCreator('SharIF Judge')
70              ->setLastModifiedBy('Sharif Judge')
71              ->setTitle('Sharif Judge Users')
72              ->setSubject('Sharif Judge Users')
73              ->setDescription('List of Sharif Judge users ('.$now.')');
74
75      // Name of the file sent to browser
76      $output_filename = 'sharifjudge_users';
77
78      // Set active sheet
79      $phpspreedsheet->setActiveSheetIndex(0);
80      $sheet = $phpspreedsheet->getActiveSheet();
...
106     'fill' => array(
107             'fillType' => Fill::FILL_SOLID,
108             'color' => array('rgb' => '173C45')
109     ),
```

```
...
191     'fill' => array(
192             'fillType' => Fill::FILL_SOLID,
193             'color' => array('rgb' => (($i%2)?'F0F0F0':'FAFAFA'))
194     )
..
199     // Set text align to center
200     $sheet->getStyle( $sheet->calculateWorksheetDimension() )
201             ->getAlignment()
202             ->setHorizontal(Alignment::HORIZONTAL_CENTER);
...
212     'outline' => array(
213             'borderStyle' => Border::BORDER_THIN,
214             'color' => array('rgb' => '444444'),
215     ),
...
218     header('Cache-Control: max-age=0');
219     $objWriter = IOFactory::createWriter($phpspreedsheet, ucfirst($ext));
220     $objWriter->save('php://output');
...
```

BAB 5

IMPLEMENTASI DAN PENGUJIAN

Bab ini membahas tentang implementasi dan pengujian perangkat lunak berdasarkan rancangan yang sudah dibuat. Ada dua jenis pengujian yang dilakukan, yaitu pengujian fungsional dan pengujian eksperimental. Bab ini juga membahas tentang lingkungan yang digunakan untuk pengujian perangkat lunak ini.

5.1 Lingkungan untuk Implementasi dan Pengujian

Berikut spesifikasi lingkungan perangkat keras dan perangkat lunak yang digunakan untuk melakukan implementasi dan pengujian perangkat lunak

Tabel 5.1: Lingkungan Perangkat Keras

Parameter	**Nilai**
Processor	*Intel Core i5 4200u*
Graphics Processing Unit (GPU)	*Intel HD Graphics HD4000* dan *Nvidia GeForce 840M*
Random Access Memory (RAM)	12.00GB DDR3
Storage	120GB *SSD* dan 1TB *Harddisk*

Tabel 5.2: Lingkungan Perangkat Lunak

Parameter	**Nilai**
Sistem Operasi	Windows 10 *10 Education 64-bit*
Bahasa Pemrograman	PHP, JavaScript, CSS dan HTML
Text Editor	*Atom*
Framework	*CodeIgniter*
Perangkat Lunak pendukung	*XAMPP Control Panel* v3.2.2 *Google Chrome Version* 65.0.3325.181 (Official Build) (64-bit) *Firefox Quantum* 59.0.2 (64-bit) *Microsoft Excel* 2016

5.2 Implementasi

Hasil implementasi dari rancangan perangkat lunak yang sudah dibuat, terdiri dari tiga bagian, yaitu

1. Kode Program
   Perubahan dan penambahan kode program untuk mengimplementasi kebutuhan *Sharif Judge*, ditulis dalam bahasa pemrograman PHP. Seluruh perubahan kode program telah dijabarkan dalam setiap sub bab pada bab 4. Kode program untuk halaman *Logs* dapat dilihat di Lampiran A. Kode program untuk halaman *Hall of Fame* dapat dilihat di Lampiran B.

2. Basis Data
   Terdapat penambahan tabel dalam mengimplementasi kebutuhan *Sharif Judge* pada sub bab 4.5. Tabel tersebut diberi nama *shj_logins*. Berikut struktur tabel *shj_logins*

Tabel 5.3: Struktur Tabel *shj_logins*

Atribut	**Tipe Data**	**Ukuran**	**Default**
login_id (primary key)	int	11	None
username	varchar	20	None
ip_address	varchar	15	None
timestamp	timestamp	11	current_timestamp
last_24h_login_id	int	11	null

Terdapat penambahan *key* dan *value* dalam mengimplementasi kebutuhan *Sharif Judge* pada sub bab 4.7. Pada tabel *shj_settings* ditambahkan *shj_key* dengan nama *lock_student_display_name* dan *shj_value* dengan nilai 0. Berikut struktur tabel *shj_settings*

Tabel 5.4: Struktur Tabel *shj_settings*

shj_key	**shj_value**
timezone	Asia/Jakarta
tester_path	pathC:/xampp/htdocs/tester
assignments_root	pathC:/xampp/htdocs/assignments
file_size_limit	50
output_size_limit	1024
queue_is_working	0
default_late_rule	/** Put coefficient (from 100) in variable $co...
enable_easysandbox	1
enable_c_shield	1
enable_cpp_shield	1
enable_py2_shield	1
enable_py3_shield	1
enable_java_policy	1
enable_log	1
submit_penalty	300
enable_registration	1
registration_code	0
mail_from	shj@example.com
mail_from_name	*Sharif Judge*
reset_password_mail	<p> Someone requested a password reset for your S...
add_user_mail	<p> Hello! You are registered in SharIF Judge at ...
moss_userid	
results_per_page_all	40
results_per_page_final	80
week_start	0
lock_student_display_name	1

Terdapat penambahan atribut dalam mengimplementasi kebutuhan *Sharif Judge* pada sub bab

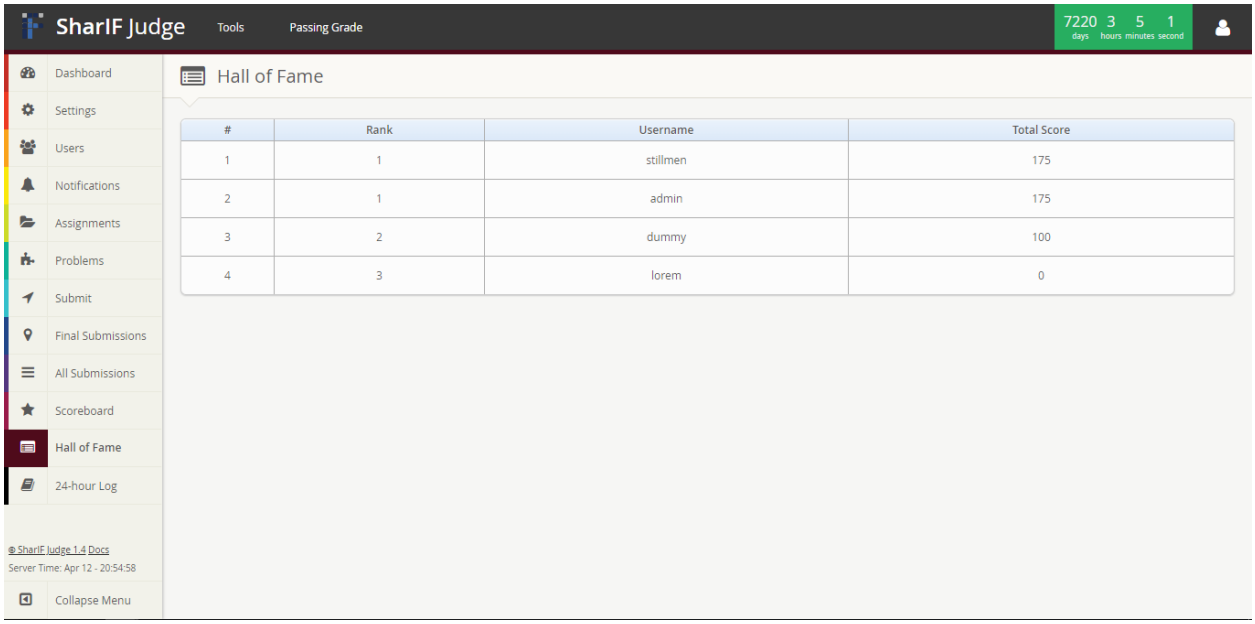
4.8. Pada tabel *shj_assignments* ditambahkan atribut baru dengan nama *archived_assignment*. Berikut struktur tabel *shj_assignments*

Tabel 5.5: Struktur Tabel *shj_assignments*

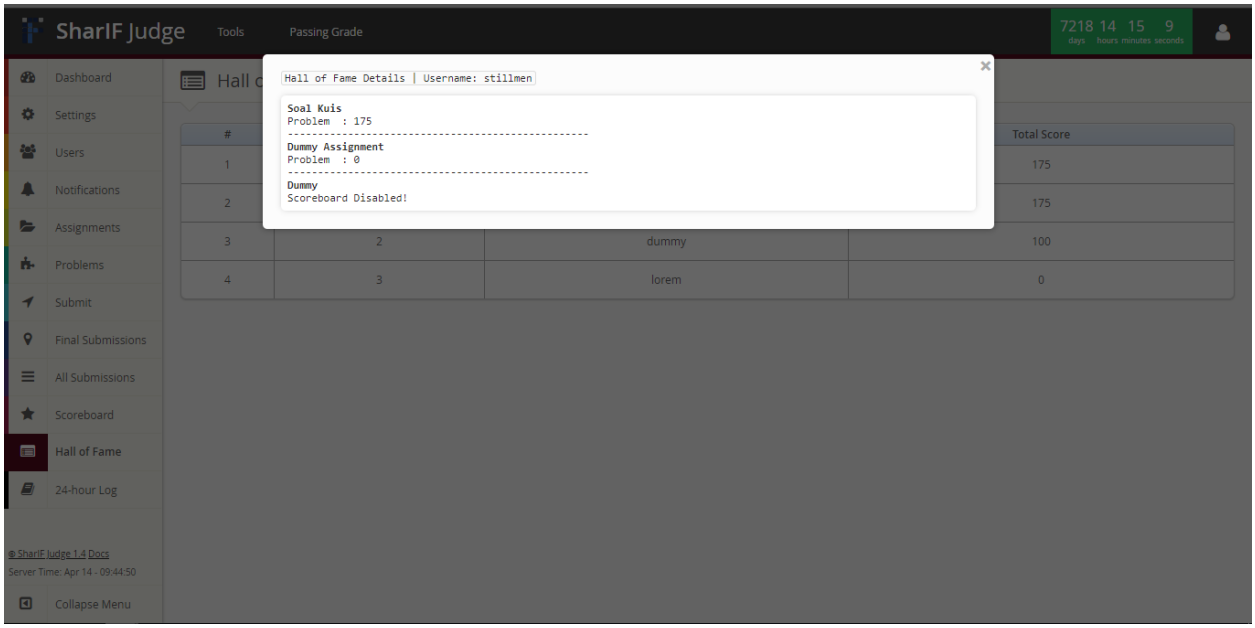
Atribut	**Tipe Data**	**Ukuran**	**Default**
id (primary key)	int	11	None
name	varchar	50	-
problems	smallint	4	None
total_submits	int	11	None
open	tinyint	1	None
scoreboard	tinyint	1	None
javaexceptions	tinyint	1	None
description	text	-	None
start_time	datetime	1	None
finish_time	datetime	1	None
extra_time	int	11	None
late_rule	text	-	None
participants	text	-	None
moss_update	varchar	30	None
archived_assignment	tinyint	1	None

3. Tampilan
   Tampilan untuk untuk pengembangan Sharif Judge ini dirancang bedasarkan rancangan tampilan yang sudah dibuat. Berikut beberapa tampilan halaman baru pada pengembangan Sharif Judge

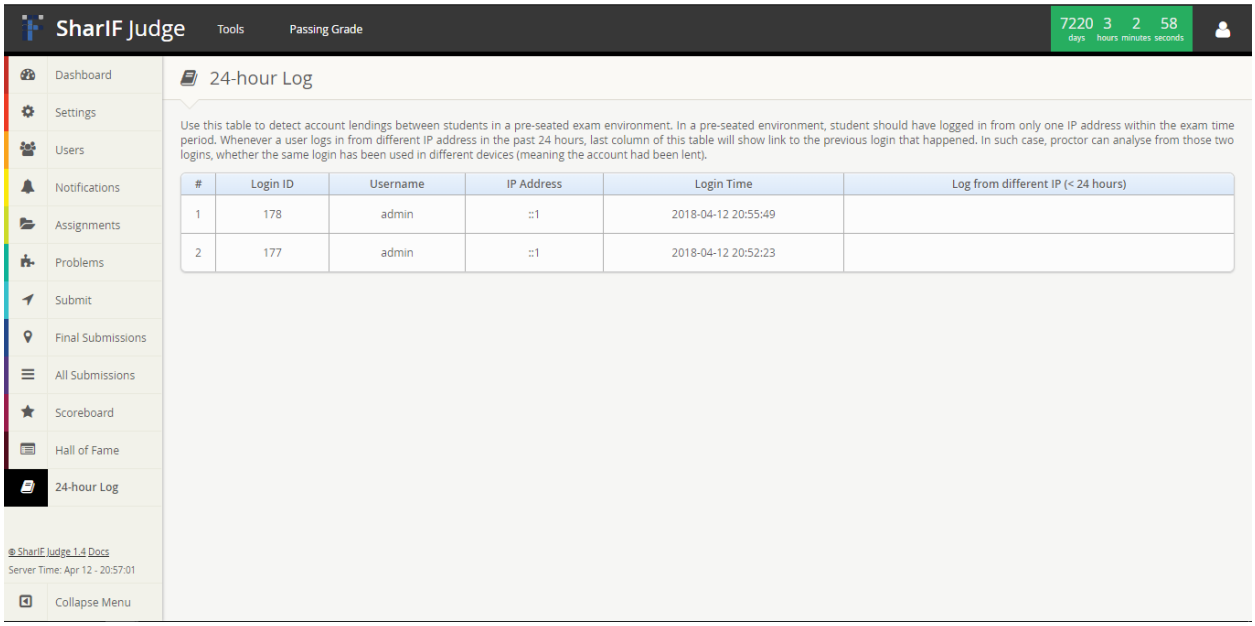
Gambar 5.1: Tampilan Halaman Login



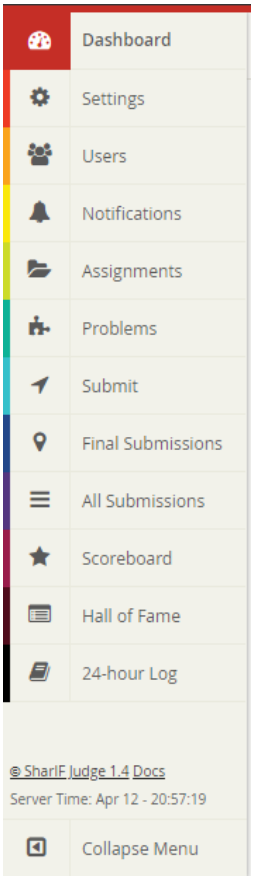
Gambar 5.2: Tampilan Halaman Hall of Fame



Gambar 5.3: Tampilan Detail dari Hall of Fame



Gambar 5.4: Tampilan Halaman Logs



Gambar 5.5: Tampilan Side Menu

5.3 Pengujian Fungsional

Pengujian fungsional bertujuan untuk memastikan setiap fitur baru pada SharIF Judge dapat berfungsi dengan baik.

5.3.1 Mengunduh Soal (deskripsi & PDF) yang Telah Dibatasi

Hasil yang diharapkan dari pengujian fitur ini adalah soal dapat dibatasi sesuai dengan ketentuan yang telah dirancang pada sub bab 4.3. Pengujian dimulai dengan membuat empat buah *assignment*.

Select	Name	Problems	Submissions	Coefficient	Start Time	Finish Time	Status	PDF	Actions
☑	Participant Hanya Username Tertentu	1 problem	1 submission	100 %	2018-04-14 13:23:11	2018-04-16 00:00:00	Open		
☐	Assignment Belum Dimulai	1 problem	3 submissions	100 %	2018-04-16 00:00:00	2018-04-21 00:00:00	Open		
☐	Assignment Telah Selesai	1 problem	2 submissions	Finished	2018-04-09 00:00:00	2018-04-13 00:00:00	Open		
☐	Closed Assignment	5 problems	4 submissions	Finished	2018-04-14 13:23:31	2018-04-14 13:23:36	Close		

Gambar 5.6: Empat Buah *Assignment* yang Dibuat

Assignment dengan nama "*Participant* Hanya *Username* Tertentu", merupakan *assignment* yang dikhususkan untuk peserta tertentu. *Assignment* dengan nama "*Assignment* Belum Mulai", merupakan *assignment* yang belum dimulai. *Assignment* dengan nama "*Assignment* Telah Selesai", merupakan *assignment* yang waktu pengumpulannya telah habis. *Assignment* dengan nama "*Closed Assignment*", merupakan *assignment* yang memiliki status *close*.

Jika peserta yang tidak terdaftar sebagai *participant*, mencoba untuk mengunduh *assignment* dengan nama "*Participant* Hanya *Username* Tertentu", maka muncul pesan *error "You are not registered for submitting."* seperti Gambar 5.7

An Error Was Encountered

You are not registered for submitting.

Gambar 5.7: Pesan *Error "You are not registered for submitting."*

Jika peserta mencoba untuk mengunduh *assignment* dengan nama "*Assignment* Belum Mulai", maka muncul pesan *error "Selected assignment has not started."* seperti Gambar 5.8

An Error Was Encountered

Selected assignment has not started.

Gambar 5.8: Pesan *Error "Selected assignment has not started."*

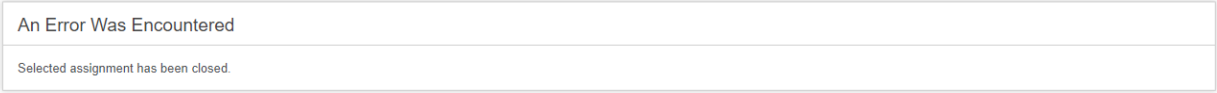
Jika peserta mencoba untuk mengunduh *assignment* dengan nama "*Assignment* Telah Selesai", maka muncul pesan *error "Selected assignment has finished."* seperti Gambar 5.9

An Error Was Encountered

Selected assignment has finished.

Gambar 5.9: Pesan *Error "Selected assignment has finished."*

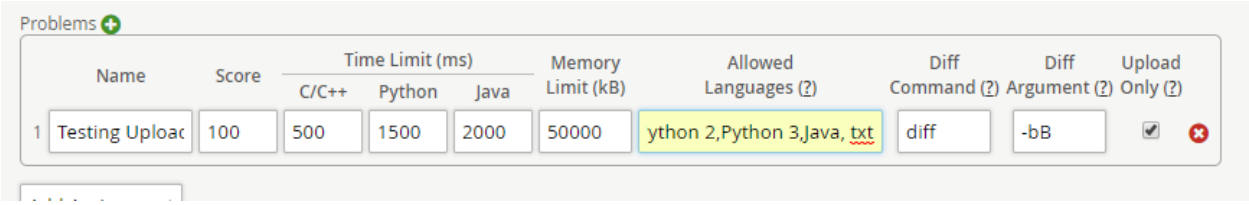
Jika peserta mencoba untuk mengunduh *assignment* dengan nama "*Closed Assignment*", maka muncul pesan *error "Selected assignment has been closed."* seperti Gambar 5.10



Gambar 5.10: Pesan *Error "Selected assignment has been closed."*

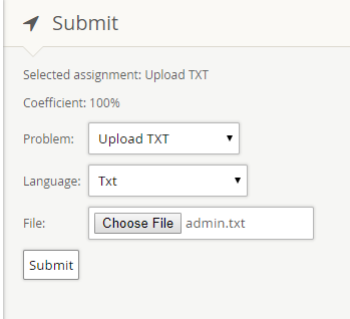
5.3.2 Membuat *Assignment* yang Menerima *File* dengan Ekstensi TXT

Hasil yang diharapkan dari pengujian fitur ini adalah *assignment* yang dibuat bisa menerima *file* dengan ekstensi TXT dan pengguna bisa mengumpulkan file dengan ekstensi TXT ke *assignment* tersebut. Pengujian dimulai dengan membuat *assignment* yang bersifat "*Upload Only*" dan menambahkan '*txt*' pada *text field Allowed Language*.

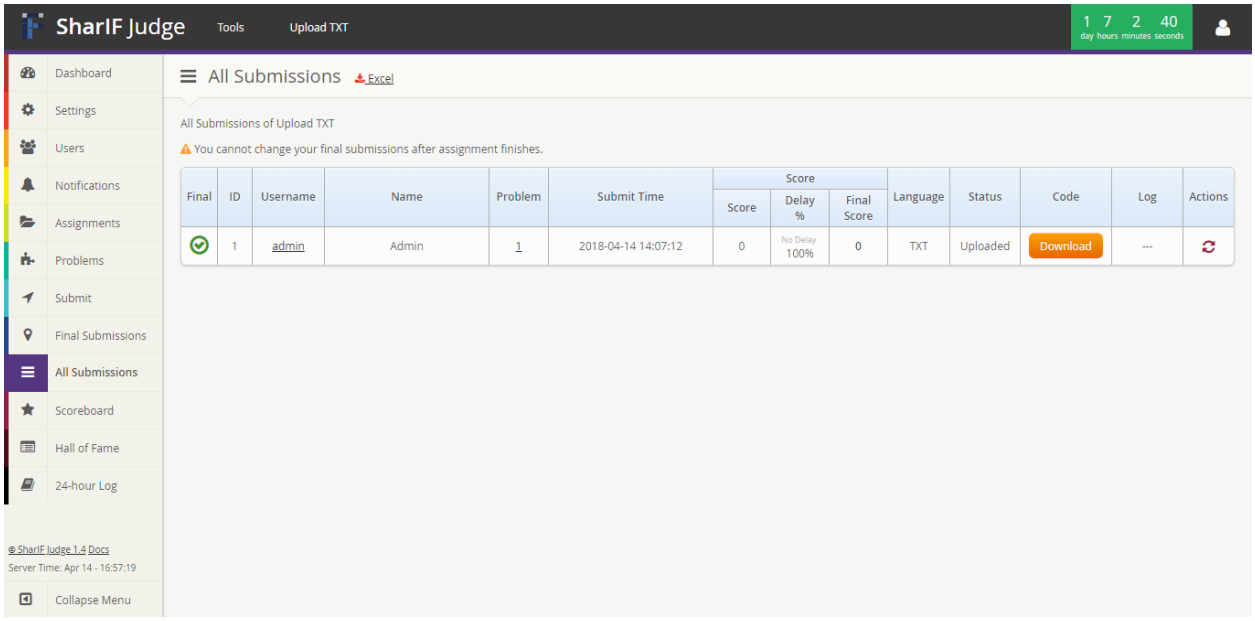


Gambar 5.11: Pembuatan *Assignment* Upload TXT

Setelah *assignment* berhasil dibuat, selanjutnya dicoba untuk mengumpulkan *file* dengan ekstensi TXT pada halaman *Submit* seperti Gambar 5.12. Jika berhasil mengumpulkan *file* TXT, maka pengguna langsung diarahkan ke halaman *All Submission* seperti Gambar 5.13

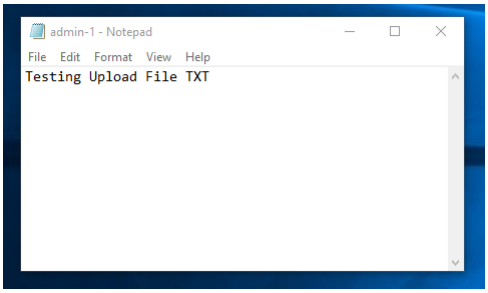


Gambar 5.12: *Submit File* TXT

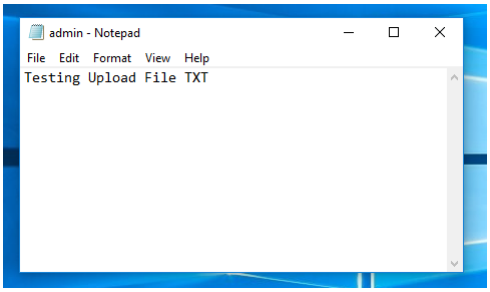


Gambar 5.13: Halaman *All Submission* setelah Mengumpulkan *File* TXT

Pengujian dilanjutkan dengan mengunduh dan mencocokan isi dari *file* TXT yang baru saja dikumpulkan. Jika isi dari *file* TXT hasil unduh sama dengan *file* TXT utama, maka fitur ini dapat berjalan dengan baik. Gambar 5.14 merupakan isi dari *file* TXT yang dikumpulkan dan Gambar 5.15 isi dari *file* TXT utama.



Gambar 5.14: *File* TXT Hasil Unduh

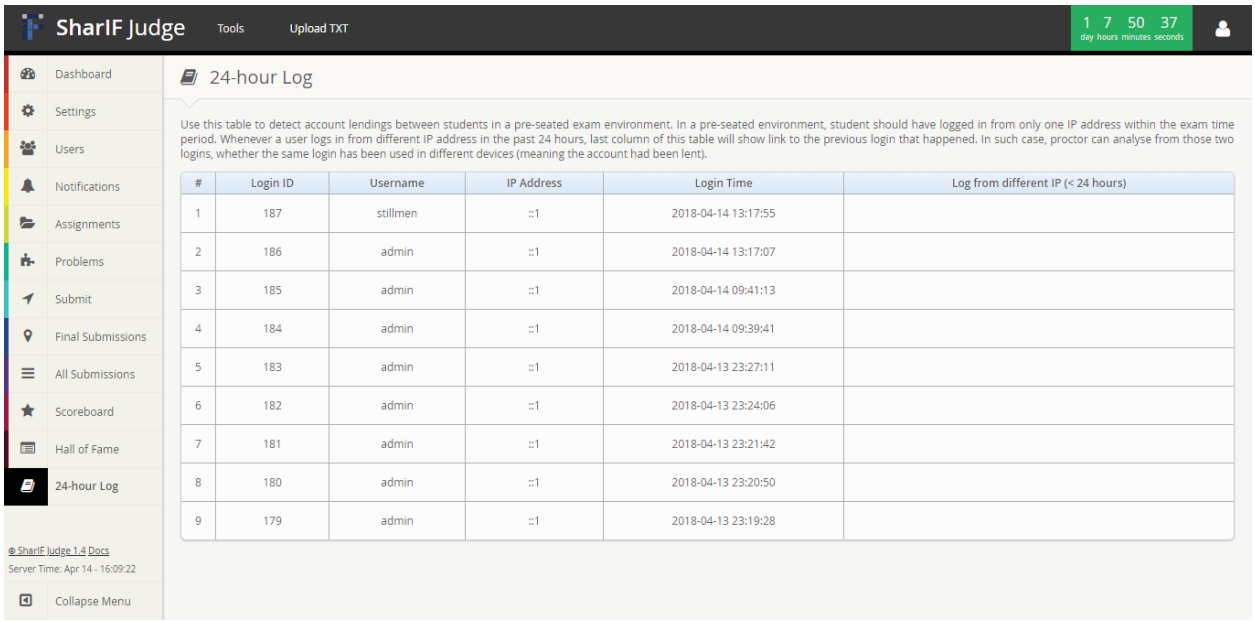


Gambar 5.15: File TXT Utama

5.3.3 Mengakses Halaman *24-hour Logs*

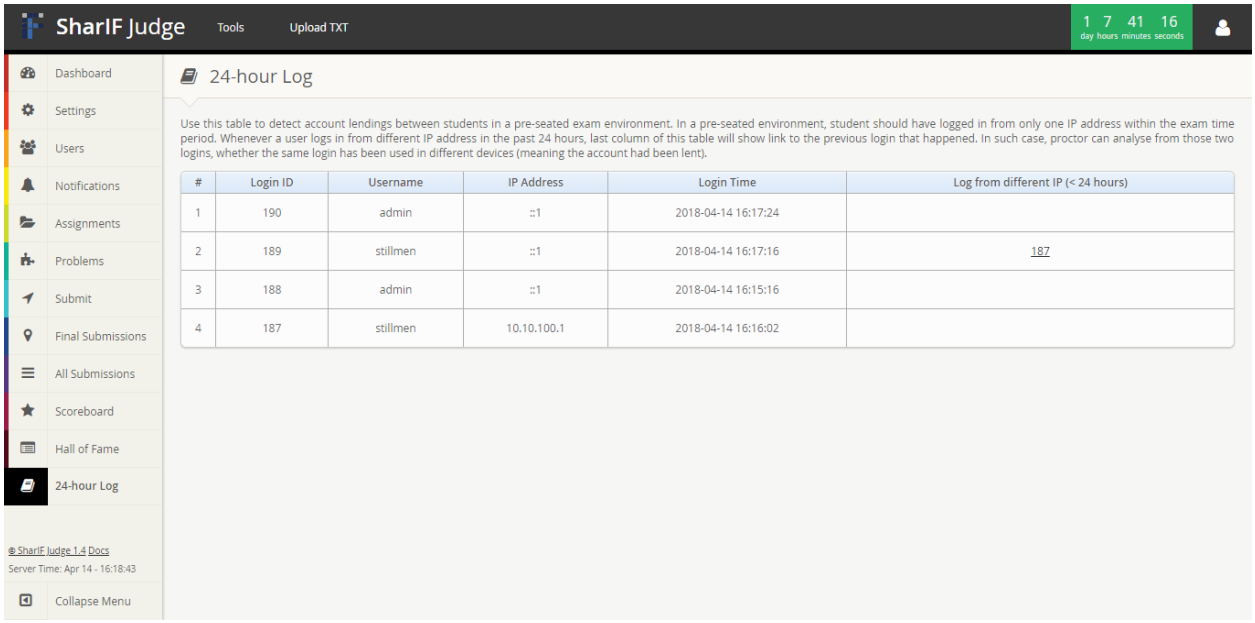
Hasil yang diharapkan dari pengujian fitur ini adalah halaman *24-hour Logs* dapat menampilkan seluruh aktivitas *login* dari pengguna. Pengujian dimulai dari *login* menggunakan *username* dengan

role admin. Setelah berhasil *login*, akses halaman *24-hour Logs* yang terletak di paling bawah *side menu*. Halaman *24-hour Logs* tampil seperti Gambar 5.16



Gambar 5.16: Halaman *24-hour Logs* yang Tampil

Pengujian dilanjutkan dengan mencoba *login* menggunakan *username* yang sama namun menggunakan *ip address* yang berbeda. Hasil yang diharapkan adalah halaman *24-hour Logs* dapat mencatat dan menampilkan Login ID yang menggunakan *ip address berbeda*.

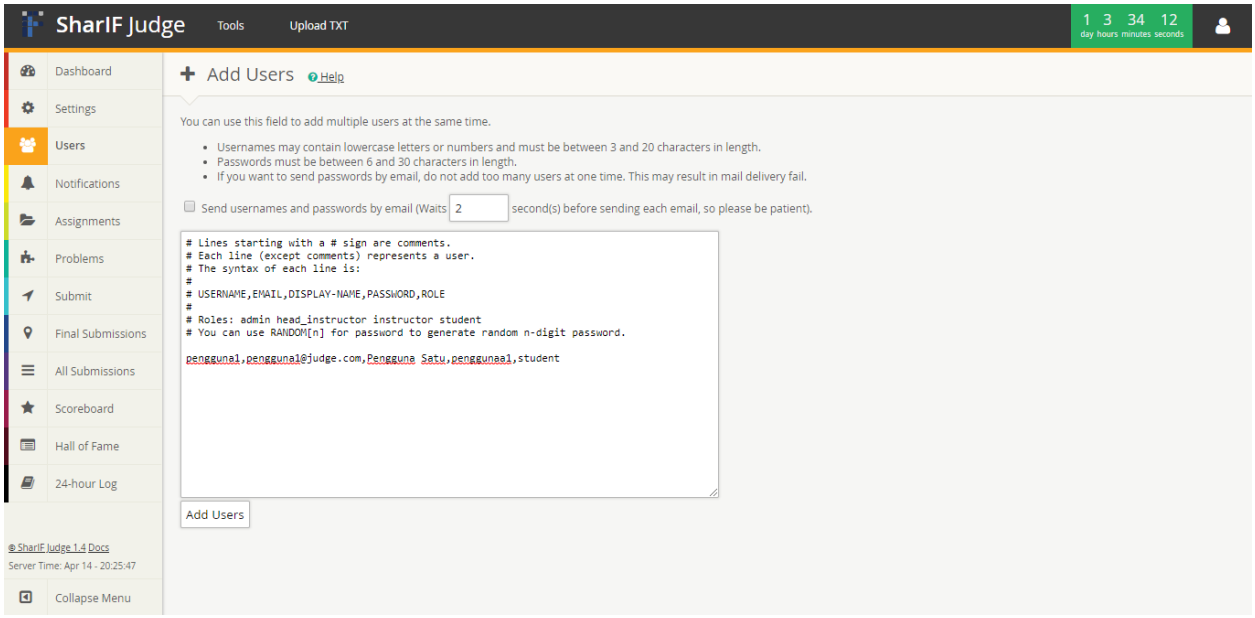


Gambar 5.17: Halaman *24-hour Logs* Mencatat Aktivitas *Login* Pengguna

Terlihat pada Gambar 5.17, *username* stillmen pertama kali *login* menggunakan *ip address* 10.10.100.1 (baris 4). Setelah mencoba untuk *login* menggunakan *ip address* yang berbeda, halaman *24-hour Logs* dapat mencatat dan menampilkan *Login ID* dari *username* stillmen (baris 2).

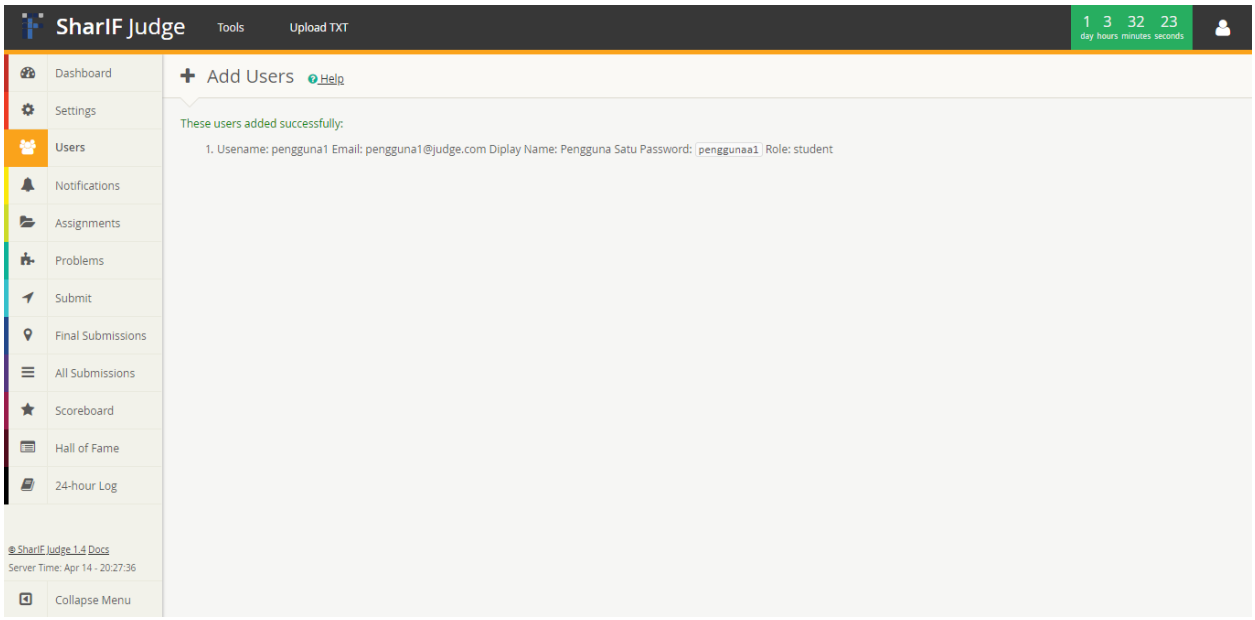
5.3.4 Mendaftarkan Peserta Menggunakan Tambahan Parameter "*Display Name*"

Hasil yang diharapkan dari pengujian fitur ini adalah peserta yang didaftarkan langsung memiliki *Display Name*. Pengujian dimulai dengan menekan tombol *Add Users* pada halaman *Users*. Gunakan parameter "pengguna1,pengguna1@judge.com,Pengguna Satu,penggunaa1,student" untuk menambah peserta baru seperti Gambar 5.18



Gambar 5.18: Halaman *Add User*

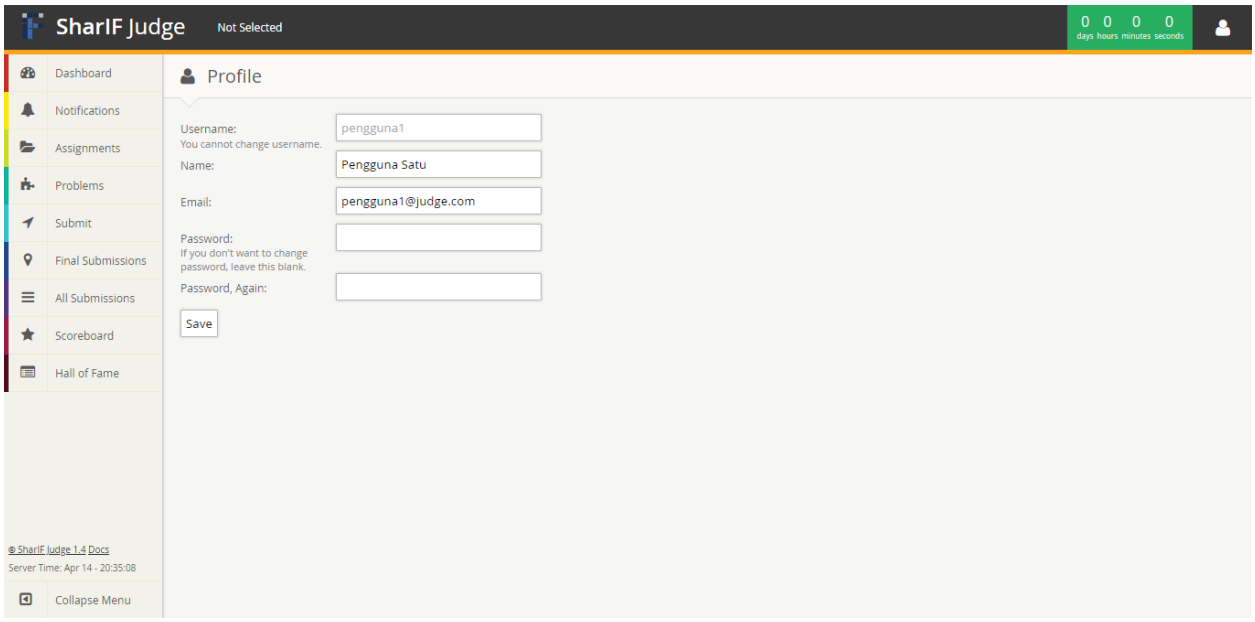
Jika pengguna baru berhasil ditambahkan, maka akan muncul pesan bahwa tersebut berhasil didaftarkan.



Gambar 5.19: Pengguna Berhasil Didaftarkan

Pengujian dilanjutkan dengan *login* menggunakan *user* yang baru didaftarkan dan mengecek *Display Name*. Gambar 5.20 menunjukan bahwa *Display Name* yang muncul sesuai dengan parameter

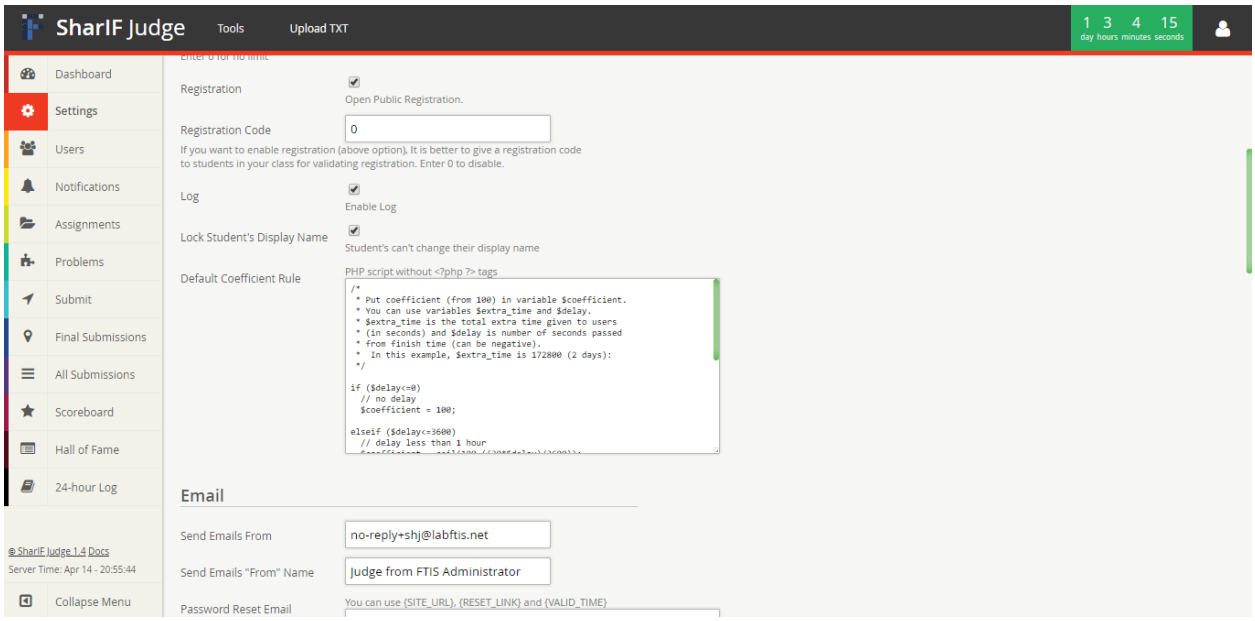
Display Name yang telah dimasukan sebelumnya.



Gambar 5.20: *Display Name* yang Tampil Sesuai dengan Parameter yang Dimasukan

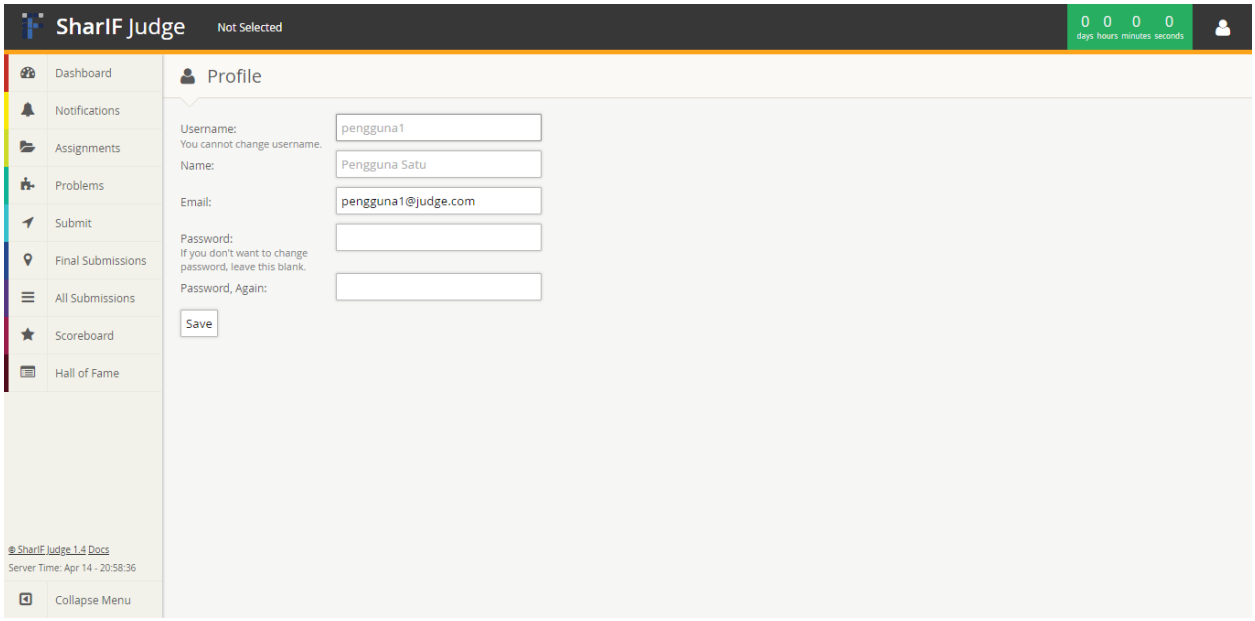
5.3.5 *Disable Display Name* Peserta Menggunakan Fitur "*Lock Student's Display Name*"

Hasil yang diharapkan dari pengujian fitur ini adalah *text field Display Name* pada halaman *Profile* menjadi *disable*, sehingga para peserta tidak dapat mengubahnya. Pengujian dimulai dengan mengaktifkan fitur *Lock Student's Display Name* pada halaman *Settings* seperti Gambar 5.21.



Gambar 5.21: Mengaktifkan Fitur *Lock Student's Display Name*

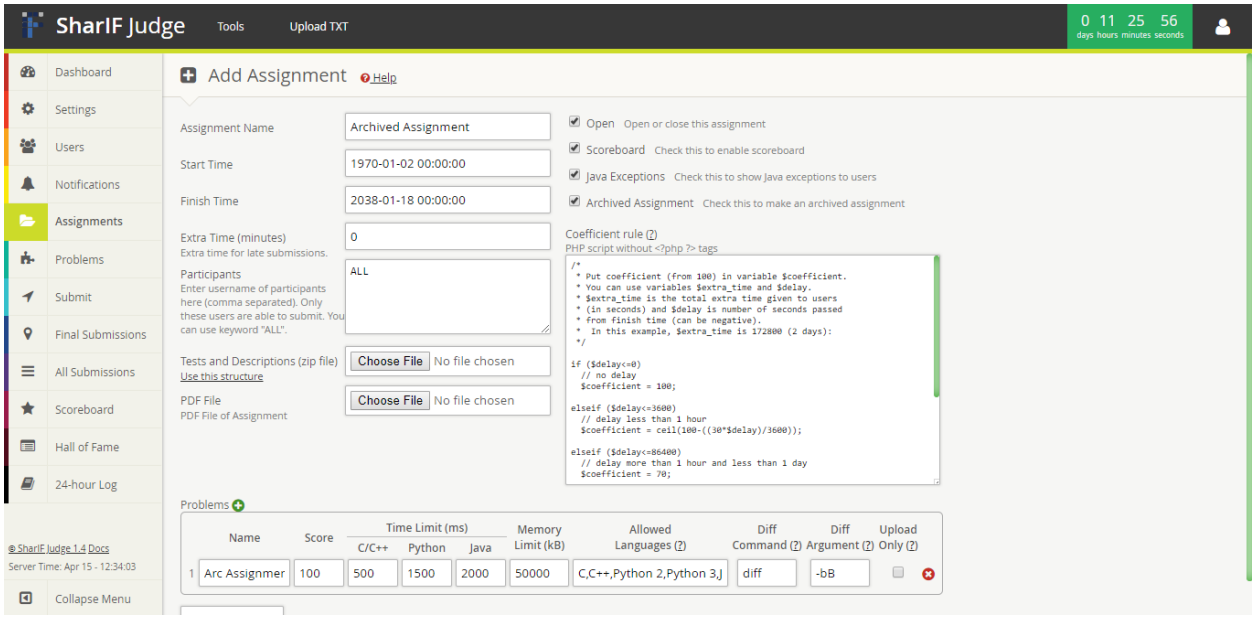
Pengujian dilanjutkan dengan mengecek *text field Display Name* pada halaman *Profile*. Gambar 5.22 menunjukan bahwa *text field Display Name* telah disable.



Gambar 5.22: *Text Field Display Name* Menjadi *Disable*

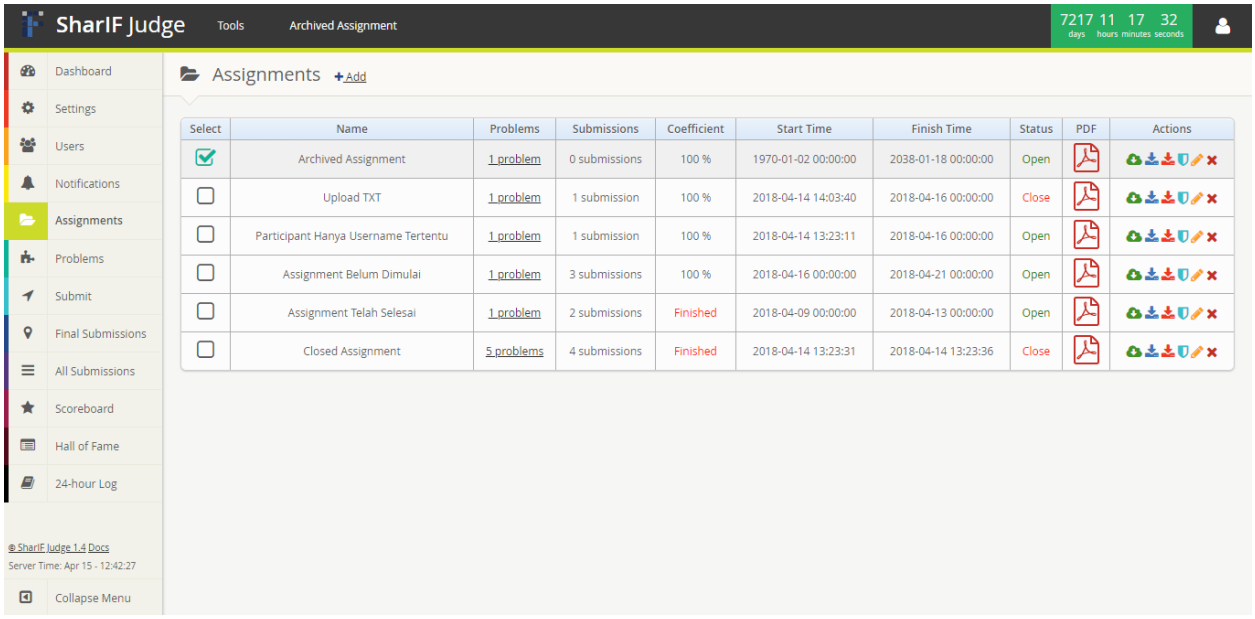
5.3.6 Menambahkan *Assignment* yang Mengaktifkan Fitur "*Archived Assignment*"

Hasil yang diharapkan dari pengujian fitur ini adalah *assignment* yang diibuat memiliki batas waktu pengumpulan sampai tanggal 18 Januari 2038. Pengujian dimulai dengan membuat *assignment* dan mengaktifkan fitur *Archived Assignment* pada halaman *Add Assignment* seperti Gambar 5.23.



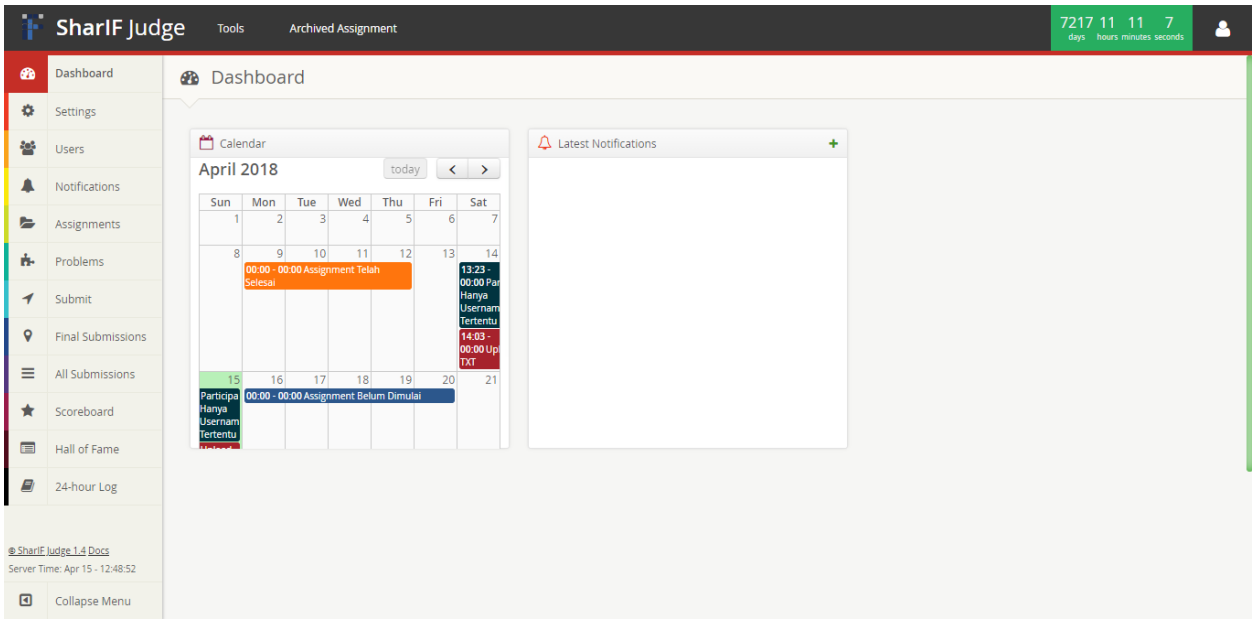
Gambar 5.23: Mengaktifkan Fitur *Archived Assignment*

Gambar 5.24 menunjukan bahwa *assignment* yang baru dibuat (baris 1) memiliki nilai *Finish Time* "2038-01-18 00:00:00" yang artinya *assignment* tersebut dapat dikumpulkan sampai tanggal 18 Januari 2038.



Gambar 5.24: *Finish Time* dengan Nilai "2038-01-18 00:00:00"

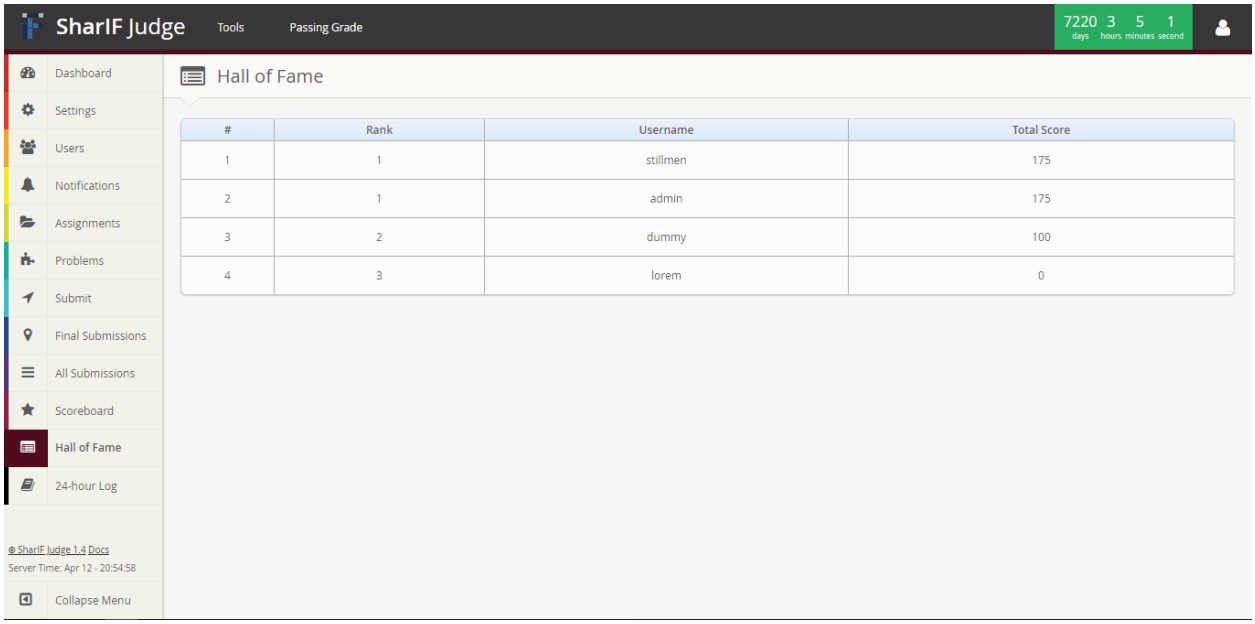
Pengujian dilanjutkan dengan mengecek kalendar pada halaaman *Dashboard*. *Assignment* yang mengaktifkan fitur *Archived Assignment* tidak akan muncul pada kalendar seperti Gambar 5.25



Gambar 5.25: *Archived Assignment* Tidak Muncul pada Kalendar

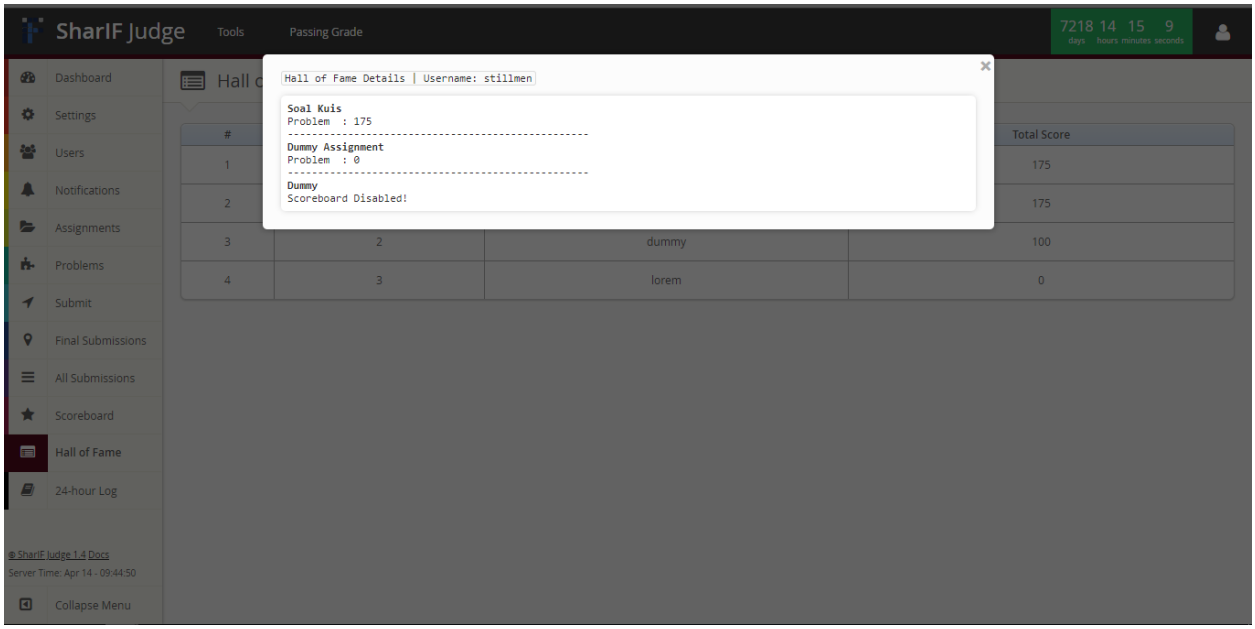
5.3.7 Mengakses Halaman *Hall of Fame*

Hasil yang diharapkan dari pengujian fitur ini adalah halaman *Hall of Fame* dapat menampilkan total nilai semua pengguna dari setiap *assignment* yang telah dikumpulkan. Pengujian dimulai dengan mengakses halaman *Hall of Fame* yang terletak di bawah *side menu Scoreboard*. Halaman *Hall of Fame* tampil seperti Gambar 5.26



Gambar 5.26: Halaman *Hall of Fame*

Pengujian dilanjutkan dengan mengklik baris pada tabel untuk melihat *details* nilai pengguna. Gambar 5.27 menunjukan *details* nilai yang diperoleh pengguna dengan *username stillmen*.



Gambar 5.27: *Details* Nilai yang Diperoleh

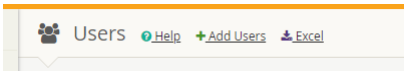
5.3.8 *Login* Menggunakan *Username* yang Terdaftar pada *Server RADIUS*

Hasil yang diharapkan dari pengujian fitur ini adalah pengguna dapat *login* ke *SharIF Judge* menggunakan *username* yang terdaftar pada *server RADIUS*. *Username* yang digunakan dalam pengujian fitur ini adalah xxxxx. Pengujian dimulai dengan *login* menggunakan *username* xxxxxx ke perangkat lunak *SharIF Judge* yang telah diinstal pada *server* lab FTIS UNPAR (selanjutnya digunakan untuk pengujian eksperimental). *SharIF Judge* yang telah diinstal, dapat diakses pada *URL* `http://asd.labftis.net/`. Gambar 5.28 menunjukan bahwa *username* xxxxx berhasil *login* ke *SharIF Judge*.

Gambar 5.28: Berhasil *Login* ke *SharIF Judge*

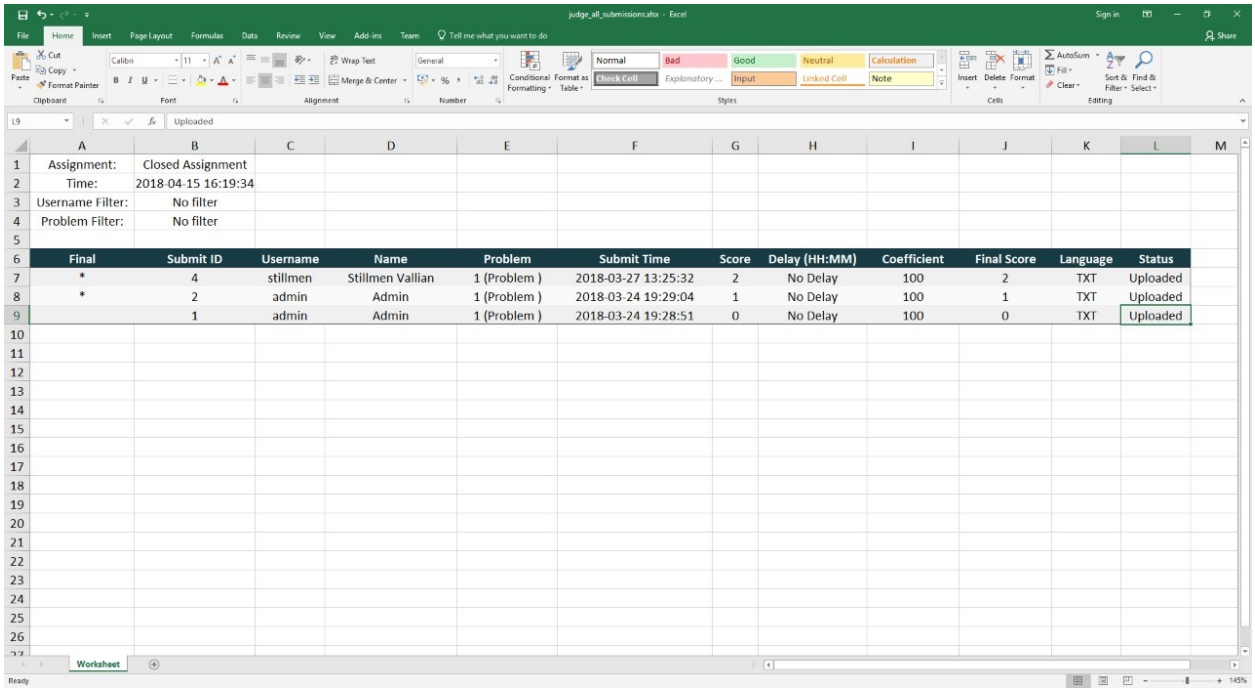
5.3.9 Mengunduh *File Excel* dari Halaman *All Submission* dan *Users*

Hasil yang diharapkan dari pengujian fitur ini adalah *file excel* berhasil diunduh dari halaman *All Submission* dan *Users*. *File excel* yang diunduh pada halaman *All Submission* berisikan daftar seluruh jawaban pengguna *SharIF Judge*. *File excel* yang diunduh pada halaman *Users* berisikan dafttar seluruh pengguna *SharIF Judge*. Pengujian dimulai dengan mengklik tombol unduh *excel* yang terletak di atas halaman. Gambar 5.29 menunjukan letak tombol unduh *excel* pada halaman *Users*



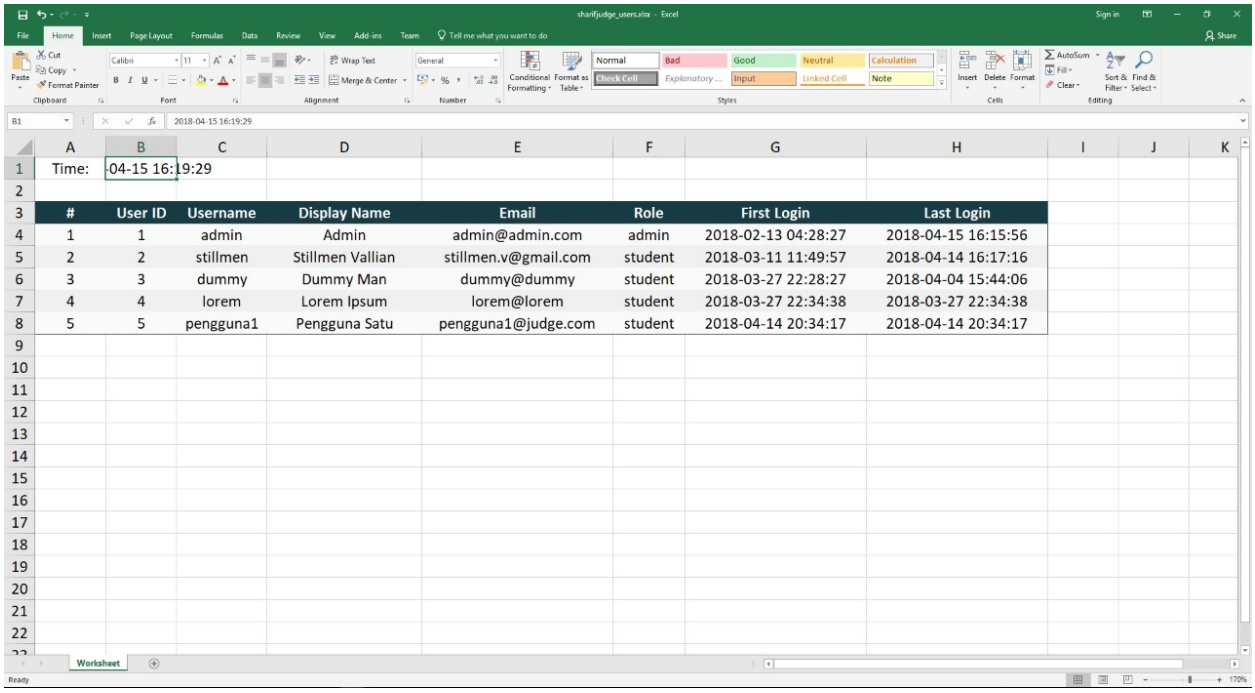
Gambar 5.29: Letak Tombol Unduh *Excel* pada Halaman *Users*

Excel yang diunduh dari halaman *All Submission* bernama *judge_all_submissions.xlsx* sedangkan halaman *Users* bernama *sharifjudge_users.xlsx*. *File excel* tersebut bisa dibuka menggunakan *Microsoft Excel 2016*. Gambar 5.30 menunjukan isi dari *excel judge_all_submissions.xlsx* dan Gambar 5.31 menunjukan isi dari *excel sharifjudge_users.xlsx*.



	A	B	C	D	E	F	G	H	I	J	K	L
1	Assignment:	Closed Assignment										
2	Time:	2018-04-15 16:19:34										
3	Username Filter:	No filter										
4	Problem Filter:	No filter										
5												
6	Final	Submit ID	Username	Name	Problem	Submit Time	Score	Delay (HH:MM)	Coefficient	Final Score	Language	Status
7	*	4	stillmen	Stillmen Vallian	1 (Problem)	2018-03-27 13:25:32	2	No Delay	100	2	TXT	Uploaded
8	*	2	admin	Admin	1 (Problem)	2018-03-24 19:29:04	1	No Delay	100	1	TXT	Uploaded
9		1	admin	Admin	1 (Problem)	2018-03-24 19:28:51	0	No Delay	100	0	TXT	Uploaded

Gambar 5.30: Isi *Excel judge_all_submissions.xlsx*



Time: 04-15 16:19:29

#	User ID	Username	Display Name	Email	Role	First Login	Last Login
1	1	admin	Admin	admin@admin.com	admin	2018-02-13 04:28:27	2018-04-15 16:15:56
2	2	stillmen	Stillmen Vallian	stillmen.v@gmail.com	student	2018-03-11 11:49:57	2018-04-14 16:17:16
3	3	dummy	Dummy Man	dummy@dummy	student	2018-03-27 22:28:27	2018-04-04 15:44:06
4	4	lorem	Lorem Ipsum	lorem@lorem	student	2018-03-27 22:34:38	2018-03-27 22:34:38
5	5	pengguna1	Pengguna Satu	pengguna1@judge.com	student	2018-04-14 20:34:17	2018-04-14 20:34:17

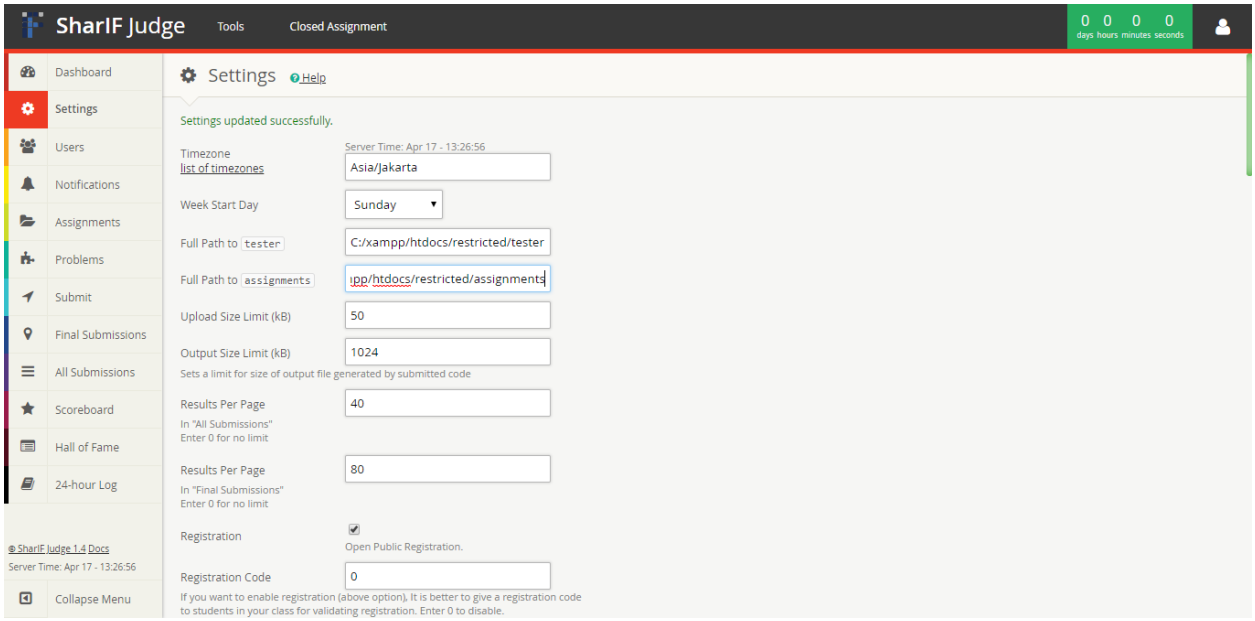
Gambar 5.31: Isi *Excel sharifjudge_users.xlsx*

5.4 Pengujian Eksperimental

Pengujian eksperimental dilakukan dengan cara menginstal perangkat lunak *SharIF Judge* pada server lab FTIS UNPAR. SharIF Judge digunakan untuk mata kuliah Algoritma & Struktur Data (ASD) semester genap 2017/2018. Selama pengujian berlangsung, terdapat beberapa persoalan yang muncul. Persoalan yang muncul dicatat ke dalam isu *repository SharIF Judge* di *GitHub*. Semua persoalan dapat dilihat di `https://github.com/ayenz/Sharif-Judge/issues`.

5.4.1 *Remove the Assignments Folder*

Isu dengan kode unik #4 meminta agar *folder assignments* untuk dihapus. Hal ini bertujuan agar direktori *URL* `/assignments` dapat diakses. *Folder assignments* merupakan *folder default* untuk menyimpan *assignment*. *Folder default* penyimpanan *assignment* tersebut harus diubah terlebih dahulu. Untuk mengubah *folder default* penyimpanan, diperlukan perubahan nilai pada halaman *Settings* dan kode pada kelas *Controller Install.php*. Selain mengubah *folder default* penyimpanan *assignment*, perubahan juga terjadi pada *folder default tester* dan *default timezone* yang bertujuan untuk merapihkan struktur direktori.



Gambar 5.32: Perubahan yang Terjadi pada Halaman *Settings*

Gambar 5.32 menunjukan perubahan pada halaman *Settings* terjadi pada *text field* "*Timezone*", "*Full Path to tester*" dan "*Full Path to assignments*".

- Nilai yang digunakan untuk *text field "Full Path to assignments"* adalah `C:/xampp/htdocs/restricted/assignments`. Nilai tersebut memiliki arti bahwa *folder default* penyimpanan *assignment* diletakan pada direktori `C:/xampp/htdocs/restricted/assignments`.
- Nilai yang digunakan untuk *text field "Full Path to tester"* adalah `C:/xampp/htdocs/restricted/tester`. Nilai tersebut memiliki arti bahwa *folder default tester* diletakan pada direktori `C:/xampp/htdocs/restricted/tester`.
- Nilai yang digunakan untuk *text field "Timezone"* adalah Asia/Jakarta. Nilai tersebut memiliki arti bahwa *default timezone* yang digunakan pada *SharIF Judge* mengikuti *timezone* wilayah Asia/Jakarta.

Perubahan kode pada kelas *Controller Install.php* bertujuan agar penginstalan *SharIF Judge* langsung menggunakan *default* nilai seperti di atas. Berikut beberapa baris potongan kode program

```
...
196     // insert default settings to table 'settings'
197     $result = $this->db->insert_batch('settings', array(
198     array('shj_key' => 'timezone',              'shj_value' => 'Asia/Tehran
    '),
199     array('shj_key' => 'tester_path',           'shj_value' => '/home/shj/
    tester'),
200     array('shj_key' => 'assignments_root',      'shj_value' => '/home/shj/
    assignments'),
...
```

Perubahan kode terjadi pada baris 198, 199, 200. Berikut hasil perubahan kode program yang terjadi di *Install.php*

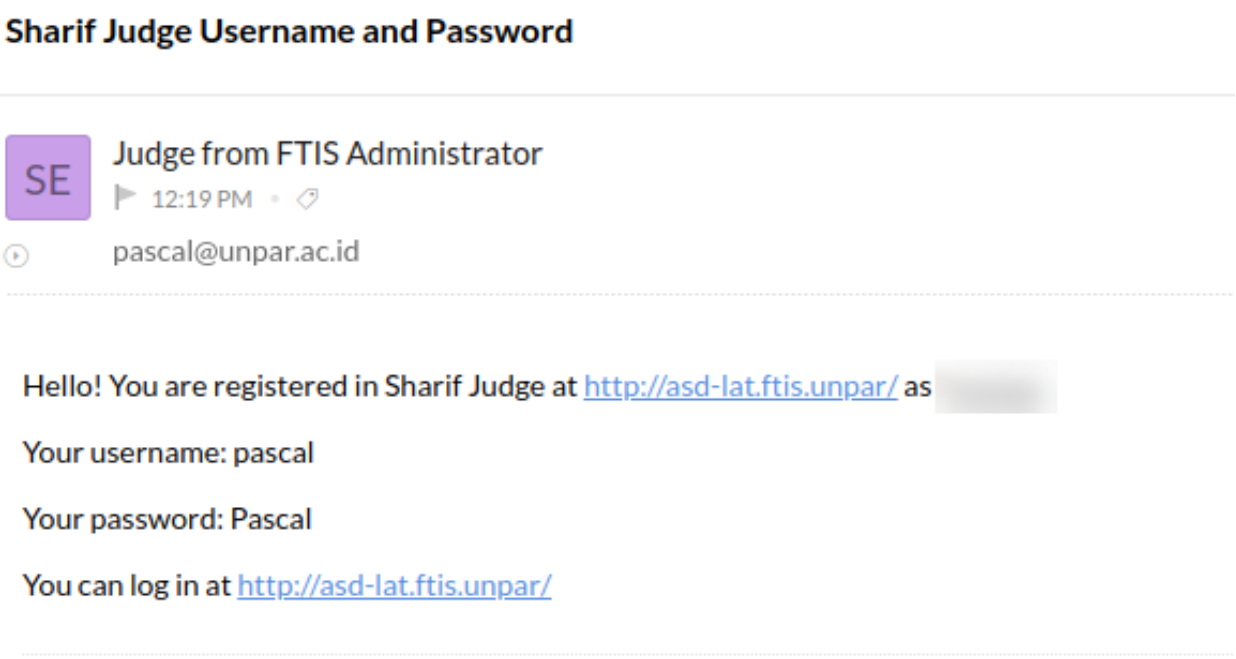
```
...
196     // insert default settings to table 'settings'
197     $result = $this->db->insert_batch('settings', array(
```

```
198     array('shj_key' => 'timezone',              'shj_value' => 'Asia/Jakarta
    '),
199     array('shj_key' => 'tester_path',           'shj_value' => dirname(
    __FILE__, 3) . "/restricted/tester"),
200     array('shj_key' => 'assignments_root',      'shj_value' => dirname(
    __FILE__, 3) . "/restricted/assignments"),
...
```

Setelah melakukan beberapa perubahan, folder assignment dapat dihapus.

5.4.2 *Add User Email are not formatted correctly*

Isu dengan kode unik #5 mengatakan bahwa pendaftaran peserta via *email* tidak berfungsi dengan baik. Pengguna mendapatkan *email* dengan format yang tidak beraturan. Contohnya *username* yang tertukar dengan *role* dan *password* tertukar dengan *display name* seperti Gambar 5.33



Gambar 5.33: Format *Email* Tidak Beraturan

Hal di atas terjadi karena penambahan parameter "*Display Name*" pada fungsi *add_user* yang terdapat di *model User_model.php*. Persoalan ini dapat diatasi dengan memperbaiki urutan data yang dikirimkan ke email. Berikut beberapa baris potongan kode program

```
...
231     $text = str_replace('{SITE_URL}', base_url(), $text);
232     $text = str_replace('{ROLE}', $user[3], $text);
233     $text = str_replace('{USERNAME}', $user[0], $text);
234     $text = str_replace('{PASSWORD}', htmlspecialchars($user[2]), $text);
235     $text = str_replace('{LOGIN_URL}', base_url(), $text);
...
```

Perubahan kode terjadi pada baris 232 dan 234. Berikut hasil perubahan kode program yang terjadi di di *User_model.php*

```
...
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);
...
```

Setelah melakukan perubahan kode program, pendaftaran peserta via *email* dapat berfungsi dengan baik. *Email* yang diterima pengguna telah menggunakan format yang benar.

5.4.3 *Cannot Create Assignments*

Isu dengan kode unik #7 mengatakan bahwa Pak Husnul selaku dosen Algoritma & Struktur Data, tidak bisa membuat *assignment*. *Error log* yang dihasilkan tidak menunjukan pesan kesalahan apapun. Berikut isi *error log* yang terdapat di direktori path/var/log/apache2/error.log

```
[Wed Jan 31 06:25:02.516154 2018] [mpm_prefork:notice] [pid 1734] AH00163: Apache
   /2.4.18 (Ubuntu) configured -- resuming normal operations
[Wed Jan 31 06:25:02.516182 2018] [core:notice] [pid 1734] AH00094: Command line:
    '/usr/sbin/apache2
```

Persoalan terjadi karena tidak ada atribut *archived_assignment* di tabel *shj_assignments*. Persoalan ini dapat diatasi dengan menambahkan atribut *archived_assignment* di tabel *shj_assignments*, sehingga struktur tabel *shj_assignments* menjadi seperti Tabel 5.5. *Assignment* berhasil dibuat setelah atribut *archived_assignment* ditambahkan.

5.4.4 *Link* ke *New Home of Sharif-Judge*

Isu dengan kode unik #8 mengatakan bahwa pada halaman *Add* atau *Edit Assignment*, terdapat *link* menuju ke dokumentasi *Sharif Judge* milik mjnaderi. *Link* tersebut diupdate menuju ke dokumentasi *SharIF Judge* yang baru yaitu `https://github.com/ftisunpar/Sharif-Judge/`. Persoalan ini dapat diselesaikan dengan mengubah *link* dokumentasi pada halaman *add_assignment.twig*. Berikut beberapa baris potongan kode program

```
...
80      <span class="title_menu_item">
81              <a href="https://github.com/mjnaderi/Sharif-Judge/blob/docs/v1.4/
   add_assignment.md" target="_blank"><i class="fa fa-question-circle color1"></i
   > Help</a>
82      </span>
...
134     <span class="form_comment">
135     <a href="https://github.com/mjnaderi/Sharif-Judge/blob/docs/v1.4/
   tests_structure.md" target="_blank">Use this structure</a>
136     </span>
...
175     <label for="form_late_rule">Coefficient rule (<a target="_blank" href="
   https://github.com/mjnaderi/Sharif-Judge/blob/docs/v1.4/add_assignment.md#
   coefficient-rule">?</a>)</label><br>
...
190     <th rowspan="2">Allowed<br>Languages (<a target="_blank" href="https://
   github.com/mjnaderi/Sharif-Judge/blob/docs/v1.4/add_assignment.md#allowed-
   languages">?</a>)</th>
```

```
191     <th rowspan="2">Diff<br>Command (<a target="_blank" href="https://github.
    com/mjnaderi/Sharif-Judge/blob/docs/v1.4/add_assignment.md#diff-command">?</a
    >)</th>
192     <th rowspan="2">Diff<br>Argument (<a target="_blank" href="https://github.
    com/mjnaderi/Sharif-Judge/blob/docs/v1.4/add_assignment.md#diff-arguments">?</
    a>)</th>
193     <th rowspan="2">Upload<br>Only (<a target="_blank" href="https://github.
    com/mjnaderi/Sharif-Judge/blob/docs/v1.4/add_assignment.md#upload-only">?</a>)
    </th>
...
```

Perubahan kode terjadi pada baris 81, 135, 175, 190, 191, 192 dan 193. Berikut hasil perubahan kode program yang terjadi di *add_assignment.twig*

```
...
80      <span class="title_menu_item">
81              <a href="https://github.com/ftisunpar/Sharif-Judge/blob/docs/v1
    .4/add_assignment.md" target="_blank"><i class="fa fa-question-circle color1
    "></i> Help</a>
82      </span>
...
134     <span class="form_comment">
135     <a href="https://github.com/ftisunpar/Sharif-Judge/blob/docs/v1.4/
    tests_structure.md" target="_blank">Use this structure</a>
136     </span>
...
175     <label for="form_late_rule">Coefficient rule (<a target="_blank" href="
    https://github.com/ftisunpar/Sharif-Judge/blob/docs/v1.4/add_assignment.md#
    coefficient-rule">?</a>)</label><br>
...
190     <th rowspan="2">Allowed<br>Languages (<a target="_blank" href="https://
    github.com/ftisunpar/Sharif-Judge/blob/docs/v1.4/add_assignment.md#allowed-
    languages">?</a>)</th>
191     <th rowspan="2">Diff<br>Command (<a target="_blank" href="https://github.
    com/ftisunpar/Sharif-Judge/blob/docs/v1.4/add_assignment.md#diff-command">?</a
    >)</th>
192     <th rowspan="2">Diff<br>Argument (<a target="_blank" href="https://github.
    com/ftisunpar/Sharif-Judge/blob/docs/v1.4/add_assignment.md#diff-arguments
    ">?</a>)</th>
193     <th rowspan="2">Upload<br>Only (<a target="_blank" href="https://github.
    com/ftisunpar/Sharif-Judge/blob/docs/v1.4/add_assignment.md#upload-only">?</a
    >)</th>
...
```

5.4.5 Hapus Perbedaan antara *Admin* dan *Student* pada PDF *Download*

Isu dengan kode unik #7 mengatakan agar mengubah beberapa bagian hasil implementasi kode pada kelas *controller Assignments.php*. Kode tersebut terdapat pada fungsi unduh *assignment* dengan ketentuan yang telah dirancang pada sub bab 4.3. Bagian kode *&& $this->user->level == 0* berfungsi untuk mengijinkan pengguna dengan *role admin* dapat mengunduh *assignment*. Bagian tersebut dihilangkan agar adanya transparansi antara dosen dan peserta, sehingga dosen saat membuat *assignment* lebih yakin bahwa soalnya tidak akan bisa diunduh. Berikut beberapa

baris potongan kode program

```
...
112             if ( ! $pdf_files )
113                     show_error("File not found");
114             elseif (!$this->assignment_model->assignment_info($assignment_id)
    ['open'] && $this->user->level == 0 )
115                     show_error('Selected assignment has been closed.');
116             elseif  ( ! $this->assignment_model->is_participant($this->
    assignment_model->assignment_info($assignment_id)['participants'],$this->user
    ->username) )
117                     show_error('You are not registered for submitting.');
118             elseif ( shj_now() > $finishtime + $extratime && $this->user->
    level == 0 )
119                     show_error('Selected assignment has finished.');
120             elseif ( shj_now() < $starttime && $this->user->level == 0 )
121                     show_error('Selected assignment has not started.');
...
```

Perubahan kode terjadi pada baris 114, 116, 118 dan 120. Berikut hasil perubahan kode program yang terjadi di *Assignments.php*

```
...
107             if ($problem_id === NULL)
108                     $pattern = rtrim($this->settings_model->get_setting('
    assignments_root'),'/')."/assignment_{$assignment_id}/*.pdf";
109             else
110                     $pattern = rtrim($this->settings_model->get_setting('
    assignments_root'),'/')."/assignment_{$assignment_id}/p{$problem_id}/*.pdf";
111             $pdf_files = glob($pattern);
112             if ( ! $pdf_files )
113                     show_error("File not found");
114             elseif (!$this->assignment_model->assignment_info($assignment_id)
    ['open'])
115                     show_error('Selected assignment has been closed.');
116             elseif  ( ! $this->assignment_model->is_participant($this->
    assignment_model->assignment_info($assignment_id)['participants'],$this->user
    ->username) )
117                     show_error('You are not registered for submitting.');
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.');
...
```

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.

[8] PHPOffice (2010) Phpspreadsheet - a pure php library for reading and writing spreadsheet files. https://github.com/PHPOffice/PhpSpreadsheet. 10 April 2018.

LAMPIRAN A

KODE PROGRAM HALAMAN *24-HOUR LOGS*

Kode Program kelas *model* halaman *24-hour Logs*.

Listing A.1: *Logs_model.php*

```
<?php
/**
 * SharIF Judge online judge
 * @file Logs_model.php
 * @author Stillmen Vallian <stillmen.v@gmail.com>
 */
defined('BASEPATH') OR exit('No direct script access allowed');

class Logs_model extends CI_Model
{

    public function __construct()
    {
        parent::__construct();
    }

    // ------------------------------------------------------------------------

    /**
     * Mencatat Logs (Tabel shj_logins)
     *
     *
     *
     */
    public function insert_to_logs($username, $ip_adrress)
    {
        $query = $this->db->get('logins')->result_array();

        //menghapus timestamp user yang lebih dari 24 jam
        foreach ($query as $row)
        {
            $temptime=strtotime('+24 hour', strtotime($row['timestamp']));
            if ($temptime < shj_now()) {
                # delete
                $this->db->where('timestamp', $row['timestamp']);
                $this->db->delete('logins');
            }
        }

        $result = $this->db->query("SELECT * FROM shj_logins WHERE username='".$username."' AND ip_address!='".$ip_adrress."' ORDER BY timestamp DESC")->row();
        if ($result === NULL) {
            $logins = array(
          'username' => $username,
          'ip_address' => $ip_adrress
        );
        $this->db->insert('logins', $logins);
        }
        else{
            $get_last_login_id = $result -> login_id;
            $logins = array(
                'username' => $username,
                'ip_address' => $ip_adrress,
                'last_24h_login_id' => $get_last_login_id
            );
            $this->db->insert('logins', $logins);
        }
    }

  // ------------------------------------------------------------------------

    /**
     * Get All Logs
     *
     * Returns an array of all logs
     *
     * @return mixed
     */
    public function get_all_logs()
    {
        return $this->db->order_by('login_id', 'desc')->get('logins')->result_array();
    }
}
```

Kode Program kelas *view* halaman *24-hour Logs*.

Listing A.2: *logs.twig*

```
{#
 # SharIF Judge
 # file: logs.twig
 # author: Stillmen Vallian <stillmen.v@gmail.com>
 #}
{% set selected = 'logs' %}
{% extends 'templates/base.twig' %}
{% block icon %}fa-book{% endblock %}
{% block title %}24-hour Log{% endblock %}
{% block head_title %}24-hour Log{% endblock %}



{% block main_content %}

<p style="text-align:justify">Use this table to detect account lendings between students in a pre-seated exam environment. In a
      pre-seated environment, student should have logged in from only one IP address within the exam time period. Whenever a user
      logs in from different IP address in the past 24 hours, last column of this table will show link to the previous login that
      happened. In such case, proctor can analyse from those two logins, whether the same login has been used in different devices
       (meaning the account had been lent).</p>

<table class="sharif_table">
    <thead>
        <tr>
            <th>#</th>
            <th>Login ID</th>
            <th>Username</th>
      <th>IP Address</th>
      <th>Login Time</th>
      <th>Log from different IP (< 24 hours)</th>
        </tr>
    </thead>
    {% for log in logs %}
        <tr>
            <td>{{ loop.index }}</td>
            <td id='{{ log.login_id }}'>{{ log.login_id }}</td>
            <td>{{ log.username }}</td>
      <td>{{ log.ip_address }}</td>
      <td>{{ log.timestamp }}</td>
      <td><a href="#{{_log.last_24h_login_id_}}">{{ log.last_24h_login_id }}</a></td>
        </tr>
    {% endfor %}
</table>
{% endblock %} {# main_content #}
```

Kode Program kelas *controller* halaman *24-hour Logs*.

Listing A.3: *Logs.php*

```
<?php
/**
 * SharIF Judge online judge
 * @file Logs.php
 * @author Stillmen Vallian <stillmen.v@gmail.com>
 */
defined('BASEPATH') OR exit('No_direct_script_access_allowed');

class Logs extends CI_Controller
{

    public function __construct()
    {
        parent::__construct();
        if ( ! $this->session->userdata('logged_in')) // if not logged in
            redirect('login');
        if ( $this->user->level <= 2) // permission denied
            show_404();
    }

    // ------------------------------------------------------------------------

    public function index()
    {

        $data = array(
            'logs' => $this->logs_model->get_all_logs()
        );

        $this->twig->display('pages/admin/logs.twig', $data);
    }

    // ------------------------------------------------------------------------

}
```

LAMPIRAN B

KODE PROGRAM HALAMAN *HALL OF FAME*

Kode Program kelas *model* halaman *Hall of Fame*.

Listing B.1: *Hof_model.php*

```
<?php
/**
 * SharIF Judge online judge
 * @file Hof_model.php
 * @author Stillmen Vallian <stillmen.v@gmail.com>
 */
defined('BASEPATH') OR exit('No direct script access allowed');

class Hof_model extends CI_Model
{

    public function __construct()
    {
        parent::__construct();
    }


    // ------------------------------------------------------------------


    /**
     * Get data for Hall of Fame
     *
     *
     * @return mixed
     */
    public function get_all_final_submission()
    {
        //include upload only file: remove " AND file_type!='txt' AND file_type!='pdf' AND file_type!='zip' ";
    return $this->db->query("SELECT username, SUM(pre_score * coefficient / 100) AS totalscore FROM shj_submissions WHERE is_final
          =1 AND file_type!='txt' AND file_type!='pdf' AND file_type!='zip' GROUP BY username ORDER BY totalscore DESC")->
          result_array();
    }


    // ------------------------------------------------------------------

    /**
     * Get details assignments & problems for selected user
     *
     *
     * @return mixed
     */
    public function get_all_user_assignments($username)
    {
        $this->load->model('assignment_model');
        $details = $this->db->query("SELECT assignment, problem, (pre_score * coefficient / 100) AS score FROM shj_submissions
              WHERE is_final=1 AND username='$username' AND file_type!='txt' AND file_type!='pdf' AND file_type!='zip' ORDER BY
              assignment ASC")->result_array();
        foreach ($details as $key => $detail) {
            $assignment_id = $detail['assignment'];
            $problem_id = $detail['problem'];
            $details[$key]['assignment'] = $this->assignment_model->assignment_info($assignment_id)['name'];
            $details[$key]['scoreboard'] = $this->assignment_model->assignment_info($assignment_id)['scoreboard'];
            $details[$key]['problem'] = $this->assignment_model->problem_info($assignment_id, $problem_id)['name'];
    }
        return $details;
    }
}
```

Kode Program kelas *view* halaman *Hall of Fame*.

Listing B.2: *halloffame.twig*

```
{#
 # SharIF Judge
 # file: halloffame.twig
 # author: Stillmen Vallian <stillmen.v@gmail.com>
 #}
{% set selected = 'halloffame' %}
{% extends 'templates/base.twig' %}
{% block icon %}fa fa-list-alt fa-lg{% endblock %}
```

```
{% block title %}Hall of Fame{% endblock %}
{% block head_title %}Hall of Fame{% endblock %}
{% block other_assets %}
    <link rel='stylesheet' type='text/css' href='{{ base_url("assets/snippet/jquery.snippet.css") }}'/>
    <link rel='stylesheet' type='text/css' href='{{ base_url("assets/snippet/themes/github.css") }}'/>
    <script type='text/javascript' src="{{ base_url("assets/snippet/jquery.snippet.js") }}"></script>
    <link rel='stylesheet' type='text/css' href='{{ base_url("assets/reveal/reveal.css") }}'/>
    <script type='text/javascript' src="{{ base_url("assets/reveal/jquery.reveal.js") }}"></script>
{% endblock %}


{% block main_content %}
<div style="height:15px"></div>
<table class="sharif_table">
    <thead>
        <tr>
            <th>#</th>
      <th>Rank</th>
            <th>Username</th>
            <th>Total Score</th>
        </tr>
    </thead>
  {% set tempTotalScore = 0 %}
  {% set tempLoop = 0 %}
    {% for hof in hofs %}
        <tr class="hof_details" style="cursor: pointer;">
            <td>{{ loop.index }}</td>
      {% if loop.index == 1 %}
        <td>1</td>
        {% set tempTotalScore = hof.totalscore %}
        {% set tempLoop = tempLoop+1 %}
      {% elseif tempTotalScore == hof.totalscore %}
        <td>{{ tempLoop }}</td>
      {% else %}
        {% set tempTotalScore = hof.totalscore %}
        {% set tempLoop = tempLoop+1 %}
        <td>{{ tempLoop }}</td>
      {% endif %}
            <td class="username">{{ hof.username }}</td>
      <td>{{ hof.totalscore }}</td>
        </tr>
    {% endfor %}
</table>
{% endblock %} {# main_content #}

{% block body_end %}
<div id="shj_modal" class="reveal-modal xlarge">
    <div class="modal_inside">
        <div style="text-align: center;">Loading<br><img src="{{ base_url('assets/images/loading.gif') }}"/></div>
    </div>
    <a class="close-reveal-modal">&#215;</a>
</div>
{% endblock %}
```

Kode Program kelas *controller* halaman *Hall of Fame*.

Listing B.3: *Halloffame.php*

```
<?php
/**
 * SharIF Judge online judge
 * @file Halloffame.php
 * @author Stillmen Vallian <stillmen.v@gmail.com>
 */
defined('BASEPATH') OR exit('No direct script access allowed');

class Halloffame extends CI_Controller
{


    public function __construct()
    {
        parent::__construct();
        if ( ! $this->session->userdata('logged_in')) // if not logged in
            redirect('login');
        $this->load->model('hof_model');
    }

    // ------------------------------------------------------------------------

    public function index()
    {
        $data = array(
            'hofs' => $this->hof_model->get_all_final_submission()
        );
        $this->twig->display('pages/halloffame.twig', $data);
    }

    // ------------------------------------------------------------------------

    /**
     * Controller for shows the details of the score
     * Called by ajax request
     */
    public function hof_details()
    {
        if ( ! $this->input->is_ajax_request() )
            show_404();
```

```
41          $username = $this->input->post('username');
42
43          $json_result = $this->hof_model->get_all_user_assignments($username);
44
45          $this->output->set_header('Content-Type: application/json; charset=utf-8');
46          echo json_encode($json_result);
47      }
48  }
```

LAMPIRAN C

KODE PROGRAM *SHJ_FUNCTIONS.JS*

Kode Program *shj_functions.js*.

Listing C.1: *shj_functions.js*

```
/**
 * SharIF Judge
 * @file shj_functions.js
 * @author Mohammad Javad Naderi <mjnaderi@gmail.com>
 */

// These words are used in countdown timer
shj.time_words = ['day', 'days', 'hour', 'hours', 'minute', 'minutes', 'second' , 'seconds'];

// selectText is used for "Select All" when viewing a submitted code
jQuery.fn.selectText = function(){
    var doc = document
        , element = this[0]
        , range, selection
        ;
    if (doc.body.createTextRange) {
        range = document.body.createTextRange();
        range.moveToElementText(element);
        range.select();
    } else if (window.getSelection) {
        selection = window.getSelection();
        range = document.createRange();
        range.selectNodeContents(element);
        selection.removeAllRanges();
        selection.addRange(range);
    }
};

shj.html_encode = function(value) {
    return $('<div/>').text(value).html();
}

shj.supports_local_storage = function() {
    try {
        return 'localStorage' in window && window['localStorage'] !== null;
    } catch(e){
        return false;
    }
}

shj.loading_start = function()
{
    $('#top_bar_.shj-spinner').css('display', 'block');
}

shj.loading_finish = function()
{
    $('#top_bar_.shj-spinner').css('display', 'none');
}

shj.loading_error = function()
{
    noty({
        text: 'An_error_encountered_while_processing_your_request._Check_your_network_connection.',
        layout: 'bottomRight',
        type: 'error',
        timeout: 3500
    });
}

shj.loading_failed = function(message)
{
    noty({
        text: 'Request_failed._Server_says:_' + message,
        layout: 'bottomRight',
        type: 'error',
        timeout: 3500
    });
}

shj.sync_server_time = function () {
    $.ajax({
        type: 'POST',
        url: shj.site_url + 'server_time',
```

```
        data: {
            shj_csrf_token: shj.csrf_token
        },
        success: function (response) {
            shj.offset = moment(response).diff(moment());
        }
    });
}

shj.update_clock = function(){
    if (Math.abs(moment().diff(shj.time))>3500){
        //console.log('moment: '+moment()+' time: '+time+' diff: '+Math.abs(moment().diff(time)));
        shj.sync_server_time();
    }
    shj.time = moment();
    var now = moment().add('milliseconds', shj.offset);
    $('.timer').html('Server Time: '+now.format('MMM DD - HH:mm:ss'));
    var countdown = shj.finish_time.diff(now);
    if (countdown<=0 && countdown + shj.extra_time.asMilliseconds()>=0){
        countdown = countdown + shj.extra_time.asMilliseconds();
        $("div#extra_time").css("display","block");
    }
    else
        $("div#extra_time").css("display","none");
    if (countdown<=0){
        countdown=0;
    }

    countdown = Math.floor(moment.duration(countdown).asSeconds());
    var seconds = countdown%60; countdown=(countdown-seconds)/60;
    var minutes = countdown%60; countdown=(countdown-minutes)/60;
    var hours = countdown%24; countdown=(countdown-hours)/24;
    var days = countdown;
    $("#time_days").html(days);
    $("#time_hours").html(hours);
    $("#time_minutes").html(minutes);
    $("#time_seconds").html(seconds);
    if(days==1)
        $("#days_label").html(shj.time_words[0]);
    else
        $("#days_label").html(shj.time_words[1]);
    if(hours==1)
        $("#hours_label").html(shj.time_words[2]);
    else
        $("#hours_label").html(shj.time_words[3])
    if(minutes==1)
        $("#minutes_label").html(shj.time_words[4]);
    else
        $("#minutes_label").html(shj.time_words[5]);
    if(seconds==1)
        $("#seconds_label").html(shj.time_words[6]);
    else
        $("#seconds_label").html(shj.time_words[7]);
}

shj.sidebar_open = function(time){
    if (time==0){
        $(".sidebar_text").css('display', 'inline-block');
        $("#sidebar_bottom_p").css('display', 'block');
        $("#side_bar").css('width', '173px');
        $("#main_container").css('left', '173px');
    }
    else{
        $("#side_bar").animate({width: '173px'}, time, function(){
            $(".sidebar_text").css('display', 'inline-block');
            $("#sidebar_bottom_p").css('display', 'block');
        });
        $("#main_container").animate({'left':'173px'}, time*1.7);
    }
    $("i#collapse").removeClass("fa-caret-square-o-right");
    $("i#collapse").addClass("fa-caret-square-o-left");
}

shj.sidebar_close = function(time){
    if (time==0){
        $(".sidebar_text").css('display', 'none');
        $("#sidebar_bottom_p").css('display', 'none');
        $("#side_bar").css('width', '48px');
        $("#main_container").css('left', '48px');
    }
    else{
        $(".sidebar_text").css('display', 'none');
        $("#sidebar_bottom_p").css('display', 'none');
        $("#side_bar").animate({width: '48px'}, time);
        $("#main_container").animate({'left': '48px'}, time*1.7);
    }
    $("i#collapse").removeClass("fa-caret-square-o-left");
    $("i#collapse").addClass("fa-caret-square-o-right");
}

shj.toggle_collapse = function(){
    if (shj.sidebar == "open"){
        shj.sidebar = "close";
        shj.sidebar_close(200);
        if (shj.supports_local_storage())
            localStorage.shj_sidebar = 'close';
        else
            $.cookie('shj_sidebar','close',{path:'/', expires: 365});
    }
```

```
174     else if (shj.sidebar == "close"){
175         shj.sidebar = "open";
176         shj.sidebar_open(200);
177         if (shj.supports_local_storage())
178             localStorage.shj_sidebar = 'open';
179         else
180             $.cookie('shj_sidebar','open',{path:'/', expires: 365});
181     }
182 }
183
184
185
186 // Notifications
187 shj.notif_check_time = null;
188 shj.check_notifs = function () {
189     if (shj.notif_check_time == null)
190         shj.notif_check_time = moment().add('milliseconds', shj.offset - (shj.notif_check_delay * 1000));
191     $.ajax({
192         type: 'POST',
193         url: shj.site_url+'notifications/check',
194         data: {
195             time: shj.notif_check_time.format('YYYY-MM-DD_HH:mm:ss'),
196             shj_csrf_token: shj.csrf_token
197         },
198         success: function (data) {
199             if (data == "new_notification") {
200                 noty({
201                     text: 'New_Notification',
202                     layout: 'bottomRight',
203                     type: 'information',
204                     closeWith: ['click', 'button'],
205                     animation: {
206                         open: {height: 'toggle'},
207                         close: {height: 'toggle'},
208                         easing: 'swing',
209                         speed: 300
210                     }
211                 });
212                 alert("New_Notification");
213             }
214         }
215     });
216     shj.notif_check_time = moment().add('milliseconds', shj.offset);
217 }
218
219
220
221
222 /**
223  * Notifications
224  */
225 $(document).ready(function () {
226     $('.ttl_n').click(function(){
227         var id = $(this).parents('.notif').data('id');
228         window.location = shj.site_url+'notifications#number'+id;
229     });
230     $('.edt_n').click(function () {
231         var id = $(this).parents('.notif').data('id');
232         window.location = shj.site_url+'notifications/edit/'+id;
233     });
234     $('.del_n').click(function () {
235         var notif = $(this).parents('.notif');
236         var id = $(notif).data('id');
237         noty({
238             text: 'Are_you_sure_you_want_to_delete_this_notification?',
239             layout: 'center',
240             type: 'confirm',
241             animation: {
242                 open: {height: 'toggle'},
243                 close: {height: 'toggle'},
244                 easing: 'swing',
245                 speed: 300
246             },
247             buttons: [
248                 {addClass: 'btn_shj-red', text: 'Yes,_Delete', onClick: function ($noty) {
249                     $noty.close();
250                     $.ajax({
251                         type: 'POST',
252                         url: shj.site_url + 'notifications/delete',
253                         data: {
254                             id: id,
255                             shj_csrf_token: shj.csrf_token
256                         },
257                         beforeSend: shj.loading_start,
258                         complete: shj.loading_finish,
259                         error: shj.loading_error,
260                         success: function (response) {
261                             if (response.done) {
262                                 notif.animate({backgroundColor: '#FF7676'}, 1000, function () {
263                                     notif.remove();
264                                 });
265                                 noty({text: 'Notification_deleted', layout: 'bottomRight', type: 'success', timeout: 5000});
266                             }
267                             else
268                                 shj.loading_failed(response.message);
269                         }
270                     });
271                 }
272                 },
```

```
273                 {addClass: 'btn_shj-blue', text: 'No,_Don\'t_Delete', onClick: function ($noty) {
274                     $noty.close();
275                 }}
276             ]
277         });
278     });
279
280     if ( shj.check_for_notifications )
281         window.setInterval(shj.check_notifs,(shj.notif_check_delay*1000));
282
283 });
284
285
286
287
288 /**
289  * Scrollbars
290  */
291 $(document).ready(function(){
292     $('.scroll-wrapper').nanoScroller({
293         contentClass: 'scroll-content'
294     });
295     $('#main_content').resize(function(){
296         // update the scrollbar
297         $('.scroll-wrapper').nanoScroller();
298     });
299     $('.widget_contents_container').resize(function(){
300         // update the scrollbar
301         $('.scroll-wrapper').nanoScroller();
302     });
303 });
304
305
306
307
308 /**
309  * Sidebar
310  */
311 $(document).ready(function () {
312     if (shj.supports_local_storage())
313         shj.sidebar = localStorage.shj_sidebar;
314     else
315         shj.sidebar = $.cookie('shj_sidebar');
316
317     if (shj.sidebar != 'open' && shj.sidebar != 'close') {
318         shj.sidebar = 'open';
319         if (shj.supports_local_storage())
320             localStorage.shj_sidebar = 'open';
321         else
322             $.cookie('shj_sidebar', 'open', {path: '/', expires: 365});
323     }
324     if (shj.sidebar == "open")
325         shj.sidebar_open(0);
326     else
327         shj.sidebar_close(0);
328
329     $("#shj_collapse").click(shj.toggle_collapse);
330
331     // update the clock and countdown timer every 1 second
332     shj.update_clock();
333     window.setInterval(shj.update_clock, 1000);
334 });
335
336
337
338
339
340 /**
341  * Top Bar
342  */
343 $(document).ready(function () {
344     $("#top_bar").hoverIntent({
345         over: function () {
346             $(this).children(".top_menu").show();
347             $(this).addClass('shj_white');
348         },
349         out: function () {
350             $(this).children(".top_menu").hide();
351             $(this).removeClass('shj_white');
352         },
353         selector: '.top_object.shj_menu'
354     });
355     $(".select_assignment").click(
356         function () {
357             var id = $(this).children('i').addBack('i').data('id');
358             $.ajax({
359                 type: 'POST',
360                 url: shj.site_url + 'assignments/select',
361                 data: {
362                     assignment_select: id,
363                     shj_csrf_token: shj.csrf_token
364                 },
365                 beforeSend: shj.loading_start,
366                 complete: shj.loading_finish,
367                 error: shj.loading_error,
368                 success: function (response) {
369                     if (response.done)
370                     {
371                         var checkboxes = $(".select_assignment").children('i').addBack('i');
```

```
372                         checkboxes.removeClass('fa-check-square-o␣color6').addClass('fa-square-o');
373                         checkboxes.filter("[data-id='" + id + "']").removeClass('fa-square-o').addClass('fa-check-square-o␣color6'
                                );
374                         $(".assignment␣name").html($('.top␣object␣[data-id="' + id + '"]').parents('.assignment␣block').children('
                                .assignment␣item').html());
375                         shj.finish_time = moment(response.finish_time);
376                         shj.extra_time  = moment.duration(parseInt(response.extra_time, 10), 'seconds');
377                         shj.update_clock();
378                     }
379                     else
380                         shj.loading_failed(response.message);
381                 }
382             });
383         }
384     );
385 });
386
387
388
389
390
391 /**
392  * "Users" page
393  */
394 $(document).ready(function(){
395     $('.delete␣user').click(function(){
396         var row = $(this).parents('tr');
397         var user_id = row.data('id');
398         var username = row.children('#un').html();
399         noty({
400             text: 'Are␣you␣sure␣you␣want␣to␣delete␣this␣user?<br>User␣ID:␣'+user_id+'<br>Username:␣'+username+'<br><i␣class="
                    splashy-warning␣triangle"></i>␣All␣submissions␣of␣this␣user␣will␣be␣deleted.',
401             layout: 'center',
402             type: 'confirm',
403             animation: {
404                 open: {height: 'toggle'},
405                 close: {height: 'toggle'},
406                 easing: 'swing',
407                 speed: 300
408             },
409             buttons: [
410                 {addClass: 'btn␣shj-red', text: 'Yes,␣Delete', onClick: function($noty) {
411                     $noty.close();
412                     $.ajax({
413                         type: 'POST',
414                         url: shj.site_url+'users/delete',
415                         data: {
416                             user_id: user_id,
417                             shj_csrf_token: shj.csrf_token
418                         },
419                         beforeSend: shj.loading_start,
420                         complete: shj.loading_finish,
421                         error: shj.loading_error,
422                         success: function(response){
423                             if (response.done)
424                             {
425                                 row.animate({backgroundColor: '#FF7676'},1000, function(){row.remove();});
426                                 noty({text: 'User␣'+username+'␣deleted.', layout:'bottomRight', type: 'success', timeout: 5000});
427                             }
428                             else
429                                 shj.loading_failed(response.message);
430                         }
431                     });
432                 }
433                 },
434                 {addClass: 'btn␣shj-blue', text: 'No,␣Don\'t␣Delete', onClick: function($noty){$noty.close();}}
435             ]
436         });
437     });
438     $('.delete␣submissions').click(function(){
439         var row = $(this).parents('tr');
440         var user_id = row.data('id');
441         var username = row.children('#un').html();
442         noty({
443             text: 'Are␣you␣sure␣you␣want␣to␣delete␣this␣user\'s␣submissions?<br>User␣ID:␣'+user_id+'<br>Username:␣'+username,
444             layout: 'center',
445             type: 'confirm',
446             animation: {
447                 open: {height: 'toggle'},
448                 close: {height: 'toggle'},
449                 easing: 'swing',
450                 speed: 300
451             },
452             buttons: [
453                 {addClass: 'btn␣shj-red', text: 'Yes,␣Delete', onClick: function($noty) {
454                     $noty.close();
455                     $.ajax({
456                         type: 'POST',
457                         url: shj.site_url+'users/delete_submissions',
458                         data: {
459                             user_id: user_id,
460                             shj_csrf_token: shj.csrf_token
461                         },
462                         beforeSend: shj.loading_start,
463                         complete: shj.loading_finish,
464                         error: shj.loading_error,
465                         success: function(response){
466                             if (response.done)
```

```
                                noty({text: 'Submissions of user '+username+' deleted successfully.', layout:'bottomRight', type: 'success', timeout: 5000});
                            else
                                shj.loading_failed(response.message);
                        }
                    });
                }
                },
                {addClass: 'btn shj-blue', text: 'No, Don\'t Delete', onClick: function($noty){$noty.close();}}
            ]
        });
    });
});



/**
 * "Hall of Fame" page
 */
shj.modal_open = false;
$(document).ready(function(){
    $('.hof_details').click(function () {
        var row = $(this).closest("tr");    // Find the row
    var username = row.find(".username").text();
        var view_code_request = $.ajax({
            cache: true,
            type: 'POST',
            url: shj.site_url+'halloffame/hof_details',
            data: {
                username: username,
                shj_csrf_token: shj.csrf_token
            },
            success: function (response) {
                var currentAssignment = '';
                var prevAssignment = '';
                var temp = '';

                for (var i = 0; i < response.length; i++) {
                    if (response[i].scoreboard == 0) {
                        temp = temp + '-----------------------------------------------------<br><b>' + response[i].assignment + '</b><br>Scoreboard Disabled!';
                    }
                    else{
                        if (i == 0) {
                            temp = temp + '<b>' + response[i].assignment + '</b> <br>' + response[i].problem + ' : ' + response[i].score +'<br>';
                            prevAssignment = response[i].assignment;
                        }
                        else{
                            currentAssignment = response[i].assignment;
                            var index = currentAssignment.localeCompare(prevAssignment); //comparing previous assignment's name with current assignment's name
                            if (index == 0) {
                                temp = temp + response[i].problem + ' : ' + response[i].score + '<br>';
                                prevAssignment = currentAssignment;
                            }
                            else{
                                temp = temp + '-----------------------------------------------------<br><b>'+currentAssignment + '</b><br>' + response[i].problem + ' : ' + response[i].score + '<br>';
                                prevAssignment = currentAssignment;
                            }
                        }
                    }
                }
                $('.modal_inside').html('<pre class="code-column">'+temp+'</pre>');
                $('.modal_inside').prepend('<p><code> Hall of Fame Details | Username: '+username+'</code></p>');
            }
        });
        if (!shj.modal_open) {
            shj.modal_open = true;
            $('#shj_modal').reveal(
                {
                    animationspeed: 300,
                    on_close_modal: function () {
                        view_code_request.abort();
                    },
                    on_finish_modal: function () {
                        $(".modal_inside").html('<div style="text-align: center;">Loading<br><img src="'+shj.base_url+'assets/images/loading.gif"/></div>');
                        shj.modal_open = false;
                    }
                }
            );
        }
    });
});



/**
 * Set dir="auto" for all input elements
 */
$(document).ready(function(){
    $('input').attr('dir', 'auto');
});
```