Kustomisasi Sharif Judge untuk Kebutuhan Program Studi Teknik Informatika

Stillmen Vallian–2014730083

1 Data Skripsi

Pembimbing utama/tunggal: **Pascal Alfadian Nugroho**
Pembimbing pendamping: -
Kode Topik : **PAN4303**
Topik ini sudah dikerjakan selama : **1** semester
Pengambilan pertama kali topik ini pada : Semester **43 - Ganjil 17/18**
Pengambilan pertama kali topik ini di kuliah : **Skripsi 1**
Tipe Laporan : **B -** Dokumen untuk reviewer pada presentasi dan **review Skripsi 1**

2 Detail Perkembangan Pengerjaan Skripsi

Detail bagian pekerjaan skripsi sesuai dengan rencan kerja/laporan perkembangan terkahir :

1. Melakukan studi literatur mengenai *Sharif Judge* dan *CodeIgniter*.
   status : Ada sejak rencana kerja skripsi.
   hasil :

   (a) ***CodeIgniter***
   CodeIgniter merupakan sebuah *framework* bagi pengguna yang ingin membangun aplikasi web menggunakan PHP. Tujuan utamanya adalah memungkinkan para pengguna mengembangkan proyek-proyek menjadi lebih cepat dibandingkan menulis kode dari awal. *Framework* ini memiliki banyak libary untuk tugas-tugas yang biasa diperlukan, serta antarmuka dan struktur logis yang sederhana untuk mengakses library ini. *CodeIgniter* membuat para pengguna lebih fokus pada proyek dengan meminimalkan jumlah kode yang dibutuhkan untuk tugas yang diberikan [1].

   Beberapa keunggulan dari *CodeIgniter* yaitu:

   - *Framework* yang Ringan
     Inti dari sistem *CodeIgniter* hanya membutuhkan *library* yang kecil. Hal ini sangat berbeda dengan framework lain yang membutuhkan *resource* yang lebih. *Library* tambahan dimuat secara dinamis atau sesuai dengan permintaan sehingga sistem dapat berjalan cepat.
   - Menggunakan Konsep M-V-C
     CodeIgniter menggunakan pendekatan *Model-View-Controller* yang memungkinkan pemisahan anatara logika dan presentasi.
   - Menghasilkan *Clean URLs*
     URL yang dihasilkan oleh *CodeIgniter* berish dan *search-engine friendly*. *CodeIgniter* menggunakan pendekatan *segment-based* seperti:
     example.com/news/article/345
   - *Packs a Punch*
     CodeIgniter dilengkapi dengan *library* yang umumnya diperlukan untuk tugas pengembangan

[1] British Columbia Institute of Technology, "CodeIgniter Documentation," CodeIgniter User Guide - CodeIgniter 3.1.6 documentation. *https://www.codeigniter.com/user_guide/* (diakses 6 Oktober 2017)

web seperti mengakses database, mengirim email, memvalidasi data *form*, menjaga *session*, memanipulasi gambar, bekerja dengan *XML-RPC* data dan masih banyak lagi.

- *Extensible*
  Sistem dapat dengan mudah diperluas dengan menggunakan *library* pengguna, *helper*, atau melalui *class extensions* dan *system hooks*.
- Tidak Membutuhkan *Template Engine*
  CodeIgniter dilengkapi dengan *template parser* sederhana yang dapat digunakan secara opsional. *Template Engine* tidak dapat menandingi kinerja dari *native* PHP. Sintak yang harus dipelajari untuk menggunakan *Template Engine* biasanya lebih mudah dari mempelajari dasar-dasar PHP. Perhatikan potongan kode PHP di bawah ini:

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

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

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

  Terlihat *Template Engine* sedikit lebih bersih, namun harus ditukar dengan performa yang kurang baik karena *pseudo-code* harus dikonversi kembali menjadi PHP. Salah satu tujuan dari *CodeIgniter* adalah performa maksimal, oleh karena itu *CodeIgniter* tidak menggunakan *Template Engine*.
- Dokumentasi yang Baik
  Dokumentasi merupakan salah satu bagian terpenting dari kode itu sendiri. *CodeIgniter* berkomitmen membuat kode yang sangat bersih dan terdokumentasi dengan baik.

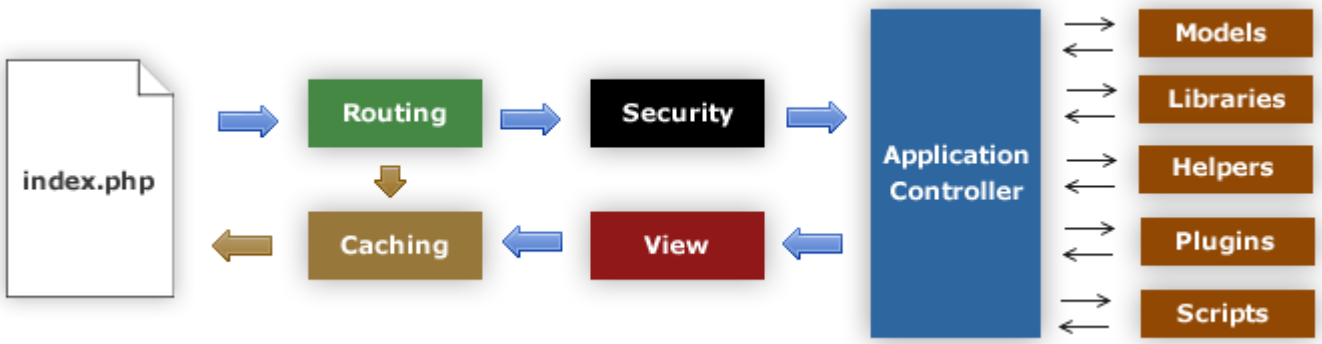
Fitur-fitur *CodeIgniter*

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*

***Flow Chart* Aplikasi**

Gambar 1 menunjukan bagaimana data mengalir ke seluruh sistem [2]:



Gambar 1: *Flow Chart* Aplikasi

i. File *index.php* berfungsi sebagai *front controller* dan menginisialisasi *resource* utama yang dibutuhkan untuk menjalankan *CodeIgniter*.

ii. *Router* memeriksa *HTTP request* untuk menentukan apa yang harus dilakukan.

iii. Jika terdapat *file cache*, maka akan langsung dikirimkan ke *browser*.

iv. *HTTP request* dan data pengguna yang dikirim akan terlebih dahulu disaring untuk alasan keamanan. *Application controller* akan dimuat setelah proses penyaringan selesai.

v. *Controller* akan memuat *model, core libraries, helpers* dan *resource* lain yang dibutuhkan untuk memproses permintaan khusus.

vi. *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.

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 minimal karena presentasi terpisah dari *scripting* PHP [3].

- *Model* merepresentasikan bagian struktur data pengguna. Biasanya kelas *Model* akan berisikan fungsi-fungsi yang membantu pengguna untuk mengambil, menyimpan, dan memperbarui informasi pada database.
- *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.
- *Controller* berfungsi sebagai perantara antara *Model*, *View*, dan *resource* lain yang dibutuhkan untuk memproses *HTTP request* dan menghasilkan halaman web.

CodeIgniter memiliki pendekatan yang cukup fleksibel terhadap *MVC* karena *Model* tidak selalu diperlukan. Para pengguna dapat membangun aplikasi minimal menggunakan *Controller* dan *View*. Hal tersebut dapat dilakukan jika pengguna tidak memerlukan adanya pemisahan tambahan

[2]British Columbia Institute of Technology, "CodeIgniter Documentation," CodeIgniter User Guide - CodeIgniter 3.1.6 documentation. *https://www.codeigniter.com/user_guide/* (diakses 6 Oktober 2017)

[3]British Columbia Institute of Technology, "CodeIgniter Documentation," CodeIgniter User Guide - CodeIgniter 3.1.6 documentation. *https://www.codeigniter.com/user_guide/* (diakses 6 Oktober 2017)

atau pengguna merasa bahwa mempertahankan sebuah *Model* membutuhkan kompleksitas yang lebih tinggi [4].

Desain dan Tujuan Arsitektur

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

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

CodeIgniter merupakan sistem yang *loosely coupled* dengan singularitas komponen yang tinggi (*dynamically instantiated*). *CodeIgniter* berusaha untuk sederhana, fleksible, dan kinerja tinggi dengan paket yang sekecil mungkin [5].

(b) ***Sharif Judge***

Sharif Judge adalah *grader* otomatis yang mampu menilai ketepatan serta performansi program yang dikumpulkan mahasiswa. Perangkat lunak ini diciptakan oleh Mohammad Javad Naderi dan bersifat *open source*. *Web Interface* perangkat lunak ini ditulis menggunakan PHP ()*framework CodeIgniter*) dan *backend* menggunakan *BASH* [6]. Selain sebagai *grader* otomatis, *Sharif Judge* juga memiliki beberapa fitur lainnya seperti:

- Beberapa peran *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 Masalah (*PDF/Markdown/HTML*)
- Penilaian ulang (*rejudge*)
- Papan nilai

[4] British Columbia Institute of Technology, "CodeIgniter Documentation," CodeIgniter User Guide - CodeIgniter 3.1.6 documentation. *https://www.codeigniter.com/user_guide/* (diakses 6 Oktober 2017)

[5] British Columbia Institute of Technology, "CodeIgniter Documentation," CodeIgniter User Guide - CodeIgniter 3.1.6 documentation. *https://www.codeigniter.com/user_guide/* (diakses 6 Oktober 2017)

[6] Mohammad Javad Naderi, "Sharif Judge," Sharif-Judge: a free and open source online judge for C, C++, Java and Python programming courses. *https://github.com/mjnaderi/Sharif-Judge* (diakses 6 Oktober 2017)

- Notifikasi

Instalasi

Untuk menjalankan *Sharif Judge* membutuhkan sebuah *server Linux* dengan persyaratan berikut [7]:

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

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

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

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

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

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

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

[7] Mohammad Javad Naderi, "Sharif Judge Documentation," Sharif Judge v1.4 Documentation. *https://github.com/mjnaderi/Sharif-Judge/tree/docs/v1.4* (diakses 6 Oktober 2017)

Clean URLs

Secara *default*, *index.php* merupakan bagian dari seluruh *urls* yang ada pada *Sharif Judge* seperti [8]:

```
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'] = '';
```

Users

Pada *Sharif Judge*, *users* dikelompokan menjadi 4 yaitu *Admins, Head Instructor, Instructor,* dan *Students* Tabel 1 menunjukan *level users* [9].

Tabel 1: *User Roles Table*

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

Setiap *users* memiliki aksi yang berbeda-beda. Aksi yang dapat dilakukan para *users* akan disesuaikan dengan *level* masing-masing. Perhatikan tabel 2 berikut:

[8] Mohammad Javad Naderi, "Sharif Judge Documentation," Sharif Judge v1.4 Documentation. *https://github.com/mjnaderi/Sharif-Judge/tree/docs/v1.4* (diakses 6 Oktober 2017)

[9] Mohammad Javad Naderi, "Sharif Judge Documentation," Sharif Judge v1.4 Documentation. *https://github.com/mjnaderi/Sharif-Judge/tree/docs/v1.4* (diakses 6 Oktober 2017)

Tabel 2: *Permission Table*

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

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

```
USERNAME EMAIL PASSWORD ROLE

* Usernames dapat berisikan huruf kecil atau nomor dan harus terdiri
antara 3 sampai 20 karakter.
* Passwords harus terdiri antara 6 sampai 30 karakter.
* Pengguna dapat menggunakan RANDOM[n] untuk menghasilkan password
acak yang terdiri dari n-digit karakter.
* ROLE harus terdiri dari salah satu ini: 'admin', 'head_instructor',
'instructor', 'student'
```

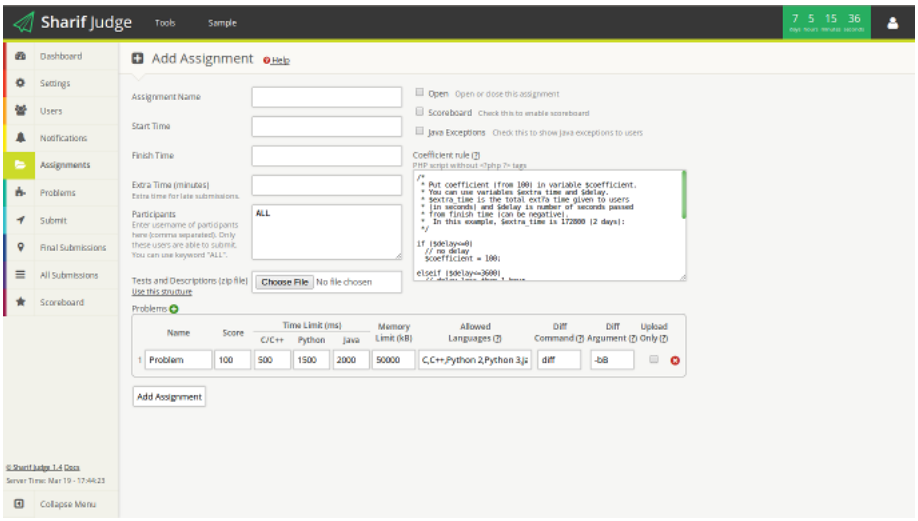
Contoh:

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

Menambah Tugas

Pengguna dapat menambahkan tugas dengan cara mengklik *Add* di halaman *Assigments* [10]. Pengguna akan melihat halaman seperti gambar 2.

[10]Mohammad Javad Naderi, "Sharif Judge Documentation," Sharif Judge v1.4 Documentation. *https://github.com/mjnaderi/Sharif-Judge/tree/docs/v1.4* (diakses 6 Oktober 2017)



Gambar 2: Tampilan Halaman *Assignments*

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

- *Assignment Name*
  Tugas akan ditampilkan dengan nama ini dalam daftar tugas.
- *Start Time*
  Users tidak dapat mengumpulkan tugas sebelum *"Start Time"*. Gunakan format ini untuk *start time*: *MM/DD/YYYY HH:MM:SS*. Contoh: *08/31/2013 12:00:00*
- *Finish Time, Extra Time*
  Users tidak dapat mengumpulkan tugas setelah *Finish Time* + *Extra Time*. Tugas yang telat akan dikalikan dengan koefisien tertentu. Pengguna harus menulis *script* PHP untuk menghitung koefisien pada bidang *"Coefficient Rule"*. Gunakan format ini untuk *finish time*: *MM/DD/YYYY HH:MM:SS*. Contoh: *08/31/2013 23:59:59*. Waktu ekstra harus dalam menit. Pengguna dapat menggunakan *. Contoh *120* (2 jam) atau *48*60* (2 hari).
- *Participants*
  Masukan *username* dari partisipan disini. Gunakan tanda koma untuk memisah *username* antar partisipan. Hanya *users* ini yang dapat mengumpulkan tugas. Pengguna dapat menggunakan kata kunci *ALL* untuk mengijinkan semua *users* agar dapat mengumpulkan tugas. Contoh: *admin, instructor1 , instructor2 ,student1 , student2,student3 , student4*.
- *Open*
  Pengguna dapat membuka atau menutup tugas untuk *students* menggunakan pilihan ini. Jika pengguna menutup tugas, *non-student users* masih dapat mengumpulkan tugas.
- Scoreboard
  Pengguna dapat mengaktifkan atau mematikan papan nilai dengan menggunakan pilihan ini.
- *Java Exceptions* Pengguna dapat mengaktifkan dan mematikan *java exceptions* yang 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 mereka membutuhkan waktu yang lebih.
- *Memory Limit*
  Pengguna dapat mengatur batas memori dalam satuan *kilobyte*, namun penggunaan *Memory Limit* tidak terlalu akurat.
- *Allowed Languages*
  Pengguna dapat mengatur bahasa untuk setiap permasalahan (dipisahkan menggunakan koma). Bahasa yang tersedia seperti *C, C++, Java, Python 2, Python 3, zip, PDF*. Pengguna dapat menggunakan *zip* atau *PDF* jika mengaktifkan pilihan *Upload Only*. Contoh: *C, C++ , zip* atau *Python 2,Python 3* atau *Java ,C*.
- *Diff Command*
  Command ini digunakan untuk membandingkan keluaran dengan keluaran yang benar. Secara *default Sharif Judge* menggunakan *diff*, namun pengguna dapat mengubah *command* pada bagian ini.
- *Diff Arguments*
  Pengguna dapat mengatur argumen dari*Diff Command* disini. Untuk melihat daftar lengkap *diff* argumen, pengguna dapat melihat *man diff*. *Sharif Judge* menambahkan dua pilihan baru yaitu *ignore* dan *identical*. *Ignore* akan menghiraukan semua baris baru dan spasi. *Identical* tidak akan menghiraukan apapun namun keluaran dari file yang dikumpulkan harus identik dengan keluaran *test case* agar dapat diterima.
- *Upload Only*
  Jika pengguna mengatur masalah sebagai *Upload-Only*, maka *Sharif Judge* tidak akan menilai tugas pada masalah tersebut. Pengguna dapat menggunakan *zip* dan *PDF* pada *allowed languages* jika mengaktifkan pilihan ini.

Contoh Tugas

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

i. Masalah 1 (Penjumlahan)
   Program pengguna akan menerima masukan bilangan *integer* n, kemudian menerima masukan lagi sebanyak n buah bilangan *integer* dan menampilkan hasil penjumlahan dari n nomor tersebut. Untuk lebih jelas, perhatikan tabel 3.

Tabel 3: Masalah 1 (Penjumlahan)

Sample Input	Sample Output
5 54 78 0 4 9	145

ii. Masalah 2 (*Max*)
   Program pengguna akan menerima masukan bilangan *integer* n, kemudian menerima masukan

lagi sebanyak n buah bilangan *integer* dan menampilkan hasil penjumlahan dari dua nilai tertinggi. Untuk lebih jelas, perhatikan tabel 4.

Tabel 4: Masalah 2 (*Max*)

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

iii. Masalah 2 (*Upload!*)

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

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

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

```
|   |   |-- output5.txt
|   |   |-- output6.txt
|   |   |-- output7.txt
|   |   |-- output8.txt
|   |   |-- output9.txt
|   |   `-- output10.txt
|   |-- desc.md
|   `-- Problem2.pdf
|-- p3
|   `-- desc.md
`-- SampleAssignment.pdf
```

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

Contoh Solusi

Permasalahan diatas dapat diselesaikan menggunakan contoh solusi berikut ini:

- Solusi Masalah 1

  Menggunakan bahasa *C*

```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++*

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

```
}
```

Menggunakan bahasa *Java*

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

- Solusi Masalah 2

  Menggunakan bahasa *C*

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

Menggunakan bahasa *C++*

```
#include<iostream>
using namespace std;
        int main(){
        int n , m1=0, m2=0;
        cin >> n;
        for(;n--;){
                int k;
                cin >> k;
```

```
                if(k>=m1){
                        m2=m1;
                        m1=k;
                }
                else if(k>m2)
                        m2=k;
                }
                cout << (m1+m2) << endl ;
        return 0;
}
```

Struktur Pengujian

Pengguna harus menyediakan sebuah *file zip* yang berisikan *test cases* ketika menambahkan tugas. *File zip* ini dapat berisikan folder-folder untuk setiap masalah. Pengguna harus memberikan nama pada folder sesuai aturan seperti *p1, p2, p3, dst.* Tugas yang menggunakan pilihan *Upload-Only* tidak membutuhkan *folder* [11].

Metode Pengecekan

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

- *Metode Input/Output Comparison*
  Dengan metode ini, pengguna harus memasukan beberapa *file input dan output* pada *folder* masalah. *Sharif Judge* akan memasukan nilai dari *file input* ke kode *users* dan membandingkan hasil keluaran dari kode *users* dengan *file output. Input files* harus berada dalam folder *"in"* dengan nama *input1.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 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
```

[11] Mohammad Javad Naderi, "Sharif Judge Documentation," Sharif Judge v1.4 Documentation. *https://github.com/mjnaderi/Sharif-Judge/tree/docs/v1.4* (diakses 6 Oktober 2017)

```
        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 */

        ...

}
```

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
```

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

```
/*
* tester.cpp
*/

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

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

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

        int sum, user_output;
        user_out >> user_output;

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

```
                        sum += a;
                }
        }

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

}
```

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

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 [12].

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.
#
```

[12] Mohammad Javad Naderi, "Sharif Judge Documentation," Sharif Judge v1.4 Documentation. *https://github.com/mjnaderi/Sharif-Judge/tree/docs/v1.4* (diakses 6 Oktober 2017)

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

2. Menganalisis kebutuhan-kebutuhan dari para dosen pengguna *Sharif Judge* dan daftar isu dalam repositori *Sharif Judge* di *Github*.
   status : Ada sejak rencana kerja skripsi.
   hasil : Analisis kebutuhan-kebutuhan sudah dilakukan. Peneliti terlebih dahulu menganalisis kebutuhan-kebutuhan yang ada pada daftar isu dalam repositori *Sharif Judge* di *Github*. Setelah menganalisis daftar isu tersebut, selanjutnya peneliti menganalisis kebutuhan dari para dosen pengguna *Sharif Judge*. Analisis kebutuhan dari para dosen pengguna *Sharif Judge* dilakukan dalam bentuk wawancara secara langsung dan melalui email. Dosen-dosen yang telah diwawancara yaitu:

   - Bapak Husnul Hakim
   - Bapak Claudio Franciscus
   - Ibu Vania Natali
   - Ibu Luciana Abednego
   - Ibu Joanna Helga

   Hasil dari analisis tersebut dicatat ke dalam *Google Sheets* dan didiskusikan bersama dosen pembimbing.

3. Merancang dan menentukan fitur yang akan diimplementasi.
   status : Ada sejak rencana kerja skripsi.
   hasil : Peneliti bersama dosen pembimbing telah merancang dan menentukan fitur yang akan diimplementasikan. Penentuan fitur tersebut bedasarkan hasil dari *Google Sheets* berikut.

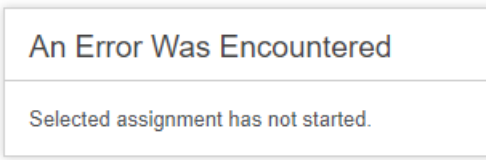
No.	Deskripsi	Sumber	Issue Number / Nama Mata Kuliah	Pembuat Issue / Nama Dosen	Status
1	Security with PHP	GitHub	#61	kathiedart	Diimplementasikan (merged)
2	Securing Assignment	GitHub	#55	wojIbuk13	Tidak diimplementasikan (pertanyaan)
3	New Function	GitHub	#53	wojIbuk13	Tidak diimplementasikan (pertanyaan)
4	Solved Problem Indicator	GitHub	#46	atiabjobayer	Tidak diimplementasikan (tidak spesifik)
5	Some Problem & Sugestion	GitHub	#45	atiabjobayer	Tidak diimplementasikan (terlalu luas)
6	Queue failed to process if submission take too long to complete?	GitHub	#32	truongan	merged
7	Compilation Error on all language	GitHub	#34	Eririn07	Tidak diimplementasikan (pertanyaan)
8	Membatasi soal (deskripsi & PDF) hanya bisa diakses saat assignment “open” dan setelah waktu mulai,	Dosen	ASD	Pak Husnul	committed a15daa0
9	Menguji kemiripan kode antar mahasiswa (Contek)	Dosen	ASD	Pak Husnul	Tidak diimplementasikan (bukan level kode, butuh port 7690 dibuka oleh UNPAR)
10	1 Akun hanya dapat login 1 waktu (Jika suatu akun sedang login, tidak ada lagi yang bisa login akun tersebut)	Dosen	ASD	Pak Husnul	in progress
11	Membatasi soal (deskripsi & PDF) hanya bisa diakses saat assignment “open” dan setelah waktu mulai,	Dosen	ASD	Ibu Vania	Diimplementasikan (committed a15daa0)
12	Sharif Judge tidak dapat menerima file dengan ekstensi *.txt untuk Pre-Test (Saat ini mengakali dengan men-zip file txt baru dikumpulkan atau mengubah file txt dengan ekstensi java)	Dosen	ASD	Ibu Vania	Diimplementasikan (committed a15daa0)
13	Membatasi soal (deskripsi & PDF) hanya bisa diakses saat assignment “open” dan setelah waktu mulai,	Dosen	DAA	Ibu Luci	Diimplementasikan (committed a15daa0)
14	Sharif Judge tidak dapat menerima file dengan ekstensi *.txt untuk Pre-Test (Saat ini mengakali dengan men-zip file txt baru dikumpulkan)	Dosen	DAA	Ibu Luci	Diimplementasikan (committed a15daa0)
15	Perlu ditambah petunjuk penamaan file input & output yg langsung bisa dilihat ketika hendak upload (input1.txt, input2.txt, dst.).	Dosen	DAA	Ibu Luci	Tidak diimplementasikan (sebelumnya sudah diimplementasikan, link ke dokumentasi)
16	Membatasi soal (deskripsi & PDF) hanya bisa diakses saat assignment “open” dan setelah waktu mulai,	Dosen	DAA	Ibu Joanna	Diimplementasikan (committed a15daa0)
17	Register peserta yg mode batch, Sharif Judge tidak minta nama orangnya (lebih baik ada nama orangnya)“	Dosen	DAA	Ibu Joanna	Diimplementasikan (merged)
18	Nama peserta seharusnya tidak bisa diganti (Bisa menjadi "mainan" dan tindak kecurangan karena dapat memberikan hint)	Dosen	DAA	Ibu Joanna	Diimplementasikan (merged)
19	Ingin memiliki fungsi dimana Assignment tidak memiliki batasan waktu (arsip soal tahun lalu dapat dikerjakan kapan saja). Assignment tipe ini tidak perlu muncul pada kalendar halaman pertama	Dosen	DAA	Ibu Joanna	Diimplementasikan (merged)
20	Ingin memiliki scoreboard global untuk semua assignment. Mirip seperti hall of fame, setiap peserta telah AC berapa soal diseluruh assignment yang ada.	Dosen	DAA	Ibu Joanna	in progress
21	Membatasi soal (deskripsi & PDF) hanya bisa diakses saat assignment “open” dan setelah waktu mulai,	Dosen	DAA dan ASD	Ko Claudio	Diimplementasikan (committed a15daa0)
22	Sharif Judge tidak dapat menerima file dengan ekstensi *.txt untuk Pre-Test (Saat ini mengakali dengan men-zip file txt baru dikumpulkan atau mengubah file txt dengan ekstensi java)	Dosen	DAA dan ASD	Ko Claudio	Diimplementasikan (committed a15daa0)
23	UI masih merepotkan (klik assignment, centang assignment yg ingin dikerjakan, klik submit, pilih nomor berapa dr assignment yg mau dikumpulkan jawabannya)	Dosen	DAA dan ASD	Ko Claudio	Tidak diimplementasikan (tidak spesifik)
24	UI ada yang tidak berguna (yang lebih banyak digunakan assignment, submit, scoreboard, dan hasil submit	Dosen	DAA dan ASD	Ko Claudio	Tidak diimplementasikan (tidak spesifik)
25	Ingin memiliki fungsi reminder. Banyak mahasiswa lupa mengerjakan tugas dan tidak bisa mengumpulkan. Fungsi reminder akan mengirimkan reminder ke email mahasiswa	Dosen	DAA dan ASD	Ko Claudio	Tidak diimplementasikan (terlalu sulit, belum ada scheduler dan sistem pengiriman email)
26	Membatasi soal (deskripsi & PDF) hanya bisa diakses saat assignment “open”, setelah waktu mulai, dan user yang login masuk sebagai “participant”	Dosen	AIF401	Pak Pascal	committed a15daa0
27	Integrasi login ke RADIUS (password sama dengan login Windows)	Dosen	AIF401	Pak Pascal	Eksplorasi dulu
28	Mengumpulkan file TXT	Dosen	AIF401	Pak Pascal	Diimplementasikan (committed a15daa0)
29	Mengumpulkan file JAR (java multi kelas)	Dosen	AIF401	Pak Pascal	Eksplorasi dulu
30	Branding Teknik Informatika	Dosen	AIF401	Pak Pascal	Diimplementasikan (akan di merge)

Fitur-fitur yang akan diimplementasikan yaitu:

- Membatasi soal (deskripsi & PDF) hanya bisa diakses saat *assignment* "open" dan setelah waktu mulai.
- Mengumpulkan file TXT yang berguna untuk *Pre-Test*.
- Menambahkan parameter *Display Name* pada register peserta.
- Menambahkan fitur "*Lock Student's Display Name*" yang berfungsi untuk mencegah para peserta mengganti *Display Name*.
- Menambahkan fitur "*Archived Assignment*" yang berfungsi agar *assisgment* tidak memiliki batasan waktu sehingga *assignment* dapat dikerjakan kapan saja. *Archived Assignment* tidak akan muncul pada kalendar *Dashboard*.
- Branding Teknik Informatika.
- Menambahkan halaman *Hall of Fame* yang berisikan *score global* untuk semua *assignment*.
- Integrasi login ke RADIUS (password sama dengan login Windows).
- Mengumpulkan file JAR (java multi kelas).
- Mengganti *method shell_exec()* menjadi *unlink()*.
- Menambahkan fungsi rekoneksi ke database.

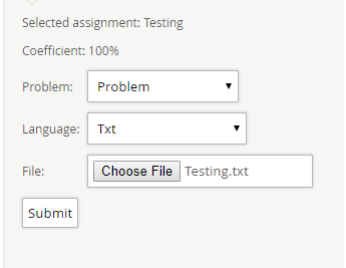
4. Mengimplementasikan fitur terhadap perangkat lunak.
   status : Ada sejak rencana kerja skripsi.
   hasil : Peneliti telah berhasil mengimplementasikan beberapa fitur terhadap perangkat lunak yaitu:

   - Membatasi soal (deskripsi & PDF) hanya bisa diakses saat *assignment* "open" dan setelah waktu mulai.
     Ketika peserta mencoba untuk mengunduh soal (deskripsi & PDF) pada saat *assignment* belum dimulai, maka akan muncul pesan error *Selected assignment has not started.*



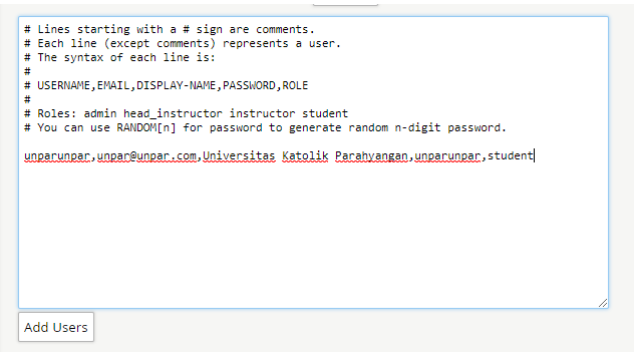
Gambar 3: Pesan error: *Selected assignment has not started.*

   - Mengumpulkan file TXT yang berguna untuk *Pre-Test*.
     Peserta dapat mengumpulkan file dengan ekstensi **.txt*.

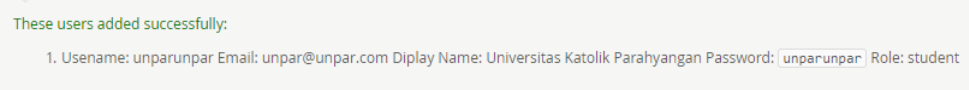


Gambar 4: *Submit file txt*

- Menambahkan parameter *Display Name* pada register peserta. *Admin* dapat mendaftarkan peserta dengan tambahan parameter baru yaitu *Display Name*.

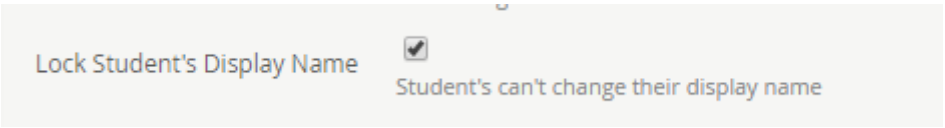


Gambar 5: Mendaftarkan peserta dengan tambahan parameter *Display Name*

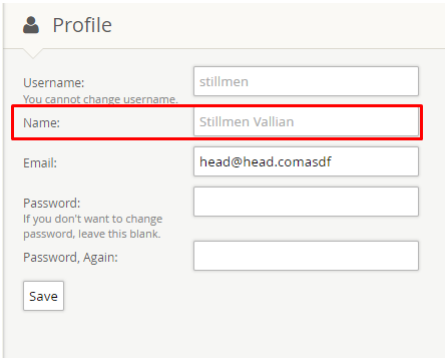


Gambar 6: Pendaftaran peserta berhasil

- Menambahkan fitur "*Lock Student's Display Name*" yang berfungsi untuk mencegah para peserta mengganti *Display Name*.
  Fitur ini terdapat pada halaman *Settings* dan hanya bisa diaktifkan oleh *admin*. Jika fitur *Lock Student's Display Name* diaktifkan maka para peserta tidak dapat mengubah *Display Name*. Perhatikan gambar berikut.

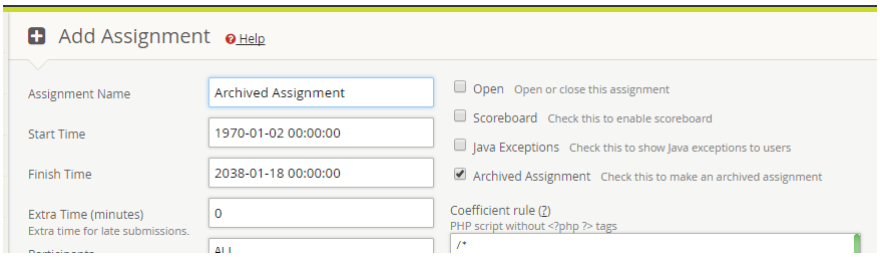


Gambar 7: Fitur *Lock Student's Display Name*

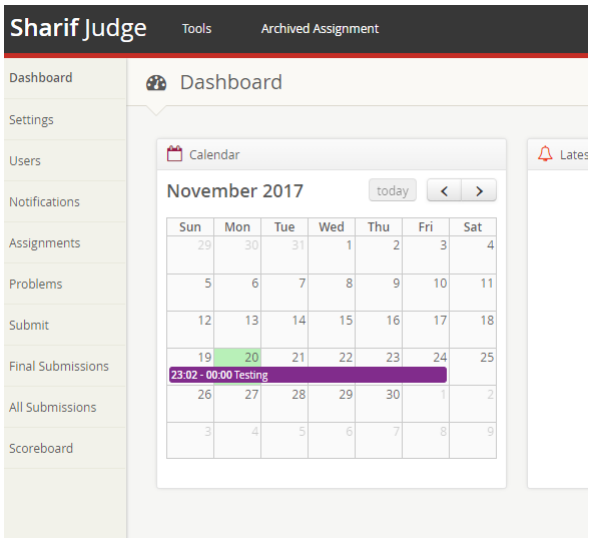


Gambar 8: *Display Name* peserta menjadi *disable*

- Menambahkan fitur "*Archived Assignment*" yang berfungsi agar *assisgment* tidak memiliki batasan waktu sehingga *assignment* dapat dikerjakan kapan saja.
  Fitur ini terdapat pada halaman "*Add Assignment*". *Assignment* yang telah diset menjadi *Archived Assignment* tidak akan muncul pada kalendar halaman pertama. Perhatikan gambar berikut

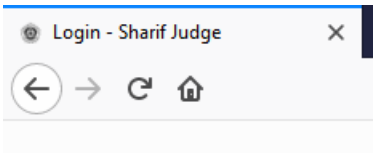


Gambar 9: Fitur *Archived Assignment*

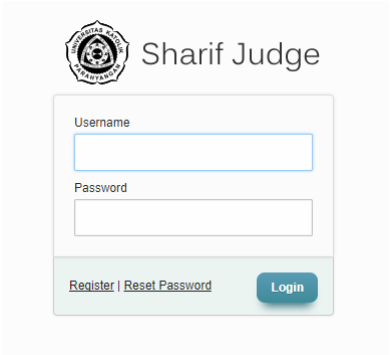


Gambar 10: *Archived Assignment* tidak muncul pada kalendar *dashboard*

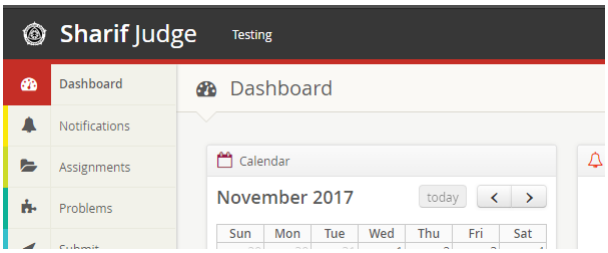
- Branding Teknik Informatika.
  Mengganti logo *Sharif Judge* menjadi logo Universitas Katolik Parahyangan.



Gambar 11: *Icon*



Gambar 12: Halaman *Login*



Gambar 13: *Top Bar*

- Mengganti *method shell_exec()* menjadi *unlink()*.
- Menambahkan fungsi rekoneksi ke database.

5. Mengujikan perangkat lunak ke mata kuliah selama satu semester.
   status : Ada sejak rencana kerja skripsi.
   hasil :

6. Membuat dokumentasi perangkat lunak.
   status : Ada sejak rencana kerja skripsi.
   hasil :

7. Menulis dokumen skripsi.
   status : Ada sejak rencana kerja skripsi.
   hasil : Dokumen skripsi telah ditulis sampai bab 2. Bab 1 berisikan latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi dan sistematika pembahasan. Bab 2 berisikan studi literatur tentang *CodeIgniter* dan *Sharif Judge*.

3 Pencapaian Rencana Kerja

Persentase penyelesaian skripsi sampai dengan dokumen ini dibuat dapat dilihat pada tabel berikut :

1*	2*(%)	3*(%)	4*(%)	5*	6*(%)
1	5	5			-
2	5	5			-
3	5	5			-
4	35	20	15	implementasi fitur sebagian pada Skripsi 1	-
5	15		15		
6	15		15		
7	20	5	15	bagian bab 1 dan bab 2 serta bagian awal analisis di Skripsi 1	-
Total	100	40	60		-

Keterangan (*)
1 : Bagian pengerjaan Skripsi (nomor disesuaikan dengan detail pengerjaan di bagian 5)
2 : Persentase total
3 : Persentase yang akan diselesaikan di Skripsi 1
4 : Persentase yang akan diselesaikan di Skripsi 2
5 : Penjelasan singkat apa yang dilakukan di S1 (Skripsi 1) atau S2 (skripsi 2)
6 : Persentase yang sidah diselesaikan sampai saat ini

4 Kendala yang dihadapi

Kendala - kendala yang dihadapi selama mengerjakan skripsi :

- Terlalu banyak tugas pada mata kuliah lain
- Terlalu banyak godaan berupa hiburan (anime, game, film, jalan-jalan)

Bandung, 21/11/2017

Stillmen Vallian

Menyetujui,

Nama: Pascal Alfadian Nugroho
Pembimbing Tunggal