

PEMUTARAN ULANG KETIKAN MAHASISWA PADA SHARIF JUDGE

ANDREAS RONALDI—6182101026

1 Data Skripsi

Pembimbing utama/tunggal: **Pascal Alfadian, Nugroho, M.Comp.**

Pembimbing pendamping: -

Kode Topik : **PAN5501**

Topik ini sudah dikerjakan selama : **1** semester

Pengambilan pertama kali topik ini pada : Semester **49** - Ganjil **24/25**

Pengambilan pertama kali topik ini di kuliah : **Skripsi 1**

Tipe Laporan : **B** - Dokumen untuk reviewer pada presentasi dan **review Skripsi 1**

2 Latar Belakang

Institusi yang memberikan pendidikan, perlu memiliki cara untuk mengetahui pemahaman pelajarnya. Salah satu caranya adalah dengan memberikan tugas. Tugas merupakan sebuah bentuk penilaian dari pengajar kepada pelajarnya¹. Tugas diberikan kepada pelajar untuk membantu pelajar mendalami materi yang sudah diberikan sebelumnya oleh pengajar dan juga untuk melihat seberapa jauh pemahaman pelajar terhadap materi yang sudah diberikan.

Pada bidang informatika, banyak materi pembelajaran yang dapat diberikan. Salah satu pembelajaran utama dalam bidang informatika adalah keterampilan pemrograman. Dikarenakan itu, perlu sebuah sistem untuk melatih keterampilan pemrograman yaitu dengan memberikan tugas menulis kode program sesuai dengan petunjuk yang diberikan dan program tersebut dapat berjalan sesuai dengan petunjuk². Secara tradisional, tugas ini diberikan dengan cara pengajar menyiapkan dan mendistribusikan tugas tersebut kepada pelajar, kemudian dikumpulkan kembali hasil program pekerjaan pelajar, dan pengajar akan menilai kode program sesuai ketepatan dengan program yang diinginkan secara manual seperti gambar 1. Karena menilaian kode program mencakup keluaran program dan juga analisis kode, maka proses tersebut memakan waktu yang cukup lama untuk dilakukan. Walaupun begitu, cara tradisional ini masih bekerja jika jumlah pelajarnya sedikit. Tetapi semakin banyak kode program yang harus di periksa maka semakin banyak waktu yang dibutuhkan dan semakin banyak pula kesalahan yang berhubungan dengan manusia. Salah satu masalah lain yang muncul juga adalah pelajar tidak dapat mengetahui apakah kode program berada pada jalur yang benar dalam menemukan solusi tugas tersebut.



Abbildung 1: Sistem Tradisional Pemberian Tugas

¹Prihatini, F. N. dan Indudewi, D. (2016) Kesadaran dan Perilaku Plagiarisme dikalangan Mahasiswa(Studi pada Mahasiswa Fakultas Ekonomi Jurusan Akuntansi Universitas Semarang). *Dinamika Sosial Budaya*.

²Demir, ö., Soysal, A., Arslan, A., Yürekli, B., & Yilmazel, Ö. (2010). Automatic grading system for programming home-work. Proceedings of the Annual International Conference on Computer Science Education: Innovation & Technology CSEIT 2010 & Proceedings of the Annual International Conference on Software Engineering SE 2010. https://doi.org/10.5176/978-981-08-7466-7_itcse-19.

Pemberian tugas menulis kode program memiliki banyak masalah. Oleh karena itu, dibutuhkannya sistem baru untuk memberikan tugas kepada pelajar bidang informatika. Sistem baru yang dimaksud tentunya untuk melakukan penilaian secara otomatis. Sebuah sistem yang mengambil kode program pelajar dan memberikan sebuah nilai numerik yang menandakan hasil dari kode program tersebut³. Suatu hal yang menarik, Tugas kode program dapat dibagi menjadi 2 jenis yaitu tugas individu dan tugas kelompok. Pada tugas kelompok merupakan tugas yang ditanggung oleh banyak pelajar, biasanya program yang dibuat memiliki antarmuka dan harus diperiksa oleh pengguna khusus yang mengetahui fitur-fitur yang dibutuhkan. Sedangkan tugas individu merupakan sebuah tugas yang diberikan untuk satu individu, biasanya program yang dibuat bersifat algoritmik dan tidak memerlukan antarmuka untuk dijalankan. Program algoritmik adalah sebuah jenis program yang dibuat berdasarkan algoritma untuk menyelesaikan masalah tertentu. Algoritma sendiri adalah langkah-langkah dalam pemecahan masalah secara sistematis⁴. Algoritma itu seperti resep makanan, dimana akan ada bahan-bahan yang dibutuhkan dan serangkaian langkah untuk membuat suatu makanan yang dijelaskan.

Sebagian besar program yang bersifat algoritmik hanya perlu mengambil *input* dari *input* standar seperti angka, huruf, dan sebuah kata atau kalimat dengan format yang sudah ditentukan, seolah-olah *input* ini merupakan *output* dari program lain. Kemudian program algoritmik akan memproses *input* tersebut dalam komputer dan mengeluarkan hasil komputasinya dalam format yang sudah ditentukan untuk dibaca oleh program lain dan memanfaatkan hasil komputasi tersebut. Singkatnya, program algoritmik itu seperti *filter* antar program. Dengan ini, sistem penilaian secara otomatis dapat dibuat dengan membuat sebuah program yang mengambil kode program, memasukkan *input* sesuai format ke dalam program tersebut, membaca hasil keluaran program, dan menilai hasil keluaran program tersebut⁵. Sistem penilaian otomatis ini diberikan nama *Online Judge*. Terlebih lagi sistem ini dapat dilakukan secara *offline* maupun *online*. Gambar 2 menunjukkan bagaimana *online judge* berintegrasi dengan sistem pemberian tugas yang sudah ada.

Tugas pemrograman sudah menjadi keseharian dalam pembelajaran pada bidang informatika. Termasuk pada perguruan tinggi pada bidang informatika, maka *online judge* menjadi sebuah kebutuhan termasuk pada Universitas Katolik Parahyangan atau yang biasa disebut UNPAR. *Online Judge* yang digunakan oleh UNPAR dinamakan SharIF-Judge⁶ yang merupakan hasil dimodifikasi oleh Stillmen Vallian terhadap Sharif-Judge⁷ buatan Mohammad Javad Naderi yang dibuat menggunakan *framework* CodeIgniter dan Bash. Gambar 3 merupakan halaman utama setelah masuk ke dalam SharIF-Judge.

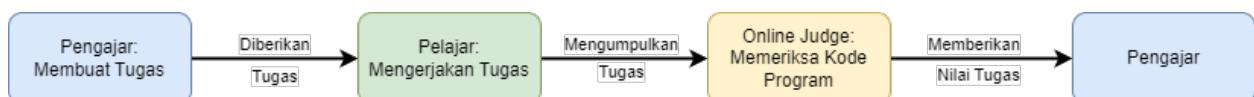


Abbildung 2: Sistem Integrasi oleh *Online Judge*

³Kurnia, A., Lim, A., dan Cheang, B. (2001) Online judge. *Computers & Education*.

⁴IDCloudHost (2020) Algoritma pemrograman beserta contohnya. <https://idcloudhost.com/blog/algoritma-pemrograman-pengertian-fungsi-cara-kerja-dan-contohnya/>. 6 Desember 2024.

⁵Kurnia, A., Lim, A., dan Cheang, B. (2001) Online judge. *Computers & Education*.

⁶Vallian, S. (2018) Kustomisasi Sharif Judge Untuk Kebutuhan Program Studi Teknik Informatika. Skripsi. Universitas Katolik Parahyangan, Indonesia.

⁷Version 1.4 (2014) *Sharif Judge Documentation*. Mohammad Javad Naderi. Tehran, Iran.

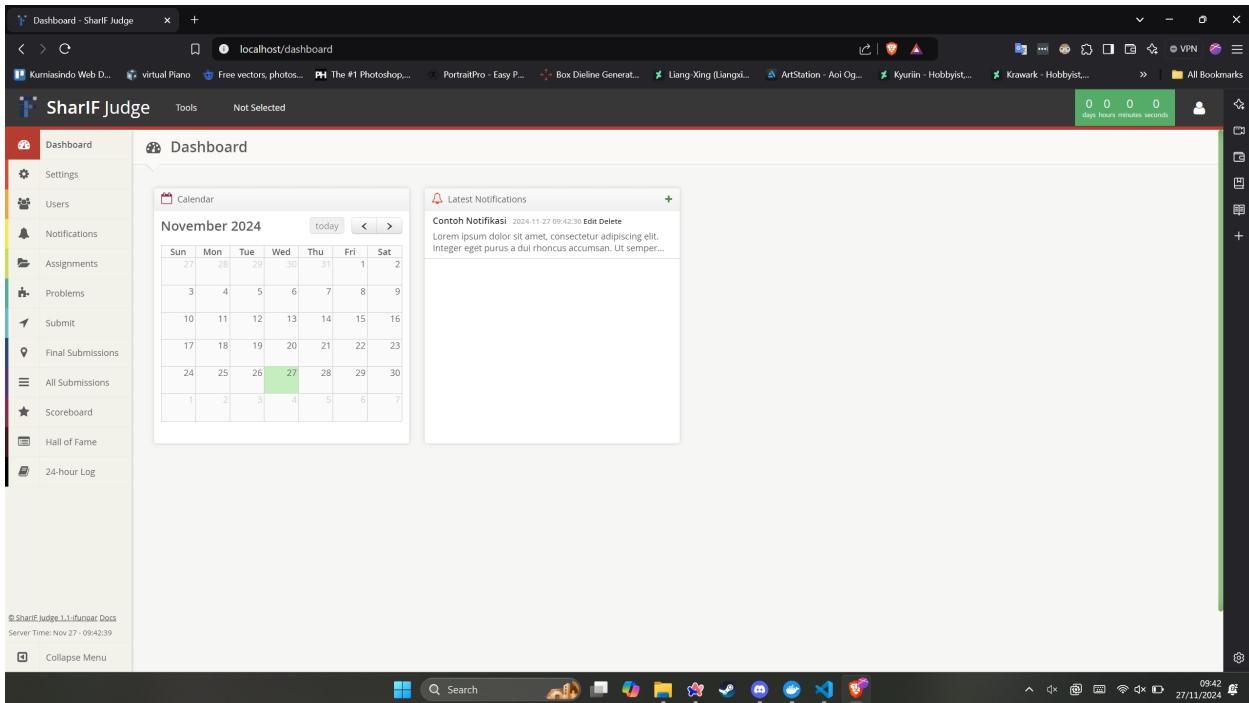


Abbildung 3: Tampilan Awal SharIF Judge

Ujian juga merupakan sebuah bentuk penilaian dari pengajar kepada pelajarnya. Tentunya pelajar maupun mahasiswa ingin memperoleh nilai yang memuaskan dalam ujiannya. Banyak cara yang dilakukan oleh pelajar maupun mahasiswa untuk memperoleh nilai tersebut, salah satunya adalah dengan melakukan kecurangan yaitu *copy paste* atau menyalin jawaban teman atau rekan mereka⁸. Praktek ini diperparah jika ujian dilakukan secara *online*, dikarenakan pelajar dapat mengakses berbagai fasilitas di internet. Oleh karena itu, diperlukan sebuah sistem pada sistem *online judge* untuk mengawasi saat terjadinya ujian online.

Pada saat siswa mengerjakan tugas maupun ujian pembuatan kode program, umumnya penggerjaan kode tersebut dilakukan pada aplikasi eksternal seperti *visual studio code* atau *notepad*. Hal ini juga terjadi pada sistem dalam UNPAR dimana mahasiswa akan membuat kode program pada aplikasi eksternal. Ini membuat pengawasan saat pembuatan kode program lebih sulit untuk dilakukan, terlebih jika ujian dilakukan secara *online*. Maka dari itu, Nicholas Aditya Halim memodifikasi SharIF Judge agar semua sistem pemberian tugas seperti pada gambar 2 dapat dilakukan dalam sistem yang sama yaitu pada SharIF Judge. Sistem yang bangun oleh Nicholas Aditya Halim adalah “Implementasi editor kode pada Sharif Judge”⁹, dimana SharIF Judge ditambahkan sebuah *Integrated Development Environment* atau yang disebut dengan IDE. IDE merupakan sebuah sistem yang memiliki kemampuan untuk membuat kode dalam editor kode dan menjalankan kode program tersebut. Dengan adanya IDE, seluruh proses pembuatan kode program dapat dilakukan dalam SharIF Judge. Maka dari itu, seluruh proses sistem pemberian tugas dapat dilakukan dalam satu sistem saja, yaitu SharIF Judge.

Walaupun begitu, pada dasarnya IDE yang digunakan pada SharIF Judge tidak dapat mengawasi proses pembuatan kode program dan jika terjadinya praktek *copy paste* pada saat pembuatan kode program tersebut. Maka dari itu pada Tugas akhir ini, IDE pada SharIF Judge akan dimodifikasi untuk menangani hal tersebut dengan ditambahkannya fitur untuk merekam semua ketikan atau kejadian dalam editor kode dalam IDE. Lalu ketikan atau kejadian dalam editor dapat di putar kembali seperti rekaman. Fitur ini akan membuat pengawasan terhadap kegiatan kuliah lebih mudah untuk pengawas dan dapat menjadi bukti kecurangan jika dibutuhkan.

⁸Prihatini, F. N. dan Indudewi, D. (2016) Kesadaran dan Perilaku Plagiarisme dikalangan Mahasiswa(Studi pada Mahasiswa Fakultas Ekonomi Jurusan Akuntansi Universitas Semarang). *Dinamika Sosial Budaya*.

⁹Halim, N. A. (2021) Implementasi Editor Kode pada SharIF Judge. Skripsi. Universitas Katolik Parahyangan, Indonesia.

3 Rumusan Masalah

Rumusan Masalah yang akan dibahas pada tugas akhir ini adalah:

1. Bagaimana mengimplementasikan perekaman dan pemutaran ulang ketikan mahasiswa pada IDE SharIF-Judge?
2. Bagaimana cara menyimpan data pemutaran ulang mahasiswa secara rutin dengan otomatis dan tidak mengambil penyimpanan *database* sangat besar?
3. Bagaimana tanggapan pengguna terhadap implementasi perekaman dan pemutaran ulang kode ketikan pada SharIF Judge?

4 Tujuan

Tujuan yang ingin dicapai skripsi ini adalah sebagai berikut:

1. Mengimplementasikan perekaman dan pemutaran ulang ketikan mahasiswa pada IDE SharIF-Judge.
2. Mencari cara penyimpanan data efektif dan mengimplementasikannya pada perekaman dan pemutaran ulang ketikan.
3. Mendapatkan umpan balik dari tanggapan pengguna terhadap perekaman dan pemutaran ulang ketikan mahasiswa pada SharIF-Judge.

5 Detail Perkembangan Pengerjaan Skripsi

Detail bagian pekerjaan tugas akhir sesuai dengan rencana kerja/laporan perkembangan terahir :

1. Melakukan studi mengenai bahasa pemrograman PHP¹⁰.

Status : Ada sejak rencana kerja skripsi.

Hasil : Bahasa pemrograman PHP sudah dipelajari.

Berikut merupakan ringkasan dari hasil pembelajaran bahasa pemrograman PHP.

PHP atau *Hypertext Preprocessor* merupakan sebuah *scripting language* yang dibuat untuk web development¹¹. Awalnya PHP dibuat oleh Denmark-Kanada dan Rasmus Lerdorf pada tahun 1993 dan dirilis pada tahun 1995.

Untuk membuat file php, file akan memiliki extensi php seperti contohnya adalah index.php. Biasanya file PHP akan di proses pada web servernya oleh sebuah PHP *interpreter*. Pada file php, tag <?php menandakan dimana php *interpreter* akan menterjemahkan teks menjadi kode php, sedangkan diluar tag tersebut akan dianggap sebagai HTML biasa. Dalam PHP, echo menandakan dimana PHP interpreter akan mencetak text dimana kutik ke dalam HTML. Berikut contoh kode 1 sederhana dari file php.

Listing 1: Contoh Sederhana File PHP

```

1 Hello
2 <? php
3 echo 'World'
4 ?>

```

Kode 1 akan menghasilkan tulisan Hello World pada websitenya. Sebagai umum bahasa pemrograman, PHP memiliki beberapa fitur yang umum yaitu sebagai berikut:

¹⁰Dokumentasi php. <https://www.php.net/manual/en/>(10 Desember 2024)

¹¹<https://www.php.net/>

- **Comments**

Salah satu hal yang penting dalam bahasa pemrograman adalah untuk memberikan komentar di dalam kodennya. Ada beberapa cara untuk memberikan komentar yaitu sebagai berikut:

Listing 2: Contoh Komentar Satu baris

```
1 // komentar satu baris
2 /* komentar sebagian kode dalam suatu baris */
```

Untuk memberikan komentar untuk beberapa baris, caranya adalah sebagai berikut:

Listing 3: Contoh Komentar Banyak Baris

```
1 /*
2  Ini
3  Sebuah
4  Komentar
5 */
6
7 // atau
8
9 /* Ini
10  * Juga
11  * Sebuah
12  * Komentar
13 */
```

- **Variabel**

Pada PHP, untuk mendefinisikan sebuah variabel dimulai dengan tanda \$, dan diikuti oleh sekumpulan karakter alfabet, nomor, dan simbol _. Nama variabel pada PHP juga *case-sensitive*, yang berarti nama \$text dan \$Text merupakan variabel yang berbeda. PHP juga tidak perlu dan tidak bisa mendeklarasi variabel. Setelah variabel diberikan nilai, PHP akan membuat variabel dengan tipe sesuai dengan nilai yang diberikan. Seperti contoh untuk membuat variabel adalah sebagai berikut:

Listing 4: Contoh Pembuatan Variabel

```
1 $isPerson = TRUE
2 $name = 'Kenzhi'
3 $age = 20
```

PHP juga tidak akan mengeluarkan error ketika variabel dengan tipe angka di tetapkan kembali nilainya seperti \$age = 'remaja'.

Jika ingin mengetahui isi nilai variabel, dapat menggunakan fungsi `var_dump()`. Sebagai contoh untuk mengetahui variabel \$name akan dipanggil `var_dump($name)`.

- **Type**

Bahasa pemrograman juga memiliki tipe-tipe variabel. Tipe-tipe dalam PHP adalah sebagai berikut:

- **bool**: nilai boolean (TRUE atau FALSE)
- **int**: bilangan bulat
- **float**: bilangan ril
- **string**: sebuah teks atau string
- **array**: sebuah array
- **object**: sebuah object
- **null**: nilai yang menandakan nilai yang tidak diberikan

• Operators

Dalam PHP juga memiliki operators seperti umumnya. Penggunaan operator dalam PHP itu men-

gikuti bahasa pemrograman umum, contoh pengguna operator adalah sebagai berikut:

```
{nilai} {operator} {nilai}
```

Dalam bahasa pemrograman PHP, Operator dapat dibagi menjadi 5 bagian yaitu:

– Assignment Operators

Assignment Operator digunakan untuk memberikan nilai kepada variabel. Hanya ada satu *assignment operator* pada PHP yaitu `=`. Penggunaan *assignment operator* berbeda dengan lainnya dikarenakan nilai pertamanya menjadi sebuah *variabel*. Sebagai contoh cara menggunakan *assignment operator* adalah sebagai berikut:

```
$variabel = 'isi variabel'
```

– Arithmetic Operators

Arithmetic Operators digunakan untuk melakukan operasi matematika dasar, seperti penjumlahan, pengurangan, perkalian, dan sebagainya. Pada PHP, ada beberapa *arithmetic operators* yaitu sebagai berikut:

- * `+` : Penjumlahan
- * `-` : Pengurangan
- * `*` : Perkalian
- * `/` : Pembagian
- * `%` : Sisa bagi (modulo)
- * `**` : Perpangkatan

– Comparasion Operators

Comparasion Operators digunakan untuk melakukan perbandingan antara dua nilai yang akan menghasilkan sebuah nilai boolean. Pada PHP, ada beberapa *comparasion operator* yaitu sebagai berikut:

- * `<` : Lebih kecil dari
- * `<=` : Lebih kecil atau sama dengan
- * `>` : Lebih besar dari
- * `>=` : Lebih besar atau sama dengan
- * `==` : Sama dengan (mencek isi nilai sama. `'=='` akan tetap menganggap nilai sama walau-pun memiliki tipe yang berbeda. Seperti contoh `1 == '1'` akan mengembalikan TRUE)
- * `==` : Sama dengan (mencek isi nilai dan tipe nilai yang sesuai. Seperti contoh `1 === '1'` akan mengembalikan FALSE)
- * `!=` : tidak sama dengan (mencek isi nilai, kebalikan dari `'=='`)
- * `!==` : tidak sama dengan (mencek isi nilai dan juga tipe nilai, kebalikan dari `'==='`)

– Logical Operators

Logical Operators digunakan untuk melakukan operasi logika dengan dua atau lebih nilai boolean. Hasil dari *logical operators* adalah sebuah nilai boolean. Pada PHP, ada beberapa *logical operators* yaitu sebagai berikut:

- * `&&` atau `and` : menghasilkan TRUE jika kedua nilai bernilai TRUE
- * `||` atau `or` : menghasilkan TRUE jika salah satu nilai bernilai TRUE
- * `xor` : menghasilkan TRUE jika kedua nilai berbeda.

– Unary Operators

Unary Operators adalah operator yang bekerja dengan hanya satu operand. Operator ini sering digunakan untuk memodifikasi atau mengoperasikan nilai dari satu variabel saja. Pada PHP hanya ada dua *unary operator* yaitu:

- * `++` : Meningkatkan nilai variabel sebesar satu (Penggunaan didepan nilai)
- * `--` : Mengurangi nilai variabel sebesar satu (Penggunaan didepan nilai)
- * `!` : Negasi sebuah nilai boolean (Penggunaan dibelakang nilai)

• Strings

Dalam PHP, String merupakan sebuah tipe. Untuk mendeklarasi sebuah string dalam variabel dapat menggunakan kutip tunggal ('') atau kutip ganda (""). Perbedaan kedua kutip adalah dengan tanda kutip ganda string dapat dimasukkan oleh variabel lain dan juga dapat dimasukkan *escape characters* seperti *new line* (\n) atau tabs (\t). Seperti contohnya adalah

Listing 5: Contoh Pembuatan String

```
1 $test = 'an_example';
2
3 $example = "This_is_$test"; // Keluaran: 'This is an example'
```

Variabel string juga dapat di concat atau digabungkan dengan menggunakan simbol titik (.). Contohnya adalah sebagai berikut:

Listing 6: Contoh Penggabungan String

```
1 $firstName = 'Andreas';
2 $lastName = 'Ronaldi';
3
4 $fullName = $firstName . ' ' . $lastName; // Keluaran: 'Andreas Ronaldi'
```

• Array

Array merupakan sebuah tipe variabel dalam PHP. Array merupakan daftar nilai yang dikelompokkan di bawah satu nama umum. Dalam PHP, isi dalam array tidak harus memiliki tipe yang sama. Untuk mendeklarasi sebuah variabel array dapat menggunakan tanda kurung kotak []. Contoh deklarasi array adalah sebagai berikut:

Listing 7: Contoh mendeklarasi variabel array

```
1 // Mendeklarasi array kosong
2 $array = [];
3 $array = array();
4
5 // Mendeklarasi array dengan isi
6 $array = [1, 'str'];
7 $array = array(1, 'str');
```

Seperti bahasa pemrograman lainnya untuk mengakses sebuah nilai dalam array, dapat menggunakan *index* yang dimulai dengan angka 0. Seperti contoh `$arr` memiliki tiga nilai yaitu ['a', 'b', 'c']. Dapat dilihat nilai 'b' ada pada nilai ke dua. Dikarenakan index dimulai dari angka 0, maka untuk mengakses 'b', Maka untuk mengakses 'b' butuh mengakses index ke satu dengan menggunakan kurung kotak [] setelah nama variabel array, Jadi hasil notasi untuk mengakses 'b' adalah `$arr[1]`.

• Associative Arrays

Dalam array biasa, index merupakan sebuah angka yang terus bertambah yang dimulai dari 0. Tetapi *associative array* menggunakan index yang berbeda-beda seperti menggunakan string atau teks. Index pada *associative array* dinamakan *key*. Berikut contoh mendeklarasi sebuah *associative array*:

Listing 8: Contoh mendeklarasi associative array

```

1 $list = ['first' => 'a', 'second' => 'b'];
2
3 $list['first'] // Keluaran: 'a'
4 $list['second'] // Keluaran: 'b'

```

- **Conditionals**

Dalam PHP, *conditional* digunakan untuk mengeksekusi blok kode tertentu berdasarkan suatu kondisi, biasanya kondisi ini merupakan sebuah boolean. *Conditional* membantu mengontrol jalannya program dengan memungkinkan tindakan yang berbeda untuk diambil tergantung pada kondisi yang diberikan. Berikut notasi penggunaan *conditional* dalam PHP:

Listing 9: Notasi conditional

```

1 if ($condition) {
2     // kode blok akan dijalankan saat $condition merupakan TRUE
3 } elseif ($condition2) {
4     // jika kondisi di atas kondisi elseif ini merupakan FALSE, maka kode blok dapat dijalankan saat
        // $condition2 merupakan TRUE.
5     // elseif dapat dibuat sebanyak yang diinginkan.
6 } else {
7     // jika semua kondisi di atas merupakan FALSE, maka kode blok akan dijalankan.
8 }

```

Kode 9 menunjukkan berbagai notasi *conditional* yaitu **if**, **elseif**, dan **else**. Pernyataan **if** digunakan untuk menjalankan sebuah blok kode jika kondisi tertentu bernilai benar. Pernyataan **elseif** (atau **else if**) berfungsi untuk mencek beberapa kondisi. Jika kondisi dalam **if** salah, PHP akan beralih ke blok **elseif** untuk memeriksa apakah kondisi tersebut benar. Jika benar, kode di dalam blok tersebut akan dijalankan. Jika kondisi tersebut juga salah, program akan beralih ke blok **elseif** berikutnya atau blok **else** jika ada. Jika salah satu kondisi sudah dipenuhi, maka semua kondisi berikutnya tidak ada dijalankan. Kode 10 merupakan contoh penggunaan *conditional* dalam bahasa pemrograman PHP.

Listing 10: Contoh Penggunaan Conditional

```

1 $time = 14;
2
3 if ($time < 12) {
4     echo "Good_morning!"; // karena $time lebih besar dari 12, maka kondisi pertama gagal.
5 } elseif ($time < 18) {
6     echo "Good_afternoon!"; // kode akan dijalankan karena $time masih lebih kecil dari 18.
7 } else {
8     echo "Good_evening!"; // kode tidak dijalankan karena kondisi di atas sudah ada yang memenuhi.
9 }
10
11 // Keluaran: Good afternoon!

```

Seperti bahasa programming yang umum digunakan juga, PHP memiliki notasi **switch**. *Switch* digunakan ketika Anda memiliki banyak kondisi berdasarkan variabel tunggal. Umumnya *switch* lebih mudah dibaca ketika beberapa kondisi **if/elseif** yang memeriksa variabel yang sama. Berikut merupakan notasi penggunaan **switch**:

Listing 11: Notasi switch

```

1 switch (variabel) {
2     case nilai1:
3         // kode blok yang akan dijalankan hingga ditemukan break jika variabel sama dengan nilai1
4         break;
5     default:
6         // kode blok yang akan dijalankan jika tidak ada kasus yang sesuai
7 }

```

Listing 12: Contoh Penggunaan switch

```

1 $day = 3;
2
3 switch ($day) {
4     case 1:
5         echo "Monday";
6         break;
7     case 2:
8         echo "Tuesday";
9         break;
10    case 3:
11        echo "Wednesday";
12        break;
13    default:
14        echo "Invalid_day";
15 }
16
17 // Keluaran: Wednesday

```

Kode 12 merupakan contoh penggunaan **switch** dalam PHP.

- **Loops**

Dalam PHP, loop digunakan untuk menjalankan blok kode secara berulang-ulang selama kondisi yang ditentukan benar. Perulangan membantu mengotomatiskan tugas-tugas yang berulang dan mengurangi kebutuhan untuk menulis kode yang sama beberapa kali. PHP memiliki beberapa jenis perulangan yaitu **while**, **do-while**, **for**, dan **foreach**. Berikut penjelasan untuk setiap jenis perulangan dalam PHP:

- **while**

While adalah perulangan yang paling sederhana. **While** akan terus mengulang jika kondisi tertentu itu benar. Notasi penggunaan **while** adalah sebagai berikut:

Listing 13: Notasi while

```

1 while ($condition) {
2     // kode blok yang berulang.
3 }

```

- **do-while**

do-while mirip dengan **while**, perbedaannya adalah **do-while** akan memeriksa kondisi setelah menjalankan blok kode. Hal tersebut membuat blok kode akan selalu dijalankan setidaknya sekali, meskipun kondisi awalnya salah.

Listing 14: Notasi do-while

```

1 do {
2     // kode blok yang berulang.
3 } while ($condition);

```

- **for**

For mirip dengan **while**, tetapi untuk mendefinisikan variabel yang digunakan dalam kondisi sebelum perulangan, dan untuk menaikkan variabel kondisi tersebut secara manual, semuanya dilakukan pada baris dalam **for**. Kode 15 merupakan notasi untuk menggunakan **for** dalam PHP.

Listing 15: Notasi for

```

1 for (initialization; condition; increment/decrement) {
2     // kode blok yang berulang.
3 }

```

Listing 16: Contoh Penggunaan for

```

1 for ($i = 0; $i < 10; $i++) {
2     echo $i;
3 }
```

Kode 16 merupakan contoh penggunaan **for**. Dimana **\$i** langsung dideklarasikan dalam baris **for**, dan penambahan pada variabel **\$i** juga ada pada baris yang sama.

- **foreach**

Foreach secara khusus digunakan untuk mengulang nilai dalam sebuah array. **Foreach** ini ideal untuk mengulang semua nilai dalam array (baik array biasa maupun associative array) tanpa perlu mengelola indeks atau penghitung secara manual. Untuk notasi penggunaan **foreach** adalah sebagai berikut:

Listing 17: Notasi foreach

```

1 // untuk mendapatkan nilai dalam array maupun associative array
2 foreach ($array as $value) {
3     // Kode yang akan dijalankan (dapat mengakses variabel $value)
4 }
5
6 // untuk mendapatkan nilai beserta key dalam array maupun associative array
7 foreach ($array as $key => $value) {
8     // Kode yang akan dijalankan (dapat mengakses variabel $key dan $value)
9 }
```

- **Functions**

Pada bahasa pemrograman PHP, *function* atau fungsi adalah sebuah blok kode yang dapat digunakan kembali yang dibuat untuk melakukan sebuah tugas tertentu. Fungsi digunakan untuk membantu menyusun kode agar lebih terstruktur dan mengurangi pengulangan kode yang sama.

Listing 18: Notasi fungsi

```

1 function functionName($parameter1, $parameter2, ...) {
2     // Code to be executed
3     return $nilai; // Mengembalikan nilai ke pemanggil fungsi
4 }
```

Kode 18 merupakan notasi untuk membuat fungsi dalam PHP. Dimulai dengan *keyword function*, dilanjutkan dengan nama fungsinya, selanjutnya terdapat kurung ‘()’ yang didalamnya dapat diberikan parameter, dan terakhir diikuti oleh kurung kurawal ‘’. Parameter dalam fungsi itu bersifat opsional, dan digunakan untuk menerima variable dari luar kedalam fungsi tersebut. Tetapi saat fungsi dipanggil jika memiliki parameter maka pemberian paramater untuk fungsi itu bersifat wajib, jika tidak memiliki parameter yang tepat maka php *interpreter* akan mengeluarkan error. Pada kode 18 dalam blok kode, ada sebuah *keyword return* yang berarti fungsi akan mengembalikan nilai **\$nilai** (yang dapat diganti menjadi apapun) ke pemanggil fungsi tersebut. Untuk memanggil sebuah fungsi dalam PHP dapat menggunakan notasi seperti berikut.

```
functionName($parameter1, $parameter2, ...)
```

Dalam fungsi jika parameter dapat memiliki nilai awal, jadi jika saat pemanggil fungsi tidak memiliki parameternya tidak akan terjadi error. Contoh penggunaan fungsi dengan nilai awal adalah sebagai berikut:

Listing 19: Contoh Penggunaan fungsi

```

1 function sapa($name = "Tamu") {
2     echo "Hello, _$name!";
3 }
4
5 sapa("Alice"); // Keluaran: Hello, Alice!
6 sapa();         // Keluaran: Hello, Tamu!
```

Dalam PHP, fungsi tidak dapat mengakses variabel yang terdapat di luar fungsinya itu sendiri kecuali dalam blok kode fungsi ditambahkan deklarasi *keyword global*. Seperti contohnya untuk mengakses variabel \$varGlobal maka harus ditambahkan *global* \$varGlobal sebelum mengakses variabel tersebut dalam fungsi.

Listing 20: Contoh fungsi dengan variabel global

```

1 $varGlobal = "global_variabel";
2
3 function myFunc() {
4     global $varGlobal;
5     echo $varGlobal; // Keluaran: global variabel
6 }
7
8 myFunc();

```

PHP juga menyediakan cara untuk memanggil fungsi menggunakan nama fungsi yang disimpan dalam variabel. Seperti contohnya adalah sebagai berikut:

Listing 21: Contoh pemanggil fungsi menggunakan variabel

```

1 function hello() {
2     echo "Hello,_World!";
3 }
4
5 $namaFungsi = "hello"; // nama fungsi yang dimasukkan kedalam kutik ganda
6 $namaFungsi(); // Keluaran: Hello, World!

```

PHP juga memiliki *anonymous function*, yaitu fungsi yang tidak memiliki nama. Fungsi ini dapat disimpan dalam variabel, diteruskan sebagai argumen, atau digunakan sebagai callback. Notasi penggunaan *anonymous function* adalah sebagai berikut:

Listing 22: Contoh anonymous function

```

1 $myfunction = function() {
2     // Kode blok yang dijalankan
3 };
4 $myfunction();

```

Dalam *anonymous function*, bisa digunakan *keyword use* untuk mendapatkan variable yang di-deklarasikan diluar fungsi. Contoh penggunaan *use* adalah sebagai berikut:

Listing 23: Contoh penggunaan use

```

1 $test = 'test';
2 $myfunction = function() use ($test) {
3     echo $test;
4     return 'ok';
5 };
6 $myfunction(); // Keluaran : test

```

Selanjutnya adalah fungsi bernama *callback* yang merupakan fungsi yang diteruskan sebagai parameter ke dalam fungsi lain. *Callback* sering digunakan dalam event handling atau manipulasi variable array. Contoh penggunaan *callback* adalah sebagai berikut:

Listing 24: Contoh penggunaan callback

```

1 function sayHello($name) {
2     echo "Hello,_$name!";
3 }
4
5 function greetUser($callback) {
6     $callback("Andreas"); // Call the passed callback function
7 }
8
9 greetUser("sayHello"); // Keluaran: Hello, Kenzhi!

```

Terakhir adalah fungsi bernama *arrow function* yang merupakan sebuah *anonymous function* yang hanya memiliki satu ekspresi atau satu baris saja dan langsung mengembalikan ekspresi tersebut secara langsung. Contohnya adalah sebagai berikut:

Listing 25: Contoh penggunaan array function

```
1 $multiply = fn($a, $b) => $a * $b;
2
3 $multiply(2, 4) // Keluaran: 8
```

Parameter dan keluaran dalam fungsi dapat di *assign* sebuah tipe agar tidak terjadi error saat penjalanan kode program contohnya adalah sebagai berikut:

Listing 26: Contoh penggunaan array function

```
1 // tipe dalam parameter
2 function sendEmail(string $to, string $subject, string $body) {
3     //...
4 }
5
6 // tipe untuk keluaran fungsi
7 function sendEmail($to): bool {
8     return true;
9 }
```

- **Class**

Class atau kelas adalah konsep dasar dalam Pemrograman Berorientasi Objek atau OOP. Kelas adalah sebuah *blueprint* untuk sebuah *object*. Dalam kelas, bisa terdapat *attributes* (variabel) dan *methods* (fungsi) yang mendefinisikan suatu object.

Untuk mendeklarasi sebuah kelas dalam PHP, dibutuhkannya *keyword class* diikuti dengan nama kelas tersebut dan tanda kurung kurawal yang dapat diisi oleh *attributes* dan *methods*. Untuk membuat sebuah *attributes* atau *methods* dibutuhkannya *keyword visibility* yang terdiri dari *public*, *protected*, dan *private*. *visibility public* membuat *attributes* atau *methods* dapat akses dimanapun. *visibility protected* hanya dapat diakses oleh kelas tersebut dan kelas yang di *inherited*. *visibility private* hanya dapat diakses oleh kelas tersebut.

Notasi pembuatan kelas adalah sebagai berikut:

Listing 27: Notasi Kelas

```
1 class ClassName {
2     // Attributes
3     public $variable1;
4     private $variable2;
5
6     // Methods
7     public function method1() {
8         // Kode untuk fungsi method1
9     }
10
11    private function method2() {
12        // Kode untuk fungsi method2
13    }
14 }
```

Listing 28: Contoh Kelas

```
1 class Dog {
2     public $name;
3
4     public function bark() {
5         echo $this->name . ' barked!';
6     }
7 }
```

Kode 31 adalah contoh mendeklarasi sebuah kelas. Dimana Dog memiliki sebuah *attributes* \$name dan *methods* bark(). Pada *methods* sebuah kelas untuk mengakses *attributes* dan *methods* lainnya dapat menggunakan *keyword* \$this. *methods* bark() terdapat \$this->name yang akan mengakses *attributes* \$name dalam kelas tersebut.

Untuk memanggil kelas, dibutuhkannya object dengan kelas tersebut. Untuk membuat object dengan kelas tertentu dapat menggunakan *keyword new* di sebelum nama kelas. Setelah itu kelas baru akan di *assign* kedalam sebuah variable. Melalui variable itu, *attributes* dan *methods* kelas tersebut dapat di akses. Untuk memanggil *attributes* atau *methods* dalam variable tersebut, butuh sebuah tanda baca panah (->) yang menandakan php *interpreter* untuk mengakses ke dalam instance object tersebut. Untuk Seperti contoh menggunakan kode 31 untuk membuat kelas baru dan mengakses data didalam kelas tersebut adalah sebagai berikut:

Listing 29: Contoh Pemanggilan Kelas

```
1 $robert = new Dog();
2 $robert->name = 'Robert';
3 $robert->bark(); // Keluaran: Robert barked!
```

Pada saat membuat kelas baru, setelah nama kelas ada tanda kurung seperti pada pemanggil sebuah fungsi. Sebenarnya kelas memiliki method awal bernama __construct yang digunakan untuk membangun object kelas tersebut pada saat object dibuat. Fungsi __construct akan dipanggil dengan parameter yang ada saat object baru dibuat. Berikut perubahan yang akan pada Kode 31 dan Kode 29:

Listing 30: Contoh Kelas

```
1 class Dog {
2     public $name;
3
4     public function __construct($name) {
5         $this->name = $name;
6     }
7
8     public function bark() {
9         echo $this->name . '_barked!';
10    }
11 }
12
13 $robert = new Dog('Robert');
14 $robert->bark(); // Keluaran: Robert barked!
```

Seperti pada fungsi, *attributes* dan *methods* dapat ditambahkan tipe contoh perubahan yang adalah sebagai berikut:

Listing 31: Contoh Kelas

```
1 class Dog {
2     public string $name;
3
4     public function __construct(string $name) {
5         $this->name = $name;
6     }
7
8     public function bark(): string {
9         return $this->name . '_barked!';
10    }
11 }
12
13 $robert = new Dog('Robert');
14 echo $robert->bark(); // Keluaran: Robert barked!
```

- **Exceptions**

Dalam PHP, *Exceptions* dapat ditangani dengan menggunakan `try`, `catch`, dan jika dibutuhkan `finally`. Notasi penggunaan *exception* dalam PHP adalah sebagai berikut:

Listing 32: Notasi Penggunaan Exception

```

1 try {
2     // Kode blok yang mungkin akan mengeluarkan error atau exception
3 } catch (Exception $e) {
4     // Kode blok yang mengatasi jika terjadi the exception
5 } finally {
6     // Kode blok yang pasti akan dijalankan setelah try/catch
7 }

```

2. Melakukan studi literatur mengenai *framework* CodeIgniter 3¹².

Status : Ada sejak rencana kerja skripsi.

Hasil : Sudah melakukan studi mengenai cara kerja *framework* CodeIgniter 3.

Berikut ringkasan hasil studi mengenai *framework* CodeIgniter 3:

CodeIgniter 3 adalah sebuah *framework opensource* untuk mempermudah pengguna dalam membangun sebuah aplikasi *website* menggunakan bahasa PHP. CodeIgniter 3 bertujuan untuk membantu pengguna dalam membangun sebuah aplikasi *website* lebih cepat dengan menyediakan *library* yang beragam dengan fungsi yang umum digunakan dan tampilan dan *logic* yang simpel. Gambar 4 merupakan bagaimana data mengalir pada sistem CodeIgniter.

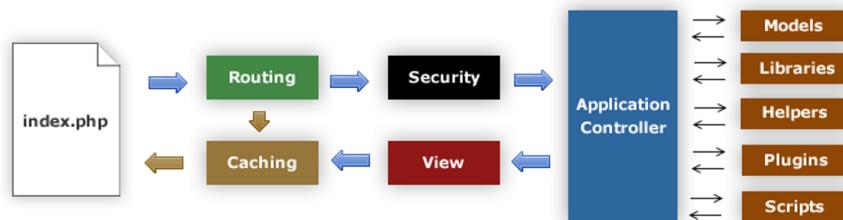


Abbildung 4: *Flow Chart* CodeIgniter

Berikut merupakan penjelasan sederhana dari *flow chart* sistem CodeIgniter 3:

- `index.php` berfungsi sebagai *front controller* yang akan melakukan inisiasi *resource* utama untuk menjalankan CodeIgniter.
 - Router meneliti *request* HTTP dan menentukan apa yang harus dilakukan dengan *request* tersebut.
 - Jika terdapat *file cache*, maka langsung dikirimkan ke *browser* melewati eksekusi sistem yang biasanya.
 - Sebelum *controller* dimuat, seluruh *request* HTTP dan data dari user disaring terlebih dahulu untuk keamanan.
 - Controller* memuat *model*, *library* utama, dan *resource* lainnya yang diperlukan.
 - View* akhir lalu dikirim ke *browser* untuk dilihat. *Cache* akan dibuat terlebih dahulu bila diaktifkan.
- (a) **Model-View-Controller**

¹²Dokumentasi CodeIgniter 3 <https://codeigniter.com/userguide3/> (10 Desember 2024)

CodeIgniter merupakan *framework* berbasis arsitektur Model-View-Controller atau yang selanjutnya akan disebut dengan MVC. MVC adalah pendekatan *software* yang memisahkan *logic* aplikasi dan tampilannya. Pendekatan ini membuat *website* hanya memiliki sedikit *script* karena tampilan *website* terpisah dari *scripting* PHP. Berikut merupakan penjelasan mengenai struktur MVC:

i. Model

Model mewakili struktur data pada sistem untuk mengambil, memasukkan, dan memperbarui data pada *database*. *Model* dapat dibuat dengan membuat sebuah kelas yang mengekstensi `CI_Model` dan diletakkan pada `application/models/`.

Listing 33: Contoh *model*

```

1 class Blog_model extends CI_Model {
2
3     public $title;
4     public $content;
5     public $date;
6
7     public function get_last_ten_entries()
8     {
9         $query = $this->db->get('entries', 10);
10        return $query->result();
11    }
12
13    public function insert_entry()
14    {
15        $this->title    = $_POST['title'];
16        $this->content  = $_POST['content'];
17        $this->date     = time();
18
19        $this->db->insert('entries', $this);
20    }
21
22    public function update_entry()
23    {
24        $this->title    = $_POST['title'];
25        $this->content  = $_POST['content'];
26        $this->date     = time();
27
28        $this->db->update('entries', $this, array('id' => $_POST['id']));
29    }
30}
31

```

Kode 33 merupakan contoh model kelas bernama `Blog_model` pada CodeIgniter. *Model* `Blog_model` dapat mengambil, menambahkan, dan memperbarui *database* bernama ‘`entries`’. File *model* tersebut akan disimpan pada `application/models/Blog_model`. Selanjutnya, pengguna dapat memanggil *Model* tersebut pada *file controller* (akan dijelaskan pada bagian [Controller](#)) untuk memanggil model pada Kode 33 dengan menggunakan *syntax* sebagai berikut:

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

Untuk memanggil *method* yang terdapat pada model tersebut, *syntax* yang digunakan adalah sebagai berikut:

```
$this->Blog_model->get_last_ten_entries();
```

Syntax diatas akan memuat *model* dengan nama `Blog_model` dan akan memanggil *method* `get_last_ten_entries`.

ii. View

View adalah informasi yang akan ditunjukkan kepada user. Biasanya *view* merupakan sebuah halaman web, tetapi pada CodeIgniter, *view* dapat berupa pecahan halaman seperti *header*,

footer, *sidebar*, dan lainnya. Pecahan halaman tersebut dapat dimasukkan secara fleksibel ke dalam *view* lainnya apabila dibutuhkan.

Listing 34: Contoh *view*

```

1 <html>
2 <head>
3     <title>My Blog</title>
4 </head>
5 <body>
6     <h1>Welcome to my Blog!</h1>
7 </body>
8 </html>
```

Kode 34 merupakan contoh dari *file view* pada CodeIgniter. File akan disimpan pada direktori `application/views/`. Untuk dapat diperlihatkan dibutuhkannya penggalian halaman pada *file controller* dengan cara sebagai berikut:

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

Syntax diatas akan mengembalikan halaman *view* dengan nama `name` yang terletak pada direktori `application/views/name.php` dan menampilkannya kepada pengguna.

iii. Controller

Controller adalah bagian utama dari aplikasi CodeIgniter, berfungsi sebagai perantara antara *model*, *view*, dan *resources* lainnya yang dibutuhkan untuk memproses HTTP *request* dan membuat sebuah web page. Kelas *Controller* akan mengekstensi `CI_Controller` dan disimpan pada `application/controllers/`. Contoh *controller* ditunjukkan pada Kode 35.

Listing 35: Contoh *controller*

```

1 <?php
2 class Blog extends CI_Controller {
3
4     public function index()
5     {
6         echo 'Hello_World!';
7     }
8
9     public function comments()
10    {
11        echo 'Look_at_this!';
12    }
13}
```

Kode 35 berfungsi dalam mengembalikan string sesuai dengan fungsi *controller* yang dipanggil. Nama file *controller* pada direktori `application/controllers/blog.php` dan metode diatas akan dijadikan segmen pada URL seperti berikut:

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

URL diatas akan mengembalikan sebuah teks ‘Hello World!’.

Listing 36: Contoh memuat *model* dan menampilkan *view*

```

1 class Blog_controller extends CI_Controller {
2     public function blog()
3     {
4         $this->load->model('blog');
5
6         $data['query'] = $this->blog->get_last_ten_entries();
7
8         $this->load->view('blog', $data);
9     }
10}
```

Pada CodeIgniter, *model* dan *view* hanya dapat dimuat melalui controller. Seperti contoh, Kode 36 akan memuat *model blog* dan mengambil data dari *database*, lalu menampilkan *view* yang memuat data tersebut.

(b) **CodeIgniter URLs**

URL pada CodeIgniter menggunakan *segment-based approach* dibandingkan dengan *query string approach* yang biasanya dipakai. *Segment-based approach* dirancang untuk *search-engine* dan dapat mempermudah pengguna juga. Berikut merupakan contoh dari URL CodeIgniter:

`example.com/news/article/my_article`

Struktur URL pada CodeIgniter juga mengikuti pendekatan MVC (referensi 2a) dan biasanya memiliki struktur sebagai berikut:

`example.com/class/function/ID`

- i. Segmen pertama mewakili kelas *controller* yang ingin dipanggil.
- ii. Segmen berikutnya mewakili fungsi kelas atau *method* yang ingin di panggil.
- iii. Segmen ketiga dan selanjutnya mewakili *identifier* atau pengenal dan variable-variable lain yang akan di kirimkan ke *controller*.

3. Melakukan studi literatur mengenai editor kode Ace¹³.

Status : Ada sejak rencana kerja skripsi.

Hasil : Sudah melakukan studi mengenai editor kode Ace.

Berikut ringkasan hasil studi mengenai editor kode Ace:

Ace merupakan sebuah editor kode yang dapat dimasukkan ke dalam sebuah website yang dibuat menggunakan bahasa *Javasciprt*. Ace memiliki kemampuan dari editor pada umumnya. Berikut merupakan beberapa fitur utama yang dimiliki oleh Ace:

- *Syntax highlighting* untuk bahasa pemrograman.
- Automatic indent dan outdent.
- Kemampuan *cut*, *copy*, dan *paste*.
- Kemampuan *drag and drop* teks menggunakan mouse.
- Banyak *Cursors* dan *selections*
- *Line wrapping*
- *Code folding*

Beberapa kelas penting yang terdapat pada library Ace adalah sebagai berikut:

- **Ace**

Merupakan kelas utama untuk menyiapkan editor kode Ace pada *browser*

- **Editor**

Entri utama untuk fungsionalitas library Ace. Editor sendiri merepresentasikan editor kode yang dibuat pada web. Pada editor sendiri disediakannya *event listener* yaitu sebagai berikut:

- (a) *mouseup* : Mendengarkan saat melepaskan tombol pada tetikus atau *mouse*.

- **EditSession**

Sebuah kelas yang menyimpan semua status dalam editor seperti isi editor, *selection*, dan lain-lain. Kelas ini dinamakan **EditSession**, tetapi untuk mengakses dari editor dinamakan *session*.

Pada kelas *session* ada beberapa *event listener* yaitu:

¹³Dokumentasi Ace. <https://ace.c9.io/>(10 Desember 2024)

- (a) `change` : Mendengarkan perubahan pada isi editor.
- (b) `changeCursor` : Mendengarkan perubahan pada kursor atau *anchor* dalam editor.
- (c) `changeSelection` : Mendengarkan perubahan pemilihan isi kode dalam editor.

- **Anchor**

Menangani posisi *pointer* pada dokumen. Saat teks dimasukkan atau dihapus, posisi *anchor* akan diperbaharui.

- **Document**

Menyimpan teks dokumen.

- **Range**

Kelas ini digunakan di berbagai tempat untuk mengindikasikan suatu wilayah di dalam editor. Kelas ini menyimpan posisi baris awal dan kolom awal, serta baris akhir dan kolom akhir.

- **Selection**

Kelas ini menyimpan posisi yang dipilih oleh pengguna dalam editor.

- **Commands**

Kelas ini digunakan untuk menjalankan perintah pada sebuah editor. Contoh perintah yang sudah ada dalam editor yaitu *insert*, *copy*, *paste*. Pada kelas ini tersedia *event listener* untuk mendengarkan segala *event* yang sudah terjadi bernama *afterExec*.

Listing 37: Contoh kode penggunaan Ace

```

1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <title>ACE in Action</title>
5       <style type="text/css" media="screen">
6           #editor {
7               position: absolute;
8               top: 0;
9               right: 0;
10              bottom: 0;
11              left: 0;
12         }
13     </style>
14     </head>
15     <body>
16
17     <div id="editor">function foo(items) {
18         var x = "All>this>is>syntax>highlighted";
19         return x;
20     }</div>
21
22     <script src="/ace-builds/src-noconflict/ace.js" type="text/javascript" charset="utf-8"></script>
23     <script>
24         var editor = ace.edit("editor");
25         editor.setTheme("ace/theme/monokai");
26         editor.session.setMode("ace/mode/javascript");
27     </script>
28     </body>
29     </html>

```

Kode 37 merupakan cara penggunaan Ace pada sebuah div dengan id `editor`. Ace juga memiliki beberapa konfigurasi, seperti contoh ini yaitu menggunakan tema *monokai* dan menggunakan *syntax highlighting* untuk bahasa pemrograman JavaScript.

4. Melakukan studi literatur mengenai SharIF Judge¹⁴.

Status : baru ditambahkan pada semester ini

Hasil : Sudah mempelajari struktur MVC SharIF Judge dan berhasil di jalankan pada local.

¹⁴Dokumentasi SharIF Judge <https://github.com/ifunpar/SharIF-Judge/blob/docs/readme.md>(10 Desember 2024)

Berikut hasil pembelajaran SharIF Judge:

SharIF Judge merupakan modifikasi dari *open source* bernama Sharif Judge, sebuah website judge gratis dengan kemampuan mengkompilasi bahasa C, C++, Java, dan Python. Sharif Judge dibuat oleh Mohammad Javad Naderi dengan interface web berbahasa PHP menggunakan *framework* CodeIgniter 3 dan BASH¹⁵. Modifikasi dilakukan untuk menambahkan fitur pada Sharif Judge dan juga untuk menyesuaikan sesuai dengan kebutuhan Teknik Informatika UNPAR.

- **Instalasi**

Ada beberapa prasyarat yang diperlukan dalam menjalankan SharIF Judge pada sebuah *server* Linux adalah sebagai berikut:

- *Webserver* dengan PHP versi 5.3 atau lebih dengan `mysqli` extension
- PHP Command Line Interface (CLI)
- *Database* MySQL atau PostgreSQL
- PHP harus memiliki akses untuk menjalankan *shell commands* dengan fungsi `shell_exec`
- Kemampuan untuk mengompilasi dan menjalankan kode yang dikumpulkan (`gcc, g++, javac, java, python2, dan python3`)
- Perl

Setelah perangkat yang sudah memenuhi prasyarat, berikut merupakan cara instalasi SharIF Judge:

- (a) Unduh versi terakhir dari Sharif Judge dan menempatkannya pada direktori publik.
- (b) Pindahkan folder `system` dan `application` ke luar direktori publik. Kemudian simpan alamatnya pada `index.php`.
- (c) Buat sebuah *Database* MySQL atau PostgreSQL.
- (d) Atur pengaturan koneksi *database* pada `application/config/database.php`.
- (e) Atur pengaturan koneksi RADIUS dan SMTP pada `application/config/secrets.php` jika dibutuhkan.
- (f) Atur agar direktori `application/cache/Twig` dapat ditulis oleh php.
- (g) Buka halaman utama SharIF Judge pada *browser* dan ikuti proses instalasi.
- (h) Log in dengan akun admin
- (i) Pindahkan folder `tester` dan `assignments` ke luar direktori publik. Kemudian simpan alamatnya pada halaman pengaturan.

Dikarenakan prasyarat untuk menjalankan SharIF Judge adalah untuk menjalankannya pada sistem operasi linux. Sedangkan perangkat lunak dengan sistem operasi Windows tidak dapat menjalankannya secara langsung. Oleh karena itu, dibutuhkannya sebuah sistem lain untuk menjalankan SharIF Judge. Sistem tersebut adalah *docker*, yang merupakan sebuah wadah atau *container* yang dapat menjalankan sebuah aplikasi dan microservices pada sistem operasi windows, mac, atau linux¹⁶. Untuk menjalankan SharIF Judge dalam *docker* aplikasi yang dijalankan adalah sebuah web service yang menggunakan sistem operasi linux dan menjalankan SharIF Judge. Oleh karena itu, dibutuhkan dua file tambahan untuk mendefinisikan wadah yang diperlukan agar *docker* dapat menjalankan SharIF Judge yaitu `docker-compose.yml` dan `Dockerfile` pada folder *root* SharIF Judge. Isi file tersebut ada pada Kode 38 dan Kode 39.

Listing 38: docker-compose.yml

¹⁵Version 1.4 (2014) *Sharif Judge Documentation*. Mohammad Javad Naderi. Tehran, Iran.

¹⁶Dokumentasi Docker. <https://docs.docker.com/> (10 Desember 2024)

```

1  version: "3"
2  services:
3    codeigniter-3:
4      build: .
5      ports:
6        - "80:80"
7      volumes:
8        - ./var/www/html
9      depends_on:
10        - db
11
12  db:
13    image: mysql:5.7
14    ports:
15      - "3306:3306"
16    environment:
17      MYSQL_DATABASE: judge
18      MYSQL_USER: user
19      MYSQL_PASSWORD: judge
20      MYSQL_ROOT_PASSWORD: root
21    command: --sql_mode=STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,
22      NO_ENGINE_SUBSTITUTION
23    volumes:
24      - db_data:/var/lib/mysql
25
26  phpmyadmin:
27    image: phpmyadmin/phpmyadmin
28    ports:
29      - "8080:80"
30    environment:
31      PMA_HOST: db
32      MYSQL_ROOT_PASSWORD: freehost
33    depends_on:
34      - db
35
36  volumes:
37    db_data:

```

Listing 39: Dockerfile

```

1 # Menggunakan image PHP 7.3 sebagai base image
2 FROM php:7.3-apache
3
4 # Install dependensi dan ekstensi PHP yang dibutuhkan untuk CodeIgniter
5 RUN apt-get update && apt-get install -y \
6     libpng-dev \
7     libjpeg-dev \
8     libfreetype6-dev \
9     zip \
10    unzip \
11    default-jdk \
12    g++ \
13    python2 \
14    python3
15
16 # Install ekstensi GD dan mysqli
17 RUN docker-php-ext-configure gd --with-freetype-dir=/usr/include/ --with-jpeg-dir=/usr/include/ \
18     && docker-php-ext-install gd mysqli
19
20 # Aktifkan mod_rewrite untuk Apache
21 RUN a2enmod rewrite
22
23 # Copy kode CodeIgniter ke dalam container
24 COPY . /var/www/html/
25
26 # Set direktori kerja
27 WORKDIR /var/www/html/
28
29 # Make Folder tester writeable by PHP
30 RUN chmod 777 /var/www/html/restricted/tester
31 RUN chmod 777 /var/www/html/application/cache/Twig

```

```

32
33 # Expose port 80
34 EXPOSE 80
35
36 # Jalankan Apache server
37 CMD ["apache2-foreground"]

```

Pada *root folder* SharIF Judge jalankan *command docker-compose up* dalam sebuah *command prompt* untuk menjalankan SharIF Judge. Pastikan aplikasi *docker* sudah berjalan. Lalu SharIF Judge dapat di akses pada *port* 80 dalam *localhost* (<http://localhost/>).

- **Users**

Pada SharIF Judge, pengguna dibagi menjadi 4 buah *role*. Role yang tersedia adalah sebagai berikut:

- admin*
- head instructor*
- instructor*
- student*

Setiap *role* memiliki akses pada aksi yang berbeda berdasarkan *role*-nya. Tabel 1 merupakan aksi-aksi yang dapat dilakukan untuk setiap pengguna pada SharIF Judge.

Tabelle 1: *Tabel fitur untuk setiap role*

| Aksi | Admin | Head Instructor | Instructor | Student |
|---------------------------------|-------|-----------------|------------|---------|
| Mengubah <i>Settings</i> | ✓ | ✗ | ✗ | ✗ |
| Mengelola Pengguna | ✓ | ✗ | ✗ | ✗ |
| Mengelola <i>Assignment</i> | ✓ | ✓ | ✗ | ✗ |
| Mengelola Notifikasi | ✓ | ✓ | ✗ | ✗ |
| <i>Rejudge</i> | ✓ | ✓ | ✗ | ✗ |
| Mengelola <i>Queue</i> | ✓ | ✓ | ✗ | ✗ |
| Mendeteksi Kode yang Mirip | ✓ | ✓ | ✗ | ✗ |
| Melihat Semua <i>Submission</i> | ✓ | ✓ | ✓ | ✗ |
| Mengunduh Kode Final | ✓ | ✓ | ✓ | ✗ |
| Memilih <i>Assignment</i> | ✓ | ✓ | ✓ | ✓ |
| <i>Submit</i> Kode | ✓ | ✓ | ✓ | ✓ |

- **Penyimpanan Kode Submission**

Pada SharIF Judge, Kode akan disimpan pada lokasi *Assignment* yang dapat di ubah pada halaman *Settings*. Berikut merupakan format penyimpanan sebuah kode:

```
assignment_<a_id>/p_<p_id>/<nama user>/<nama file>-<s_id>. <file ext>
```

Penjelasan untuk format di atas adalah sebagai berikut:

- <*a_id*>
id pada *assignment*.
- <*p_id*>
id pada *problem*.
- <*nama user*>
Nama dari pengguna yang mengumpulkan kode/file.
- <*nama file*>
Nama file yang dikumpulkan, *editor* jika mengumpulkan menggunakan editor kode.
- <*s_id*>
id pada *submission*.

– <file ext>

Extensi file kode yang dikumpulkan.

Sebagai contoh, pengguna bernama `kenzhi` mengumpulkan kode dengan nama file `probA.java` ke dalam *problem* pertama dari *assignment* dengan id 5. `kenzhi` sudah melakukan pengumpulan pada *problem* yang sama sebanyak 5 kali dan *submission* kali ini akan menjadi nomor 6, sehingga *submission id* adalah 6. Maka kode pengguna akan disimpan pada alamat:

```
assignment_5/p_1/kenzhi/probA-6.java
```

• Antrean Penilaian Kode

Pada SharIF Judge, Kode yang dikumpulkan akan di jalankan satu per satu pada antrean menggunakan `bash`. Berikut merupakan cara SharIF Judge menilai kode dari awal pengumpulan pada sistem:

- (a) *Controller Submit* akan menyimpan file pada folder sesuai pada [4](#).
- (b) *Controller Submit* akan memasukkan data *submission* kedalam *model Queue_model*.
- (c) *Model Queue_model* akan menyimpan data *submission* pada *database submission* dan menambahkan data *queue*.
- (d) Selanjutnya *Controller Submit* akan memanggil fungsi `process_the_queue()` yang akan menjalankan fungsi `run()` pada *controller Queueprocess*.
- (e) *Controller Queueprocess* akan menjalankan `tester.sh` pada folder `tester` dengan data dari *queue*.
- (f) `tester.sh` akan menilai kode yang akan dibaca oleh *controller Queueprocess* yang akan menyimpan hasil penilaian.
- (g) Terakhir *Queueprocess* akan menyimpan hasil penilaian pada *database submission* dan menghapus data *queue* menggunakan *Queue_model*.

Dikarenakan SharIF Judge menggunakan *framework* CodeIgniter 3, maka SharIF memiliki arsitektur MVC juga. Berikutnya merupakan hasil pembelajaran Model-View-Controller pada SharIF Judge:

• Model

Hasil pembelajaran MVC akan dimulai dengan *model* yang berada pada direktori `application/models`. Direktori *Model* berisi kelas-kelas yang digunakan untuk mengelola dan mengembalikan data dari *database*. Berikut merupakan file *model* dan penjelasan fungsi-fungsinya yang terdapat pada SharIF Judge:

– `Assignment_model.php`

Model ini digunakan untuk mengelola tabel `assignments` dan mengembalikan informasi yang digunakan dalam halaman *assignment* dan *problem*. Fungsi yang dimiliki adalah sebagai berikut:

- * `add_assignment($id, $edit)`
Menambahkan atau memperbarui sebuah *assignment*.
- * `delete_assignment($assignment_id)`
Menghapus sebuah *assignment*.
- * `all_assignments()`
Mengembalikan daftar semua *assignment* dan informasinya.
- * `new_assignment_id()`
Mendapatkan nomor terkecil dan dapat digunakan sebagai *id assignment* terbaru.
- * `all_problems($assignment_id)`
Mengembalikan daftar semua *problems* dari sebuah *assignment*.

- * `problem_info($assignment_id, $problem_id)`
Mengembalikan semua informasi sebuah *problem*
 - * `assignment_info($assignment_id)`
Mengembalikan semua informasi sebuah *assignment*
 - * `is_participant($participants, $username)`
Mengembalikan sebuah `boolean` yang menyatakan bahwa `$username` terdapat dalam `$participants`.
 - * `increase_total_submits($assignment_id)`
Menambahkan jumlah *total submits* sebanyak satu pada sebuah *assignment*.
 - * `set_moss_time($assignment_id)`
Memperbarui “*Moss Update Time*” pada sebuah *assignment*.
 - * `get_moss_time($assignment_id)`
Mengembalikan “*Moss Update Time*” pada sebuah *assignment*.
 - * `save_problem_description($assignment_id, $problem_id, $text, $type)`
Menambahkan atau memperbarui deskripsi pada sebuah *problem*.
 - * `_update_coefficients($a_id, $extra_time, $finish_time, $new_late_rule)`
Memperbarui koefisien dari sebuah *assignment*.
- `Hof_model.php`
Model ini digunakan untuk mengembalikan informasi yang digunakan dalam *hall of fame* dari tabel `submissions`. Fungsi yang dimiliki adalah sebagai berikut:
- * `get_all_final_submission()`
Mengembalikan seluruh total nilai *final submission* untuk semua *user*.
 - * `get_all_user_assignments($username)`
Mengembalikan nilai *final submission* pada semua problem untuk *user* tertentu.
- `Logs_model.php`
Model ini berfungsi untuk mengelola tabel `logins` dan mengembalikan catatan *login*. Fungsi yang dimiliki adalah sebagai berikut:
- * `insert_to_logs($username, $ip_address)`
Mencatat *login* sebuah *user* dan menghapus catatan jika melebihi 24 jam.
 - * `get_all_logs()`
Mengembalikan semua catatan *login*.
- `Notifications_model.php`
Model ini digunakan untuk mengelola tabel `notifications`. Fungsi yang dimiliki adalah sebagai berikut:
- * `get_all_notifications()`
Mengembalikan semua *notifications*.
 - * `get_latest_notifications()`
Mengembalikan 10 *notifications* terbaru.
 - * `add_notification($title, $text)`
Menambahkan *notification* baru.
 - * `update_notification($id, $title, $text)`
Memperbarui sebuah *notification*.
 - * `delete_notification($id)`
Menghapus sebuah *notification*.
 - * `get_notification($notif_id)`
Mengembalikan sebuah *notification*.

- * `have_new_notification($time)`
Mengembalikan sebuah *boolean* yang menyatakan bahwa terdapatnya *notification* baru.
- `Queue_model.php`
Model ini digunakan untuk mengelola tabel `queue` dan menampilkan data `queue`. Fungsi yang dimiliki adalah sebagai berikut:
 - * `in_queue($username, $assignment, $problem)`
Mengembalikan sebuah *boolean* yang menyatakan bahwa *username* masih memiliki *queue* dalam sebuah *problem*.
 - * `get_queue()`
Mengambil semua *submission queue*.
 - * `empty_queue()`
Menghapus semua *queue*.
 - * `add_to_queue($submit_info)`
Menambahkan sebuah *submission* ke dalam *queue*.
 - * `rejudge($assignment_id, $problem_id)`
Menambahkan seluruh *submissions* dalam sebuah *problem* ke dalam *queue* untuk dinilai ulang.
 - * `rejudge_single($submission)`
Menambahkan sebuah *submission* ke dalam *queue* untuk dinilai ulang.
 - * `get_first_item()`
Mengembalikan *item* pertama dalam tabel `queue`.
 - * `remove_item($username, $assignment, $problem, $submit_id)`
Menghapus sebuah *item* tertentu dalam tabel `queue`.
 - * `save_judge_result_in_db ($submission, $type)`
Menyimpan hasil penilaian *judge* ke dalam *database*.
 - * `add_to_queue_exec($submit_info)`
Menambahkan sebuah *dummy submission* yang digunakan hanya untuk dijalankan ke dalam *queue*.
- `Scoreboard_model.php`
Model ini digunakan untuk mengelola tabel `scoreboard`. Fungsi yang dimiliki adalah sebagai berikut:
 - * `_generate_scoreboard($assignment_id)`
Menghasilkan *scoreboard* untuk sebuah *assignment* dari nilai akhir semua *submission*.
 - * `update_scoreboards()`
Memperbarui *scoreboard* untuk semua *assignment*.
 - * `update_scoreboard($assignment_id)`
Memperbarui *scoreboard* untuk sebuah *assignment*.
 - * `get_scoreboard($assignment_id)`
Mengembalikan *scoreboard* pada sebuah *assignment*.
- `Settings_model.php`
Model ini digunakan untuk mengelola tabel `settings`. Fungsi yang dimiliki adalah sebagai berikut:
 - * `get_setting($key)`
Mengembalikan nilai dari sebuah `$key` pada tabel `settings`.
 - * `set_setting($key, $value)`
Memperbarui nilai dari pada *setting* `$key`.

- * `get_all_settings()`
Mengembalikan seluruh *settings*.
- * `set_settings($settings)`
Memperbarui seluruh nilai perubahan *settings*.
- `Submit_model.php`
Model ini digunakan untuk mengelola tabel `submission`. Fungsi yang dimiliki adalah sebagai berikut:
 - * `get_submission($username, $assignment, $problem, $submit_id)`
Mengembalikan sebuah baris data *submission* tertentu.
 - * `get_final_submissions($a_id, $u_vl, $uname, $p_num, $fil_u, $fil_prob)`
Mengembalikan seluruh *final submission* pada sebuah *assignment*. *User* dengan role *student* hanya dapat melihat *final submission* dirinya sendiri.
 - * `get_all_submissions($a_id, $u_vl, $uname, $p_num, $fil_u, $fil_prob)`
Mengembalikan seluruh *submission* pada sebuah *assignment*. *User* dengan role *student* hanya dapat melihat *submission* dirinya sendiri.
 - * `count_final_submissions($a_id, $u_vl, $uname, $fil_u, $fil_prob)`
Mengembalikan jumlah *final submission* pada sebuah *assignment*.
 - * `count_all_submissions($a_id, $u_vl, $uname, $fil_u, $fil_prob)`
Mengembalikan jumlah *submission* pada sebuah *assignment*.
 - * `set_final_submission($username, $assignment, $problem, $submit_id)`
Memperbarui sebuah *submission* menjadi *final submission*.
 - * `add_upload_only($submit_info)`
Menyimpan hasil *upload only problem* ke dalam tabel *database*.
- `User.php`
Model ini digunakan untuk menyimpan *settings* sebuah *user*. Fungsi yang dimiliki adalah sebagai berikut:
 - * `select_assignment($assignment_id)`
Menyimpan *assignment* yang dipilih oleh *user*.
 - * `save_widget_positions($positions)`
Menyimpan posisi *widget* sebuah *user*.
 - * `get_widget_positions()`
Mendapatkan posisi *widget* sebuah *user*.
- `User_model.php`
Model ini digunakan untuk mengelola tabel `users`. Fungsi yang dimiliki adalah sebagai berikut:
 - * `have_user($username)`
Mengembalikan sebuah *boolean* yang menyatakan `$username` sudah ada pada *database*.
 - * `user_id_to_username($user_id)`
Mengembalikan *username* dari `$user_id`.
 - * `username_to_user_id($username)`
Mengembalikan *user id* dari `$username`.
 - * `have_email($email, $username)`
Mengembalikan sebuah *boolean* yang menyatakan jika *user* memiliki *email* pada *database*.
 - * `add_user($username, $email, $display_name, $password, $role)`
Menambahkan satu *user* baru ke dalam *database*.

```

* add_users($text, $send_mail, $delay)
    Menambahkan banyak user baru ke dalam database.
* delete_user($user_id)
    Menghapus sebuah user dalam database.
* delete_submissions($user_id)
    Mendelete semua submissions yang di submit oleh sebuah user.
* validate_user($username, $password)
    Mengembalikan sebuah boolean yang menyatakan bahwa $password dan $username
* selected_assignment($username)
    Mengembalikan assignment yang dipilih oleh $username.
* get_names()
    Mengembalikan semua display name pada tabel users.
* update_profile($user_id)
    Memperbarui nama, email, password, atau role sebuah user.
* send_password_reset_mail($email)
    Mengirimkan link reset password ke email user yang dapat dipakai selama 1 jam.
* passchange_is_valid($passchange_key)
    Mengembalikan sebuah boolean yang menyatakan bahwa link reset password masih dapat
    dipakai.
* reset_password($passchange_key, $newpassword)
    Memperbarui password dengan divalidasinya password change key.
* get_all_users()
    Mengembalikan seluruh user pada tabel users.
* get_user($user_id)
    Mengembalikan sebuah user yang memiliki id $user_id.
* update_login_time($username)
    Memperbarui catatan login untuk sebuah user.

```

- **View**

View merupakan tampilan yang menjadi perantara antara pengguna dan *sistem*. Pada SharIF Judge, *View* disimpan pada direktori `application/views` dan dibagi menjadi 3 direktori terpisah yaitu:

- **errors**

Pada direktori `errors`, berisi tampilan halaman *error* jika terjadi error pada penggunaan SharIF Judge. Berikut merupakan *view* yang terdapat pada direktori `errors`:

- * `error_404`
- * `error_db`
- * `error_expectation`
- * `error_general`
- * `error_php`

- **pages**

Pada direktori `pages`, berisi tampilan halaman-halaman utama. `pages` juga memiliki dua direktori selain halaman-halama. Berikut merupakan *views* dan direktori yang terdapat pada direktori `pages`:

- * `pages/admin`

Direktori `admin` berisi tampilan halaman khusus untuk *role admin*. Berikut merupakan *views* yang terdapat pada direktori `admin`:

- add_assignment.twig
- add_notification.twig
- add_user.twig
- add_user_result.twig
- delete_assignment.twig
- edit_problem_html.twig
- edit_problem_md.twig
- edit_problem_plain.twig
- install.twig
- logs.twig
- moss.twig
- queue.twig
- rejudge.twig
- settings.twig
- users.twig

* pages/authentication

Direktori *authentication* berisi tampilan halaman khusus untuk *authentication* seperti halaman direktori *Login*. Berikut merupakan *views* yang terdapat pada direktori **admin**:

- login.twig
- lost.twig
- register.twig
- register_success.twig
- reset_password.twig

* assignments.twig

* dashboard.twig

* halloffame.twig

* notification.twig

* problems.twig

* profile.twig

* scoreboard.twig

* scoreboard_tabel.twig

* submissions.twig

* submit.twig

– templates

Pada direktori *templates*, berisikan tampilan yang digunakan berulang oleh halaman utama seperti *header*, *side bar*, dan *base*. Berikut merupakan *views* yang terdapat pada direktori **templates**:

- * base.twig
- * side_bar.twig
- * simple_header.twig
- * top_bar.twig

• Controller

Pada bagian MVC terakhir, terdapat *controller* yang berada pada direktori `application/controller`. Seperti yang dijelaskan pada bagian 2(a)iii, *Controller* digunakan sebagai perantara antara *model*, *view*, dan *resources* lainnya yang dibutuhkan saat membuat sebuah web page. Direktori controller berisi kelas-kelas yang akan mengolah data yang didapat pada *model* dan menyatukan data tersebut ke dalam *views* yang akan ditampilkan kepada pengguna. Pada setiap kelas *controller*, terdapat fungsi `index()` yang menjadi fungsi utama saat kelas di akses oleh pengguna. Berikut merupakan file *controller* dan penjelasan fungsi-fungsinya yang terdapat pada SharIF Judge:

- `Assignments.php`

Berikut fungsi dengan penjelasannya pada *controller* `Assignments.php`:

- * `select()`

Memilih *assignment* yang ditampilkan pada *top bar* menggunakan *ajax request*.

- * `pdf($assignment_id, $problem_id, $no_download)`

Mengunduh *assignment* atau *problem* dalam bentuk *pdf file* ke browser.

- * `downloadtestsdesc($assignment_id)`

Mengunduh dan mencompress data uji dan deskripsi sebuah *assignment*.

- * `download_submissions($type, $assignment_id)`

Mengunduh semua *final submission* pada semua *assignment*.

- * `delete($assignment_id)`

Menghapus sebuah *assignment*.

- * `add()`

Mendapatkan *input* dari pengguna untuk menambah atau memperbarui sebuah *assignment*.

- * `_add()`

Menambahkan atau memperbarui sebuah *assignment*.

- * `edit($assignment_id)`

Menandai *assignment* yang akan di *edit* dan memanggil fungsi `add`.

- * `pdfCheck($assignment_id, $problem_id)`

Melakukan validasi ketersediaan pdf pada sebuah *assignment* atau pada sebuah *problem*.

- * `index()`

Mengambil data dari `Assignment_model` dan menaruh data dan mengembalikan `views assignments.twig`. Gambar 5 menunjukkan hasil halaman Assignment.

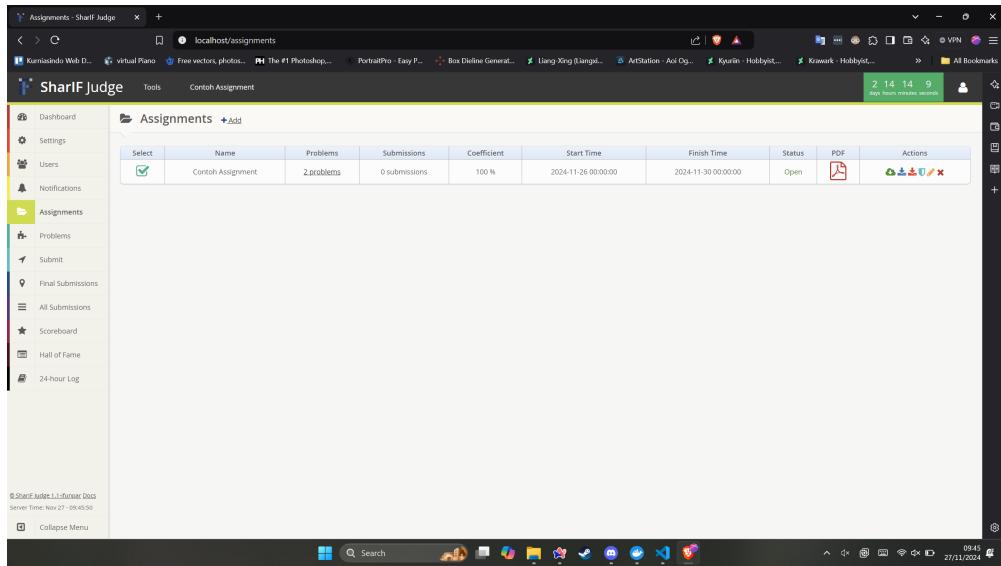


Abbildung 5: Halaman Assignments

– Dashboard.php

Berikut fungsi dengan penjelasannya pada *controller Dashboard.php*:

* *widget_positions()*

Menggunakan *ajax request* untuk menyimpan posisi *widget*.

* *index()*

Mendapatkan data dari beberapa model yaitu **Assignment_model**, **Settings_model**, **User**, dan **Notifications_model**. Data akan dimasukkan ke dalam **dashboard.twig** yang akan dikembalikan ke pengguna. Gambar 6 menunjukkan hasil halaman Dashboard yang dapat diakses oleh semua *role*.

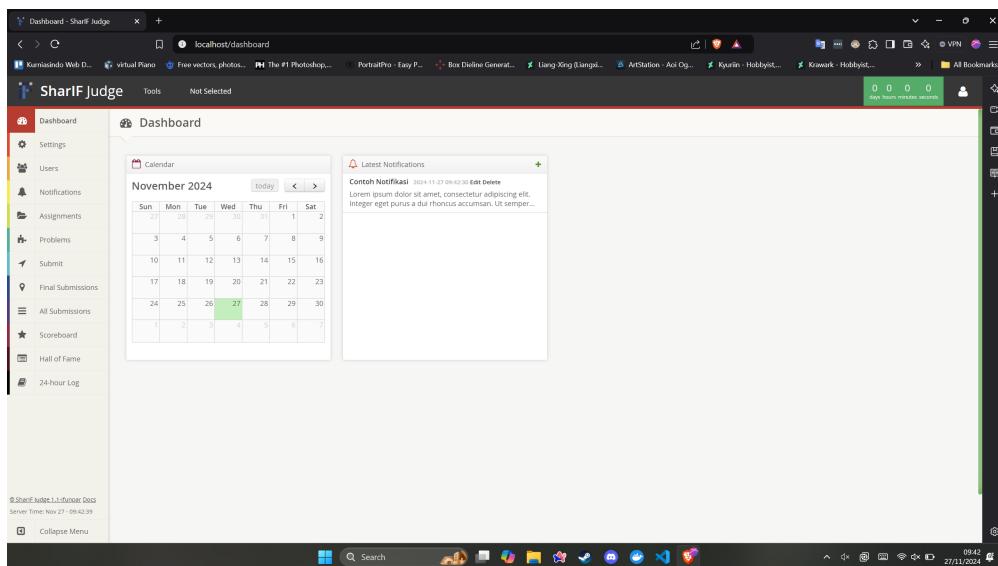


Abbildung 6: Halaman Dashboard

– Halloffame.php

Berikut fungsi dengan penjelasannya pada *controller Halloffame.php*:

* *hof_details()*

Menampilkan nilai akhir semua *problem* dan *assignments* pada sebuah *user*.

* `index()`

Mendapatkan data dari `Hof_model` dan mengembalikan `view halloffame.twig`. Gambar 7 menunjukkan hasil halaman Hall of Fame yang dapat diakses oleh semua `role`.

| Rank | Username | Display Name | Total Score |
|------|----------|--------------|-------------|
| 1 | admin | Admin | 100 |

Abbildung 7: Halaman Hall of Fame

- `Install.php`

Pada `controller Install.php` hanya ada satu fungsi yang menangani pembuatan seluruh tabel pada `database` yang dibutuhkan oleh SharIF Judge. Setelah membuat `database` akan mengembalikan `view install.twig` yang dapat diisi oleh pengguna tentang data `user` dengan role `admin` saat `form` di kirim. Gambar 8 menunjukkan hasil halaman Install.

Abbildung 8: Halaman Install

- `Login.php`

Berikut fungsi dengan penjelasannya pada `controller Login.php`:

* `_registration_code($code)`

Melakukan validasi kode registrasi.

* `register()`

Menunjukkan halaman `register.twig` dan membuat `user` baru.

- * `logout()`

Melakukan *Log out* dan mengalihkan ke halaman `login`.

- * `lost()`

Mengirimkan email *reset password*.

- * `reset($passchange_key)`

Melakukan *reset password* dengan halaman `reset_password.twig`.

- * `index()`

Mengembalikan *view login.twig* dan memeriksa username dan password pada *form* saat di kirim. Gambar 9 menunjukkan hasil halaman Login.

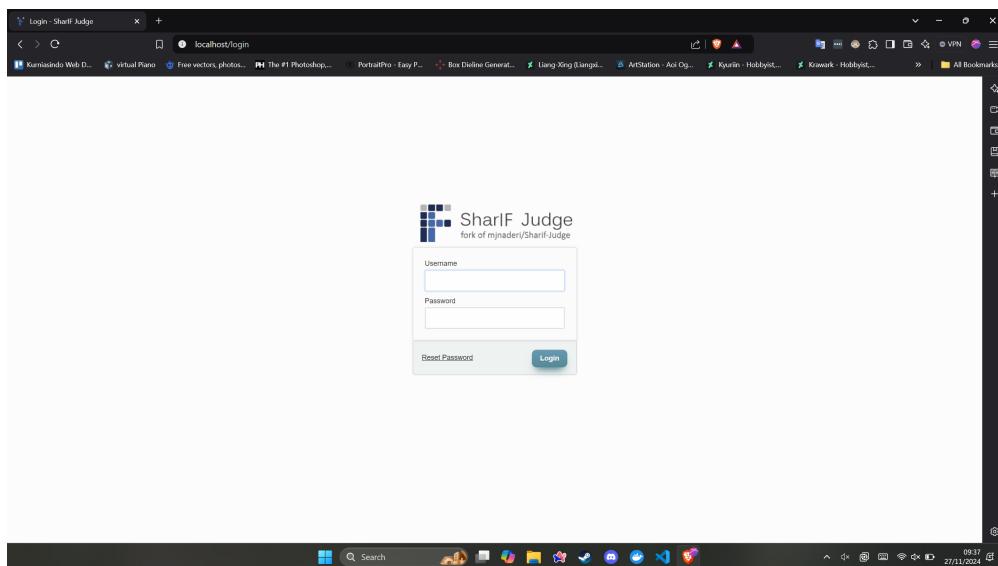


Abbildung 9: Halaman Login

– Logs.php

Pada *controller Logs.php* hanya memiliki satu fungsi yaitu `index()`, dimana fungsi tersebut akan mendapatkan data dari `Logs_model` dan memunculkan halaman `logs.twig`. Gambar 10 menunjukkan halaman Log yang dinamakan halaman 24-Hour Log.

| # | Login ID | Username | IP Address | Login Time | Log from different IP (< 24 hours) |
|---|----------|----------|------------|---------------------|------------------------------------|
| 1 | 2 | admin | 172.20.0.1 | 2024-11-27 02:48:17 | |
| 2 | 1 | admin | 172.20.0.1 | 2024-11-27 02:38:45 | |

Abbildung 10: Halaman 24-Hour Log

– **Moss.php**

Berikut fungsi dengan penjelasannya pada controller *Moss.php*:

* **update(\$assignment_id)**

Memperbaharui *settings* dari masukkan *moss_userid* pengguna.

* **_detect(\$assignment_id)**

Melakukan pemeriksaan kesamaan kode dengan Moss.

* **index()**

Mengambil data dan memasukkannya ke dalam *view moss.twig*. Gambar 11 merupakan hasil halaman moss. Fungsi *_detect* juga akan dijalankan saat *form* terkirim.

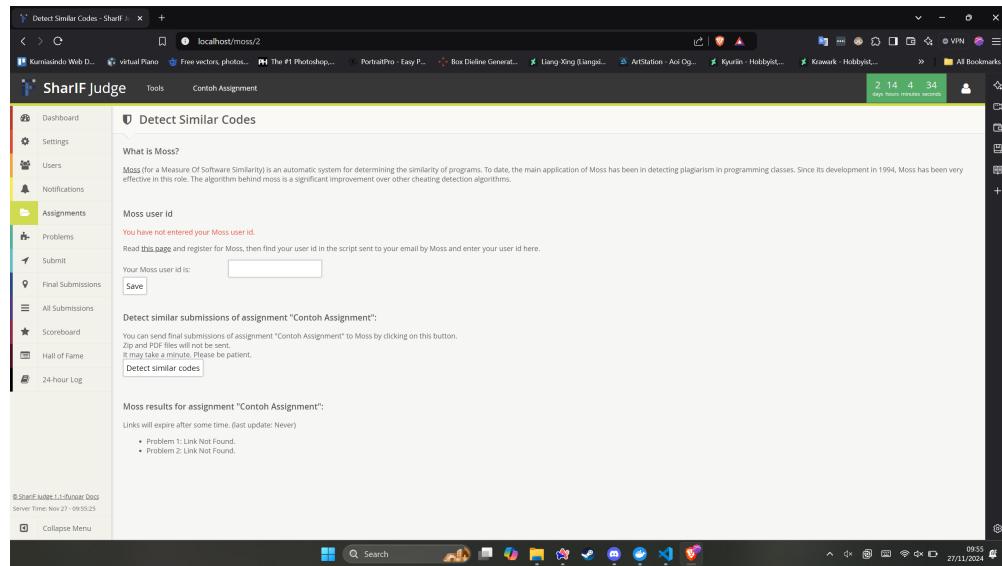


Abbildung 11: Halaman Moss

– **Notifications.php**

Berikut fungsi dengan penjelasannya pada controller *Notifications.php*:

* **add()**

Menambahkan atau memperbaharui sebuah *notification*.

* **edit(\$notif_id)**

Menandai *notification* yang akan di *edit* dan memanggil fungsi *add*.

* **delete()**

Menghapus sebuah *notification*.

* **check()**

Menggunakan *ajax request* untuk mengetahui ketersediaan *notification* baru.

* **index()**

Mendapatkan data dari dua model yaitu *Assignment_model* dan *Notifications_model*.

Data akan dimasukkan ke dalam *view notifications.twig* yang akan dikembalikan ke pengguna. Gambar 12 menunjukkan hasil halaman *Notifications*.

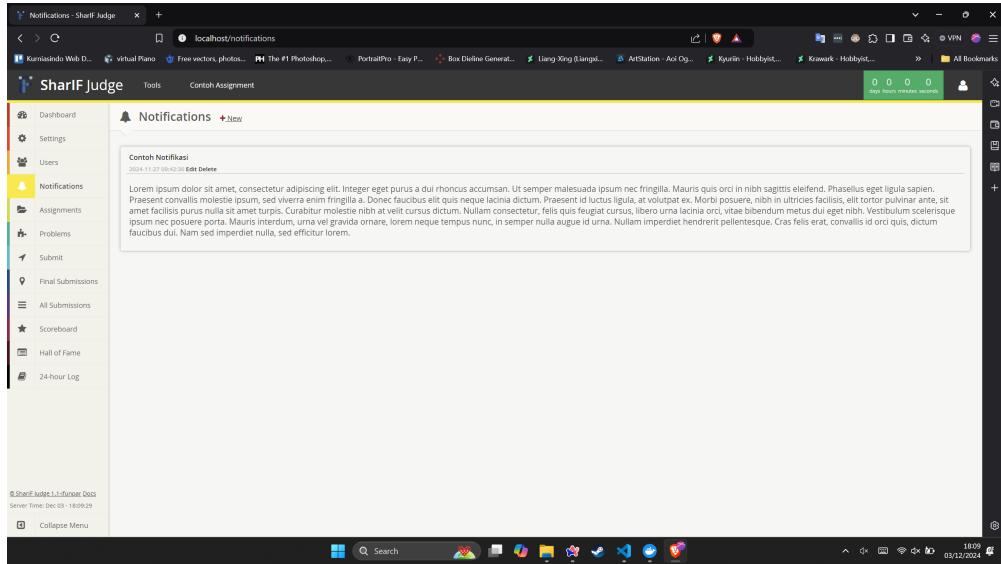


Abbildung 12: Halaman Notifications

– Problems.php

Berikut fungsi dengan penjelasannya pada *controller Notifications.php*:

- * **edit()**

Memperbarui deskripsi sebuah *problem* dalam bentuk *html* atau *markdown*.

- * **index()**

Mendapatkan data *problem* dari berbagai *model* sesuai dengan *assignment* yang dipilih dan menaruh data tersebut pada halaman *problems.twig* yang akan ditampilkan ke pengguna. Gambar 13 menunjukkan hasil halaman Problems.

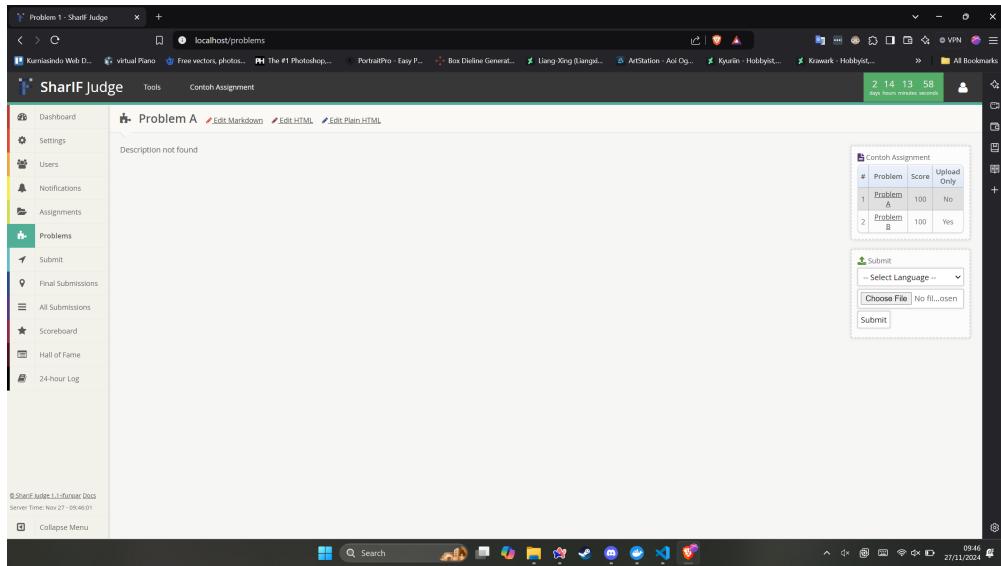


Abbildung 13: Halaman Problems

– Profile.php

Berikut fungsi dengan penjelasannya pada *controller Profile.php*:

- * **_password_check(\$str)**

Melakukan validasi *input password*.

- * **_password_again_check(\$str)**

Melakukan validasi *input* tulisan pengulangan *password*.

- * `_email_check($str)`

Melakukan validasi ketersediaan email pada *database*.

- * `_role_check($str)`

Melakukan validasi *role* pengguna saat ingin mengubah *role user*.

- * `index()`

Mendapatkan data dari berbagai *model* terutama dari *User* yang akan dimasukkan ke dalam *view profile.twig*. Fungsi ini juga menangani pengiriman *form* pembaharuan data *user* pengguna. Gambar 14 menunjukkan hasil halaman Profile.

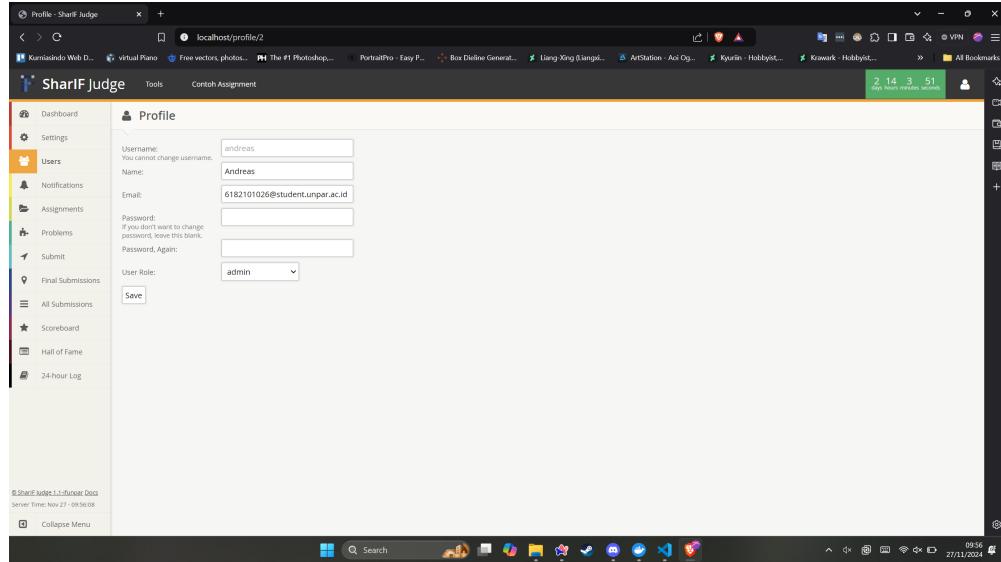


Abbildung 14: Halaman Profile

– `Queue.php`

Berikut fungsi dengan penjelasannya pada *controller Profile.php*:

- * `pause()`

Memberhentikan proses *queue*.

- * `resume()`

Melanjutkan proses *queue*.

- * `empty_queue()`

Menghapus semua *queue* yang ada.

- * `index()`

Mendapatkan data dari *model Queue*, *Assignments_model*, dan *Settings_model* yang dipakai dalam *view queue.twig* dan ditampilkan kepada pengguna. Gambar 15 menunjukkan hasil halaman Queue.

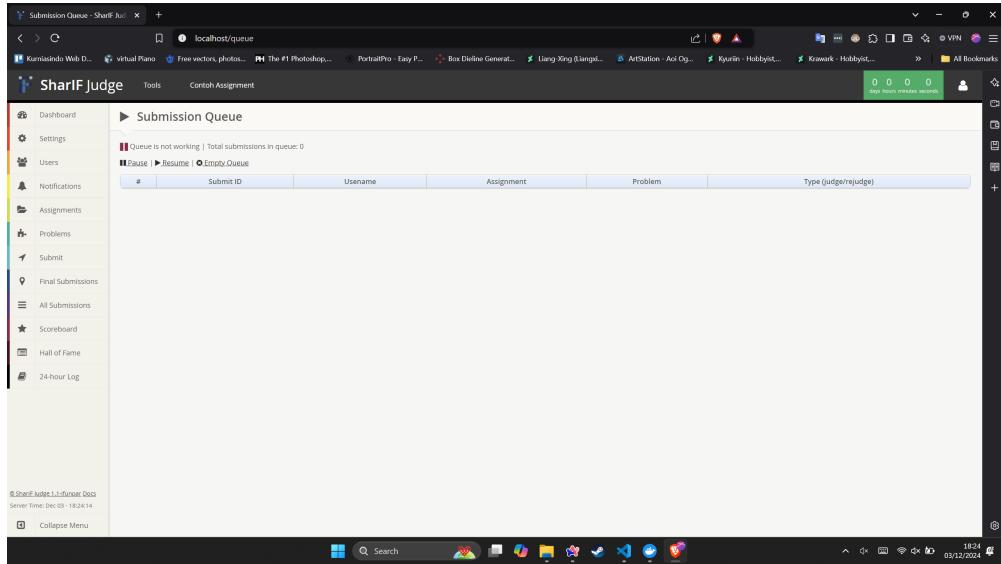


Abbildung 15: Halaman Queue

– Queueprocess.php

Controller Queueprocess.php hanya memiliki satu fungsi yaitu `run()` yang akan menjalankan *queue* satu per satu menggunakan `bash`.

– Rejudge.php

Berikut fungsi dengan penjelasannya pada *controller Profile.php*:

- * `rejudge_single()`

Melakukan *rejudge* untuk satu buah *submission*.

- * `index()`

Mendapatkan data dan menampilkan *view rejudge.twig*. Fungsi ini juga dapat melakukan *rejudge* pada sebuah *problem* tertentu. Gambar 16 menunjukkan halaman Rejudge.

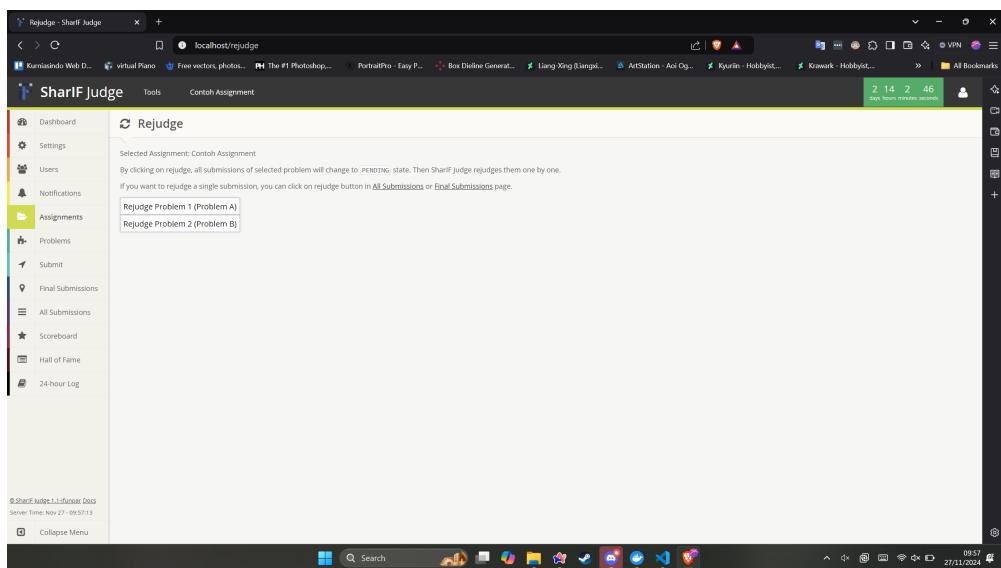


Abbildung 16: Halaman Rejudge

– Scoreboard.php

Controller Queueprocess.php hanya memiliki satu fungsi yaitu `index()` yang akan menampilkan *view scoreboard.twig* dengan data dari *Scoreboard_model*. Gambar 17 menunjukkan hasil halaman Scoreboard.

The screenshot shows the Sharf Judge application's scoreboard interface. On the left is a sidebar with navigation links: Dashboard, Settings, Users, Notifications, Assignments, Problems, Submit, Final Submissions, All Submissions, Scoreboard (which is selected), Hall of Fame, and 24-hour Log. The main content area is titled 'Scoreboard' and displays a table for 'Contoh Assignment'. The table has columns for #, Username, Name, Problem_A, Problem_B, and Total. One row is shown for 'admin' with values: #1, admin, Admin, 100, 100, and Total 200. At the bottom left of the sidebar, it says '© Sharf Judge 1.1-funyear Devs' and 'Server Time: Nov 27 - 09:54:41'. At the bottom right, there are system status icons.

Abbildung 17: Halaman Scoreboard

– `Server_time.php`

Controller Queueprocess.php hanya memiliki satu fungsi yaitu `index()` yang akan mencetak waktu pada *server*, waktu akan digunakan untuk sinkronisasi waktu.

– `Settings.php`

Berikut fungsi dengan penjelasannya pada *controller Settings.php*:

* `update()`

Memperbarui *settings* dari masukkan pengguna.

* `index()`

Mendapatkan data dari *Settings_model* dan menampilkan *view settings.twig*. Jika terdapat *error setting* pada sistem, akan ditampilkan juga pada *view* tersebut. Gambar 18 menunjukkan hasil halaman *Users*.

The screenshot shows the Sharf Judge application's settings interface. On the left is a sidebar with navigation links: Dashboard, Settings (which is selected), Users, Notifications, Assignments, Problems, Submit, Final Submissions, All Submissions, Scoreboard, Hall of Fame, and 24-hour Log. The main content area is titled 'Settings' and contains several configuration fields. These include: 'Timezone' set to 'Asia/Jakarta', 'Week Start Day' set to 'Sunday', 'Full Path to tester' set to '/var/www/html/restricted/tester', 'Full Path to assignments' set to '/var/www/html/restricted/assignment', 'Upload Size Limit (KB)' set to '50', 'Output Size Limit (KB)' set to '1024', 'Results Per Page' set to '40', 'Results Per Page In "Final Submissions"' set to '80', 'Registration' checkbox checked, 'Registration Code' set to '0', 'Log' checkbox checked, 'Lock Student's Display Name' checkbox unchecked, and 'Default Coefficient Rule' set to 'PHP script without <?php ?> tags'. At the bottom left of the sidebar, it says '© Sharf Judge 1.1-funyear Devs' and 'Server Time: Nov 27 - 09:39:09'. At the bottom right, there are system status icons.

Abbildung 18: Halaman Settings

– `Submissions.php`

Berikut fungsi dengan penjelasannya pada *controller Submissions.php*:

* `_download_excel($view)`

Menggunakan *library* PHPExcel untuk membuat sebuah *file excel* dari *submissions* yang akan diunduh pengguna.

* `final_excel()`

Menggunakan fungsi `_download_excel` untuk mendownload *final submission*.

* `all_excel()`

Menggunakan fungsi `_download_excel` untuk mendownload seluruh *submission*.

* `select()`

Menggunakan *ajax request* untuk memilih *submission* yang akan dikumpulkan atau menjadi *final*.

* `_check_type($type)`

Melakukan validasi tipe *submission* yang dikumpulkan.

* `view_code()`

Digunakan untuk melihat kode, melihat hasil kode, atau melihat *log* sebuah *submission*.

* `download_file()`

Mengunduh *file* kode sebuah *submission*.

* `the_final()`

Mendapatkan data dari `Submit_model` untuk mendapatkan *final submission* dan menampilkan halaman `submission.twig` berisi *final submission*. Gambar 19 menunjukkan halaman Final Submissions .

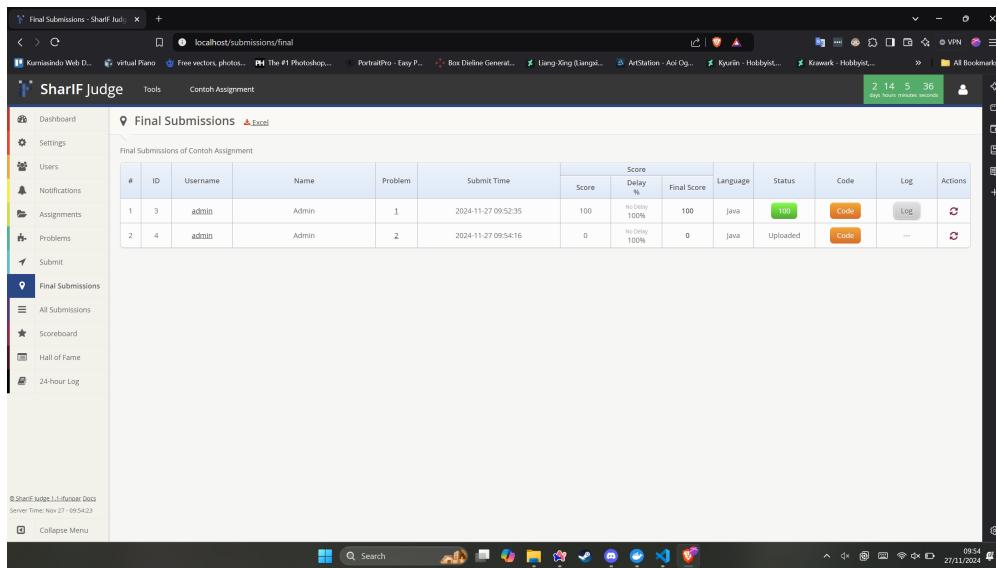


Abbildung 19: Halaman Final Submissions

* `all()`

Mendapatkan data dari `Submit_model` untuk mendapatkan seluruh *submission* dan menampilkan halaman `submission.twig` berisi semua *submission*. Gambar 20 menunjukkan halaman All Submissions .

Abbildung 20: Halaman All Submissions

– Submit.php

Berikut fungsi dengan penjelasannya pada *controller Submit.php*:

- * `_language_to_type($language)`
Mengembalikan kode singkat dari `$language` dipilih.
- * `_language_to_ext($language)`
Mengembalikan extensi file dari `$language` yang dipilih.
- * `_match($type, $extension)`
Melakukan validasi untuk `$type` dan `$extension` agar sesuai.
- * `_check_language($str)`
Melakukan validasi sudah dipilihannya bahasa.
- * `_upload()`
Menyimpan jawaban pengguna yang dikirim dan menambahkannya ke dalam *queue* untuk dinilai jika bukan *upload only problem*.
- * `load($problem_id)`
Mendapatkan isi file dan menaruh isi file ke editor kode.
- * `save($type)`
Menyimpan isi editor kode ke dalam *server* dan menjalankan atau mengumpulkan jika diinginkan.
- * `_submit($data, $problem_id, $language, $user_dir)`
Menambahkan kode ke dalam *submission* untuk dinilai.
- * `_execute($data, $problem_id, $language, $user_dir)`
Menambahkan kode ke dalam *queue* untuk di jalankan saja.
- * `get_output($problem_id)`
Mendapatkan keluaran dari kode yang telah dijalankan sebagai hasil eksekusi.
- * `index()`
Mendapatkan data dari *model Assignments_model* untuk mendapatkan *problem* dan data lainnya. Semua data akan dimasukkan dalam *view submit.twig*. Gambar 21 menunjukkan hasil halaman Submit. Halaman ini terdapat editor kode yang sudah di implementasikan [?].

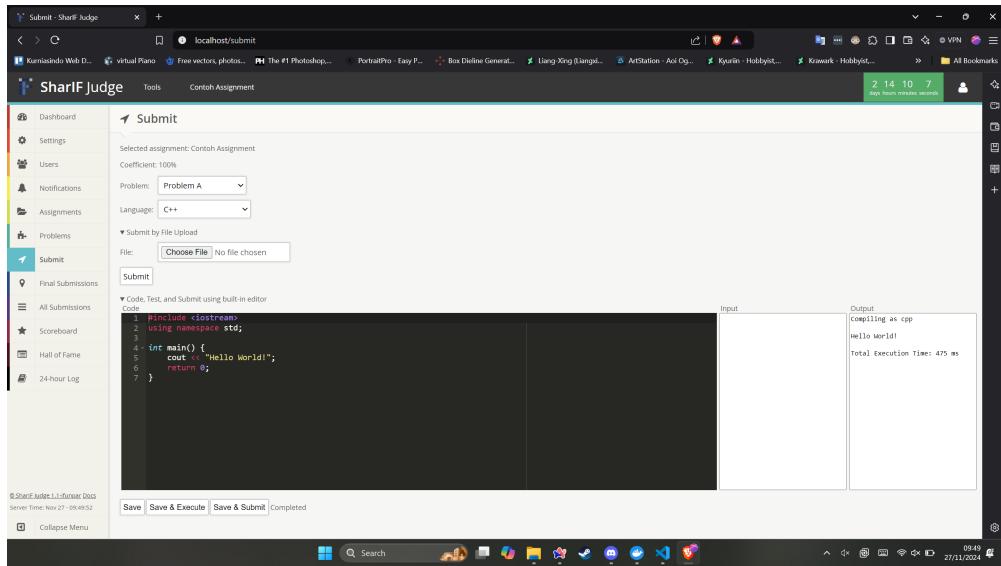


Abbildung 21: Halaman Submit

– User.php

Berikut fungsi dengan penjelasannya pada *controller* User.php:

* `add()`

Menambahkan *user* baru ke dalam *sistem*.

* `delete()`

Menghapus sebuah *user*.

* `delete_submissions()`

Menghapus seluruh *submissions* dari sebuah *user*.

* `list_excel()`

Menggunakan *library* PHPExcel untuk membuat sebuah file excel dari seluruh daftar *user* yang akan diunduh pengguna.

* `index()`

Mendapatkan data dari `User_model` dan menunjukkan `view users.twig`. Gambar 22 menunjukkan hasil halaman Users. Pada halaman ini terdapat daftar seluruh *user* yang terdaftar pada SharIF Judge. Pengguna dapat membuat, memperbarui, dan menghapus *user*.

| | User ID | Username | Display Name | Email | Role | First Login | Last Login | Actions |
|---|---------|----------|--------------|--------------------------------|-------|---------------------|---------------------|---------|
| 1 | 1 | admin | Admin | admin@unpar.ac.id | admin | 2024-11-27 09:38:45 | 2024-11-27 09:38:45 | |
| 2 | 2 | andreas | Andreas | 6182101026@student.unpar.ac.id | admin | Never | Never | |

Abbildung 22: Halaman Users

5. Melakukan studi literatur mengenai Twig¹⁷.

Status : baru ditambahkan pada semester ini

Hasil : Sudah melakukan studi mengenai menggunakan Twig.

Berikut merupakan hasil studi penggunaan Twig:

Twig merupakan sebuah *template engine* untuk PHP. Ada beberapa *expression*, *expression*, atau *statement* yang ditemukan pada template Twig adalah sebagai berikut:

- Pewarisan *Template*
- Struktur Kontrol (menggunakan kondisional, *looping*)
- Filter
- Variable pada PHP

Pada saat template dievaluasi, semua *variable* atau *expression* akan dibuang menjadi value dan *tag* yang mengontrol logika template.

Listing 40: Contoh template Twig

```

1  {% extends "base.html" %} 
2  {% block navigation %} 
3      <ul id="navigation"> 
4          {% for item in navigation %} 
5              <li> 
6                  <a href="{{item.href}}> 
7                      {% if item.level == 2 %}&nbsp;&nbsp;{% endif %} 
8                      {{ item.caption|upper }} 
9                  </a> 
10             </li> 
11         {% endfor %} 
12     </ul> 
13  {% endblock navigation %}
```

Kode 40 merupakan contoh sebuah template Twig. Terdapat dua jenis *delimiter*, yaitu `{% ... %}` dan `{{ ... }}`. *Delimiter* `{% ... %}` digunakan untuk menjalankan sebuah *statement* seperti *for-loops*, sedangkan *delimiter* `{{ ... }}` digunakan untuk mengubah sebuah *variable* atau *expression* menjadi nilai sesungguhnya.

¹⁷Dokumentasi twig. <https://twig.symfony.com/doc/3.x/> (10 Desember 2024)

6. Melakukan studi mengenai cara penyimpanan rekaman ketikan.

Status : Ada sejak rencana kerja skripsi.

Hasil : Sudah melakukan sebagian studi mengenai penyimpanan rekaman ketikan.

Berikut merupakan kesimpulan yang didapat dalam penyimpanan rekaman ketikan.

Sebelum ketikan dapat di putar kembali, dibutuhkannya fitur untuk merekam segala *event* yang terjadi pada masukkan tersebut. *Event* yang sudah direkam akan disimpan bersama dengan waktu saat di terjadinya *event* tersebut. Penyimpanan rekaman juga akan disimpan pada folder yang sama dengan penyimpanan kode submission seperti yang dijelaskan pada bagian 4 dengan nama file **recording**. Pemimpanan perekaman akan memiliki format sebagai berikut:

```
<timeline>: {event: <event>, data: <payload>}
```

<timeline> akan menunjukkan pada milidetik berapa *event* terjadi dengan menggunakan fungsi **Date.now()** pada *Javascript*. sedangkan <event> dan <payload> merupakan data *event* yang terjadi. <payload> akan disesuaikan dengan *event* yang terjadi. Sebagai contoh untuk *event insert* huruf ‘a’ akan di tuliskan menjadi sebagai berikut:

```
1733335637486: {event: "insert", data: "a"}
```

Penyimpanan *event* akan di lakukan oleh *Controller Submit* yang akan menyimpan *file* pada saat kode disimpan. Maka fungsi yang akan dimodifikasi adalah fungsi **save()**.

Untuk tugas akhir 2 akan melakukan studi mengenai tipe file yang akan digunakan seperti JSON, BSON, atau Protocol Buffer. Hal ini tidak dapat diselesaikan pada tugas akhir ini karena pada tugas akhir 1 memfokuskan analisis mengenai cara kerja berbagai *framework* yang terdapat SharIF Judge serta SharIF Judgenya juga.

7. Memodelkan dan merencanakan perubahan pada structur website dan database pada SharIF-Judge.

Status : Ada sejak rencana kerja skripsi.

Hasil : Sudah melakukan perencanaan perubahan pada structur website dan database.

Berikut perencanaan yang didapat:

Pada SharIF Judge akan ditambahkan dalam beberapa file yaitu file **shj_submit.js** yang mengatasi bagian *client side* untuk mengotomatisir rekaman *event*. Selanjutnya akan ada perubahan yang dilakukan saat editor kode di save, maka file yang berubah adalah **shj_submit.js**, *Controller Submit.php*, dan *Model Submit_model.php*.

Pada sistem perekaman ketikan dibutuhkan penyimpanan untuk list recording yang tersedia dalam sistem dengan menyimpan ke dalam database. Maka dibutuhkan tabel baru dengan nama **recording** yang memiliki beberapa *fields* yaitu:

- (a) **id**
- (b) **username**
- (c) **assignment**
- (d) **problem**
- (e) **time**
- (f) **file_name**

Pada SharIF Judge dibutuhkannya juga beberapa hal baru yaitu *Controller Recording.php*, *View recording.twig*, *Model Recording_model.php*, dan *Javascript shj_recording.js*.

Controller Recording.php akan memiliki beberapa fungsi utama yaitu:

- **index()**
fungsi ini akan mengambil list recording sesuai dengan *assignment problem* yang dipilih dan mengembalikan *view recording.twig*
- **load_recording()**
fungsi ini akan digunakan untuk mengambil file yang sudah di simpan pada server dan mengembalikannya.

Untuk *model Recording_model.php* akan memiliki beberapa fungsi utama yaitu:

- **save_recording()**
fungsi ini akan digunakan untuk menyimpan file recoding pada *database recoding*.
- **delete_recording()**
fungsi ini akan digunakan untuk menghapus file recoding pada *database recoding*.
- **get_list_recording()**
fungsi ini akan digunakan untuk mengembalikan *list recoding* pada *database*.

Javascipt yang akan dibuat pada *shj_recording.js* akan memiliki berbagai fungsi yaitu sebagai berikut:

- Fungsi untuk memilih *user* dan *problem* yang ditampilkan.
- Fungsi ajax untuk meminta *file recording* sesuai dengan pilihan.
- Fungsi untuk menjalankan, memberhentikan *recording* dari *file JSON*.

Untuk tugas akhir 2 akan memodelkan perubahan pada SharIF Judge lebih lanjut. Hal ini tidak dapat diselesaikan pada tugas akhir ini karena pada tugas akhir 1 memfokuskan analisis mengenai cara kerja berbagai *framework* yang terdapat SharIF Judge serta SharIF Judgenya juga.

8. Menulis dokumen skripsi

Status : Ada sejak rencana kerja skripsi.

Hasil : Sudah menulis sebagian dokumen skripsi yaitu bab 1, 2, dan 3.

9. Implementasi sistem pemutaran ketikan ulang

Status : Ada sejak rencana kerja skripsi.

Hasil : Akan dikerjakan pada tugas akhir 2.

10. Melakukan Pengujian dan eksperimen

Status : Ada sejak rencana kerja skripsi.

Hasil : Akan dikerjakan pada tugas akhir 2.

6 Pencapaian Rencana Kerja

Langkah-langkah kerja yang berhasil diselesaikan dalam Skripsi 1 ini adalah sebagai berikut:

1. Mempelajari bahasa pemrograman PHP
2. Melakukan studi literatur mengenai *CodeIgniter 3*, editor kode Ace, SharIF Judge, cara penyimpanan rekaman ketikan.

3. Menulis dokumen skripsi yaitu bab 1, 2, dan 3.

Bandung, 05/12/2024

Andreas Ronaldi

Menyetujui,

Nama: Pascal Alfadian, Nugroho, M.Comp.

Pembimbing Tunggal