SKRIPSI

DUKUNGAN BAHASA JAVASCRIPT PADA SHARIF JUDGE

Edwin Pranajaya

NPM: 2017730027

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2022

DAFTAR ISI

Daftar Isi iii

Daftar Gambar v

1 Pendahuluan 1
- 1.1 Latar Belakang . . . 1
- 1.2 Rumusan Masalah . . . 2
- 1.3 Tujuan . . . 2
- 1.4 Batasan Masalah . . . 2
- 1.5 Metodologi . . . 2
- 1.6 Sistematika Pembahasan . . . 3

2 Landasan Teori 5
- 2.1 *JavaScript* . . . 5
  - 2.1.1 Strings . . . 5
  - 2.1.2 *Numbers* . . . 6
  - 2.1.3 *Booleans* . . . 7
  - 2.1.4 *Operators* . . . 8
  - 2.1.5 *Variables* . . . 9
  - 2.1.6 *Functions* . . . 11
  - 2.1.7 *Conditionals* . . . 12
  - 2.1.8 *Arrays* . . . 13
  - 2.1.9 *Objects* . . . 13
  - 2.1.10 *Input and Output* . . . 14
- 2.2 CodeIgniter 3 . . . 15
  - 2.2.1 *Application Flow Chart* . . . 16
  - 2.2.2 *Model-View-Controller* . . . 16
    - 2.2.2.1 *Model* . . . 17
    - 2.2.2.2 *View* . . . 18
    - 2.2.2.3 *Controller* . . . 20
- 2.3 BASH . . . 21
- 2.4 Node.js . . . 21
  - 2.4.1 *Input* dan *Output* . . . 22
  - 2.4.2 *Package Manager* . . . 23
- 2.5 *Twig* . . . 23

3 Analisis 25
- 3.1 Analisis Sistem Kini . . . 25
  - 3.1.1 Instalasi . . . 28
  - 3.1.2 *Add Assignment* . . . 29
    - 3.1.2.1 Aturan *Submission* . . . 34
  - 3.1.3 Mode, View, Controller . . . 34
    - 3.1.3.1 Model . . . 35

3.1.3.2 View . . . 39
3.1.3.3 Controller . . . 40
3.1.4 Clean URLs . . . 44
3.1.5 Deteksi Kecurangan . . . 44
3.1.6 *Submit* . . . 45
3.1.7 *Final Submission* . . . 46
3.2 Analisis Sistem Usulan . . . 46

4 Perancangan **49**
4.1 Rancangan Perubahan Kode Program . . . 49
4.1.1 Memilih bahasa JavaScript . . . 49
4.1.2 Unggah file JavaScript . . . 50
4.1.3 Melihat bahasa yang diunggah . . . 53
4.1.4 Melihat kode program yang diunggah . . . 54
4.1.5 Pengecekan kode program . . . 55
4.1.6 Penilaian Kode Program . . . 56

Daftar Referensi **57**

A Kode Program **59**

B Hasil Eksperimen **61**

DAFTAR GAMBAR

2.1 *Flow Chart*Aplikasi 16
2.2 Perputaran *Model-View-Controller* 17

3.1 Use Case Diagram SharIF-Judge 26
3.2 Halaman *Assignment* 29
3.3 Kelas Diagram SharIF Judge 35
3.4 Tampilan *error_404.php* 39
3.5 Tampilan *dashboard.twig* 40
3.6 Tampilan *side_bar.twig* dan *top_bar.twig* 40
3.7 Halaman *Submit* 46
3.8 Halaman *Final Submission* 46
3.9 *Use Case Diagram Fitur Usulan* 47

4.1 *List* yang akan dikostumisasi pada SharIF-Judge 49

B.1 Hasil 1 61
B.2 Hasil 2 61
B.3 Hasil 3 61
B.4 Hasil 4 61

BAB 1

PENDAHULUAN

Pada bab ini, akan dibahas mengenai latar belakang penelitian, rumusan masalah, tujuan, batasan masalah, dan sistematika pembahasan yang dilakukan pada penelitian ini.

1.1 Latar Belakang

Online judge merupakan sebuah sistem yang dirancang untuk melakukan evaluasi dari sumber kode algoritma yang dikirimkan oleh pengguna [1]. Konsep dari *online judge* adalah dengan asumsi pengguna mengirimkan solusi berupa kode program, atau bahkan pengguna mengirimkan *file* yang dapat dijalankan dimana tahap selanjutnya kode atau *file* tersebut akan dievaluasi yang sering kali menggunakan infrastruktur berbasis *cloud*. Dalam perancangan *online judge* harus dipastikan bahwa *online judge* harus dapat menanggulangi berbagai macam serangan seperti memaksakan waktu kompilasi yang tinggi, atau mengakses sumber daya yang dibatasi selama proses evaluasi berlangsung.

Sharif Judge merupakan salah satu *online judge* yang gratis untuk bahasa pemrograman C, C++, Java dan Python . Perangkat lunak ini diciptakan oleh Mohammad Javad Naderi pada tahun 2014 dan bersifat *open source*. Antarmuka Sharif Judge ditulis menggunakan bahasa pemrograman PHP (framework CodeIgniter) dan backend menggunakan BASH[1].

Sharif Judge biasa digunakan untuk mempermudah evaluasi kode program. dan salah satu universitas yang menggunakannya adalah Universitas Katolik Parahyangan (UNPAR) pada jurusan Informatika (IF). Namun Sharif Judge telah dimodifikasi dan berubah namanya menjadi SharIF-Judge. SharIF-Judge ini digunakan pada beberapa kuliah di IF, seperti Dasar Pemrograman dan Algoritma Struktur Data. Tujuan SharIF-Judge digunakan pada perkuliahan, adalah agar dapat mempermudah kegiatan belajar mengajar berlangsung. Mahasiswa dapat dengan mudah melakukan pengiriman kode program atau *file* ke SharIF-Judge dan dapat langsung melihat nilainya. Pengajar yang bertanggung jawab atas kegiatan belajar mengajar tersebut tidak akan kesusahan dalam mengumpulkan hasil pekerjaan para pelajar.

Alur dari penggunaan SharIF-Judge ini diawali dengan pengajar yang membuat soal terlebih dahulu. Setelah soal disiapkan, pengajar dapat membuat *assignment* dengan click tombol *add assignment*. Didalamnya, pengajar diwajibkan untuk memasukan nama *assignment*, waktu dimulainya pengerjaan, waktu selesai pengerjaan, waktu tambahan pengerjaan, daftar peserta, deskripsi soal, dan kunci jawaban dari soal yang sudah dibuat oleh pengajar. Meskipun SharIF-Judge dapat

[1] https://github.com/mjnaderi/Sharif-Judge

membaca berbagai bahasa pemrograman, JavaScript bukan salah satu yang dapat dibaca oleh SharIF-Judge.

JavaScript adalah bahasa skrip sisi klien yang berjalan sepenuhnya di dalam *browser web*[2]. Sebagai contoh dapat dilihat *menu drop-down* yang mencolok, memindahkan teks, dan mengubah konten yang sekarang tersebar luas di situs web. Semua interaksi tersebut diaktifkan melalui JavaScript. Berdasarkan survey yang diberikan pada stackoverflow[2] pada tahun 2021, selama 9 tahun berturut-turut, JavaScript merupakan bahasa pemrograman yang paling sering digunakan. 64,96% developer dari 83,052 responden di dunia menggunakannya.

Berdasarkan data diatas, ditunjukan bahwa JavaScript termasuk dalam bahasa pemrograman yang sangat populer dan banyak digunakan. Sangat disayangkan jika SharIF Judge tidak dapat membaca bahasa tersebut. Akan kesulitan bagi para pengajar karena harus mengikuti perkembangan zaman, namun pemeriksaannya tidak mudah. Kostumisasi dari SharIF Judge akan dilakukan agar *online judge* tersebut dapat membaca JavaScript. Untuk mempermudah kostumasi, pengerjaan akan dilakukan dengan menggunakan OS Linux. Namun dikarenakan OS yang digunakan adalah Windows, akan digunakan *virtual machine* agar dapat mengerjakan dengan OS Linux pada Windows.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, maka rumusan masalah penelitian sebagai berikut:

Bagaimanakah cara agar SharIF-Judge dapat memahami bahasa pemrograman JavaScript untuk dievaluasi.

1.3 Tujuan

Berdasarkan rumusan masalah, maka tujuan penelitian ini adalah sebagai berikut:

Melakukan modifikasi pada SharIF-Judge agar dapat menerima soal JavaScript dan dapat melakukan evaluasi pada JavaScript.

1.4 Batasan Masalah

Batasan masalah yang terdapat pada penelitian ini adalah:

1. Pembuatan program menggunakan Linux
2. Versi NodeJS 16.14.2

1.5 Metodologi

Metodologi yang digunakan dalam penelitian ini adalah sebagai berikut:

1. melakukan studi literatur terhadap SharIF-Judge, JavaScript
2. Mempelajari dan menganalisa cara pembuatan soal pada SharIF-Judge dengan menggunakan bahasa pemrograman Java

[2] https://insights.stackoverflow.com/survey/2021

3. Mempelajari cara membuat tes kasus dan menganalisa cara kerjanya pada SharIF-Judge dengan menggunakan bahasa pemrograman Java
4. Membuat kode program agar SharIF-Judge dapat membaca soal dari JavaScript dan dapat mengevaluasi berdasarkan tes kasus.
5. Melakukan pengujian terhadap SharIF-Judge
6. Melaporkan hasil pembuatan dalam bentuk dokumen skripsi

1.6 Sistematika Pembahasan

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

1. Bab 1 : Pendahuluan
   Bab 1 membahas mengenai gambaran umum penelitian ini. Berisi tentang latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi, dan sistematika pembahasan.
2. Bab 2 : Landasan Teori
   Bab 2 membahas mengenai teori-teori yang mendukung berjalannya penelitian ini serta tentang JavaScript dan dokumentasi SharIF-Judge.
3. Bab 3 : Analisis
   Bab 3 membahas mengenai analisis fitur-fitur yang akan diimplementasi pada SharIF-Judge.
4. Bab 4 : Perancangan
   Bab 4 membahas mengenai perancangan yang dilakukan sebelum masuk ke tahap implementasi
5. Bab 5 : Implementasi dan Pengujian
   Bab 5 membahas mengenai implementasi dan pengujian yang telah dilakukan
6. Bab 6 : Kesimpulan dan Saran
   Bab 6 membahas hasil kesimpulan dari keseluruhan penelitian ini dan saran-saran yang dapat diberikan untuk penelitian berikutnya.

BAB 2

LANDASAN TEORI

Pada bab ini, dibahas mengenai teori–teori yang digunakan dalam penelitian ini yaitu: *CodeIgniter*, Dokumentasi SharIF-Judge, *BASH*, *JavaScript*, dan PHP

2.1 *JavaScript*

Sementara HTML dan CSS membantu membuat desain halaman web, JavaScript membantu membuat fungsionalitas di halaman web [3]. Java dan JavaScript memiliki nama yang sama tetapi keduanya merupakan bahasa pemrograman yang sangan berbeda.

2.1.1 Strings

Pada JavaScript, String merupakan nilai yang terdiri dari teks dan dapat berisi huruf, angka, simbol, tanda baca, dan emoji[3]. String *JavaScript* terkandung dalam sepasang tanda kutip tunggal (') atau tanda kutip ganda ("). Kedua tanda kutip mewakili String tetapi pastikan untuk memilih salah satu. Jika dimulai dengan satu kutipan, maka harus diakhiri dengan satu kutipan. Sebagai contoh penulisannya adalah sebagai berikut:

```
EXAMPLE
'This is a string.';
"This is the 2nd string.";
```

String tersebut tidak hanya untuk penulisan teks saja, namun ada berbagai fungsi logis yang dapat digunakan oleh pengguna yaitu sebagai berikut:

- *JavaScript String Length*
  Tujuan fungsi logis ini adalah untuk menghitung panjang dari karater yang dituliskan. Contoh penulisannya adalah sebagai berikut:

  ```
  EXAMPLE:
  "caterpillar".length;
  OUTPUT:
  11
  ```

- *toLowerCase Method*
  Fungsi logis *toLowerCase* string dalam *JavaScript* mengembalikan salinan string dengan huruf-hurufnya dikonversi menjadi huruf kecil. Namun untuk angka, simbol, dan karakter lain tidak terpengaruh dengan fungsi logis tersebut. Contoh penulisannya adalah sebagai berikut:

```
EXAMPLE:
"THE KIDS".toLowerCase();
OUTPUT:
"the kids"
```

- *toUpperCase Method*
  Fungsi logis *toUpperCase* string mengembalikan salinan string dengan huruf yang dikonversi menjadi huruf besar.Namun untuk angka, simbol, dan karakter lain tidak terpengaruh dengan fungsi logis tersebut. Contoh penulisannya adalah sebagai berikut:

```
EXAMPLE:
"I wish I were big.".toUpperCase();
OUTPUT:
"I WISH I WERE BIG."
```

- *trim*
  Fungsi logis *trim* string mengembalikan salinan string dengan karakter spasi awal dan akhir dihapus. Contoh penulisannya adalah sebagai berikut:

```
EXAMPLE:
" but keep the middle spaces ".trim();
OUTPUT:
"but keep the middle spaces"
```

2.1.2 *Numbers*

Bilangan adalah nilai yang dapat digunakan dalam operasi matematika[3]. Untuk penulisan sintaks tidak diperlukan sesuatu yang khusus untuk angka dan cukup tuliskan langsung ke JavaScript. Contoh penulisannya adalah sebagai berikut:

```
EXAMPLE
12345;
```

Numbers tersebut tidak hanya untuk penulisan angka saja, namun ada berbagai jenis yang dapat digunakan oleh pengguna yaitu sebagai berikut:

- *Decimals and fractions*
  JavaScript tidak membedakan antara bilangan bulat dan desimal, sehingga Anda dapat menggunakannya bersama-sama tanpa harus mengonversi dari satu ke yang lain. Contoh penulisannya adalah sebagai berikut:

```
EXAMPLE:
10 + 3.14159;
OUTPUT:
13.14159
```

- *Fractions*
  Pecahan tidak ada dalam *JavaScript*, tetapi dapat ditulis ulang sebagai masalah pembagian menggunakan operator pembagian "/". Perhatikan bahwa angka yang dihasilkan selalu dikonversi ke desimal sama seperti dengan kalkulator. Contoh penulisannya adalah sebagai berikut:

```
EXAMPLE:
1 / 3;
OUTPUT:
0.3333333333333333
```

  Untuk menggunakan bilangan campuran, diperlukan penggabungan bilangan bulat dan pecahan. Contoh penulisannya adalah sebagai berikut:

```
EXAMPLE:
1 + (4 / 3);
OUTPUT:
2.333333333333333
```

- *Negative numbers*
  Penulisan angka negatif dengan menempatkan operator di depan. Contoh penulisannya adalah sebagai berikut:

```
EXAMPLE:
-3;
OUTPUT:
-3;
```

  Angka negatif dapat juga diperoleh dengan mengurangi angka dari angka yang lebih kecil. Contoh penulisannya adalah sebagai berikut:

```
EXAMPLE:
5-7;
OUTPUT:
-2;
```

2.1.3 *Booleans*

Dalam JavaScript, nilai boolean adalah nilai yang dapat berupa *TRUE* atau *FALSE*[3]. Jika diperlukan mengetahui "ya" atau "tidak" tentang sesuatu, maka fungsi boolean merupakan fungsi yang tepat. Apa pun yang perlu "aktif" atau "mati", "ya" atau "tidak", "benar" atau "salah", atau yang hanya memiliki tujuan sementara, biasanya tepat untuk menggunakan boolean. Contoh penulisannya adalah sebagai berikut:

```
EXAMPLE:
var kitchenLights = false;
kitchenLights = true;
```

```
kitchenLights;
OUTPUT:
true
```

Dalam contoh ini, variabel "kitchenLights" yang disetel ke "true" akan menunjukkan bahwa lampu menyala. Jika disetel ke "false" maka itu berarti mereka tidak aktif. Menyimpan boolean dalam variabel bertujuan untuk melacak nilainya dan mengubahnya dari waktu ke waktu. Boolean digunakan sebagai fungsi untuk mendapatkan nilai variabel, objek, kondisi, dan ekspresi. Faktanya, boolean sangat penting agar kondisional berfungsi.

2.1.4 *Operators*

Operator adalah simbol antara nilai yang memungkinkan operasi yang berbeda seperti penambahan, pengurangan, perkalian, dan banyak lagi[3]. Berikut ini merupakan operator yang dimiliki oleh *javaScript*.

- *Arithmetic*

  Operator + menambahkan dua angka. Contoh penulisannya adalah sebagai berikut:

  ```
  EXAMPLE:
  1+2;
  OUTPUT:
  3
  ```

  Operator - mengurangi satu angka dari angka lainnya. Contoh penulisannya adalah sebagai berikut:

  ```
  EXAMPLE:
  50 -15;
  OUTPUT:
  35
  ```

  Untuk menggunakan bilangan campuran, diperlukan penggabungan bilangan bulat dan pecahan. Contoh penulisannya adalah sebagai berikut:

  ```
  EXAMPLE:
  1 + (4 / 3);
  OUTPUT:
  2.333333333333333
  ```

  Operator * mengalikan dua angka. Perhatikan bahwa itu adalah tanda bintang dan bukan simbol × yang biasa digunakan dalam matematika. Contoh penulisannya adalah sebagai berikut:

  ```
  EXAMPLE:
  3 * 12;
  OUTPUT:
  36
  ```

Operator / membagi satu nomor dengan yang lain. Perhatikan bahwa itu adalah garis miring dan bukan simbol yang biasa digunakan dalam matematika. Contoh penulisannya adalah sebagai berikut:

```
EXAMPLE:
12/4;
OUTPUT:
3;
```

Ekspresi JavaScript mengikuti urutan operasi, jadi meskipun + ditulis terlebih dahulu dalam contoh berikut, perkalian terjadi lebih dulu antara dua angka terakhir dan *. Contoh penulisannya adalah sebagai berikut:

```
EXAMPLE:
1 + 100 * 5;
OUTPUT:
501
```

- *Grouping*
  () operator mengelompokkan nilai dan operasi lain. Kode yang terletak di antara tanda kurung dievaluasi terlebih dahulu saat JavaScript menyelesaikan setiap operasi yang bergerak dari kiri ke kanan. Menambahkan operator pengelompokan ke contoh sebelumnya menyebabkan 1 + 100 dievaluasi terlebih dahulu. Contoh penulisannya adalah sebagai berikut:

```
EXAMPLE:
(1 + 100) * 5;
OUTPUT:
505
```

- *Concatenation*
  Operator + juga dapat menggabungkan string. Contoh penulisannya adalah sebagai berikut:

```
EXAMPLE:
"news" + "paper";
OUTPUT:
"newspaper"
```

- *Assignment*
  Operator = memberikan nilai. Ini digunakan untuk mengatur nilai variabel. Contoh penulisannya adalah sebagai berikut:

```
EXAMPLE:
var dinner = "sushi";
```

2.1.5 *Variables*

Variabel diberi nama nilai dan dapat menyimpan semua jenis nilai JavaScript[3]. Contoh penulisannya adalah sebagai berikut:

```
EXAMPLE
var x = 100;
```

Penulisan diatas menyatakan hal sebagai berikut:

- var merupakan kata yang menyatakan kepada *javascript* ingin mendeklarasikan sebuah variabel
- x adalah nama dari variabel yang dideklarasikan
- = merupakan operator yang menyatakan kepada *javascript* bahwa kata selanjutnya adalah nilai dari variabel tersebut
- 100 adalah nilai dari variabel untuk disimpan
- *Using Variables*
  Setelah mendeklarasikan variabel, variabel dapat direferensikan dengan nama di tempat lain dalam kode Anda. Contoh penulisannya adalah sebagai berikut:

  ```
  EXAMPLE:
  var x = 100;
  x + 102;
  OUTPUT:
  202
  ```

  nama variabel bahkan digunakan saat mendeklarasikan variabel lain.

  ```
  EXAMPLE:
  var x = 100;
  var y = x + 102;
  y;
  OUTPUT:
  202
  ```

- *Reassigning variables*
  Nilai baru dapat diberikan pada variabel yang ada kapan saja setelah dideklarasikan.

  ```
  EXAMPLE:
  var weather = "rainy";
  weather = "sunny";
  weather;
  OUTPUT:
  "sunny"
  ```

- *Naming variables*
  Penulisan nama variabel cukup fleksibel, namun ada beberapa aturan yang harus diikuti yaitu sebagai berikut:
  - Nama variabel harus diawali dengan huruf, garis bawah "_", atau dolar "$".
  - Setelah huruf pertama, dapat digunakan angka, huruf, garis bawah, atau tanda dolar.
  - Dilarang menggunakan kata kunci khusus dari *JavaScript* apapun.

  Maka dari itu berikut ini adalah contoh penulisan nama variabel yang valid

```
EXAMPLE:
var camelCase = "lowercase word, then uppercase";
var dinner2Go = "pizza";
var I_AM_HUNGRY = true;
var _Hello_ = "what a nice greeting"
var $_$ = "money eyes";
```

2.1.6 *Functions*

Fungsi *JavaScript* adalah blok kode yang dapat digunakan kembali yang melakukan tugas tertentu, mengambil beberapa bentuk input dan mengembalikan output[3]. Untuk mendefinisikan suatu fungsi, harus digunakan kata kunci fungsi, diikuti dengan nama, diikuti dengan tanda kurung “( )”. Kemudian dituliskan logika fungsi di antara tanda kurung kurawal “”. Contoh penulisannya adalah sebagai berikut:

```
EXAMPLE
function addTwoNumbers(x, y) {
    return x + y;
}
```

Setelah fungsi JavaScript didefinisikan atau dideklarasikan, fungsi dapat digunakan dengan merujuk namanya dengan tanda kurung tepat setelahnya. Perhatikan bahwa suatu fungsi tidak harus memiliki parameter.

```
EXAMPLE:
function greetThePlanet() {
    return "Hello world!";
}
greetThePlanet();
OUTPUT:
"Hello world!"
```

Namun, jika suatu fungsi memiliki parameter, Anda harus memberikan nilai di dalam tanda kurung saat menggunakan fungsi:

```
EXAMPLE:
function square(number) {
    return number * number;
}
square(16);
OUTPUT:
256
```

Dalam contoh di atas, *number* tersebut harus diberikan angka agar fungsi berfungsi dan menerima output yang sesuai. Jika tidak, kode Anda tidak akan berfungsi dan Anda akan mendapatkan pesan kesalahan. Jika fungsi tersebut membutuhkan lebih dari satu argumen, pisahkan dengan menggunakan koma di dalam tanda kurung tersebut.

2.1.7 *Conditionals*

Conditional statements mengontrol perilaku dalam *JavaScript* dan menentukan apakah potongan kode dapat dijalankan atau tidak[3]. Terdapat 3 jenis kondisional dalam *JavaScript* yaitu sebagai berikut:

- *IF*: di mana jika suatu kondisi benar, itu digunakan untuk menentukan eksekusi untuk blok kode.
- *ELSE*: di mana jika kondisi yang sama salah, itu menentukan eksekusi untuk blok kode.
- *ELSE IF*: ini menentukan tes baru jika kondisi pertama salah.

Berikut merupakan Contoh penulisan untuk *IF*:

```
    EXAMPLE:
    if (10 > 5) {
    var outcome = "if block";
    }
    outcome;
    OUTPUT:
    "if block"
```

Berikut merupakan contoh penulisan untuk *ELSE*

```
    EXAMPLE:
if ("cat" === "dog") {
      var outcome = "if block";
} else {
      var outcome = "else block";
}
outcome;
    OUTPUT:
    "else block"
```

Berikut merupakan contoh penulisan untuk *ELSE IF*

```
    EXAMPLE:
    if (false) {
      var outcome = "if block";
    } else if (true) {
      var outcome = "else if block";
    } else {
      var outcome = "else block";
    }
```

```
    outcome;
    OUTPUT:
    "else if block"
```

2.1.8 *Arrays*

Array adalah nilai seperti wadah yang dapat menampung nilai lain[3]. Nilai-nilai di dalam array disebut elemen. Contoh penulisannya sebagai berikut:

```
    EXAMPLE:
    var breakfast = ["coffee", "croissant"];
    breakfast;
    OUTPUT:
    ["coffee", "croissant"]
```

Elemen pada array tidak semuanya harus memiliki tipe nilai yang sama. Elemen dapat berupa nilai JavaScript apa pun bahkan array lainnya. Contohnya penulisannya sebagai berikut:

```
    EXAMPLE:
    var hodgepodge = [100, "paint", [200, "brush"], false];
    hodgepodge;
    OUTPUT:
    [100, "paint", [200, "brush"], false]
```

Untuk mengakses salah satu elemen di dalam array, harus digunakan tanda kurung dan angka seperti ini: myArray[3]. Array JavaScript dimulai dari 0, jadi elemen pertama akan selalu berada di dalam [0].

```
    EXAMPLE:
    var sisters = ["Tia", "Tamera"];
    sisters[0];
    OUTPUT:
    "Tia"
```

Untuk mendapatkan elemen terakhir, Anda dapat menggunakan tanda kurung dan ‘1‘ kurang dari properti panjang larik.

```
    EXAMPLE:
    var actors = ["Felicia", "Nathan", "Neil"];
    actors[actors.length -1];
    OUTPUT:
    "Neil"
```

2.1.9 *Objects*

Objek pada JavaScript adalah variabel yang berisi beberapa nilai data. Nilai dalam objek JavaScript dikenal sebagai properti[3]. Objek menggunakan kunci untuk menamai nilai, seperti yang dilakukan

dengan variabel. Contohnya adalah sebagai berikut:

```
EXAMPLE:
var course = {
   name: "GRA 2032",
   start: 8,
   end: 10
};
```

Ada dua cara untuk mengakses nilai properti objek. Dapat digunakan notasi titik dengan nama properti setelah titik objectName.propertyName seperti pada contoh di bawah ini:

```
EXAMPLE:
var course = {
   name: "GRA 2032",
   start: 8,
   end: 10
};
course.name;
OUTPUT:
"GRA 2032"
```

Atau Dapat digunakan notasi braket dengan nama properti di dalam string di dalam tanda kurung siku - objectName[“propertyName”] seperti pada contoh di bawah ini:

```
EXAMPLE:
var course = {
   name: "GRA 2032",
   start: 8,
   end: 10
};
course["name"];
OUTPUT:
"GRA 2032"
```

2.1.10 *Input and Output*

Agar *Javascript* dapat menerima input dari console, diperlukan modul *readline*. Modul *Readline* pada Node.js memungkinkan pembacaan aliran *input* baris demi baris [4]. Modul ini membungkus output standar proses dan objek dengan proses input yang standar. Modul *Readline* memudahkan untuk memberikan *input* dan membaca *output* yang diberikan oleh pengguna. berikut merupakan contoh penulisannya:

```
var readline = require('readline');
```

Untuk mengeluarkan output, pengguna dapat menggunakan kode *console.log()*, berikut ini merupakan contoh kode program dari awal *input* hingga akhir mengeluarkan *output*:

```
const readline = require('readline')

const inquirer = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

inquirer.question("input first number ", number1 => {
  inquirer.question("Input second number ", number2 => {
    num1 = parseInt(number1)
    num2 = parseInt(number2)
      console.log(`${num1 + num2}`);
      inquirer.close();
  });
});

inquirer.on("close", function() {
  console.log("Good bye!");
  process.exit(0);
});
```

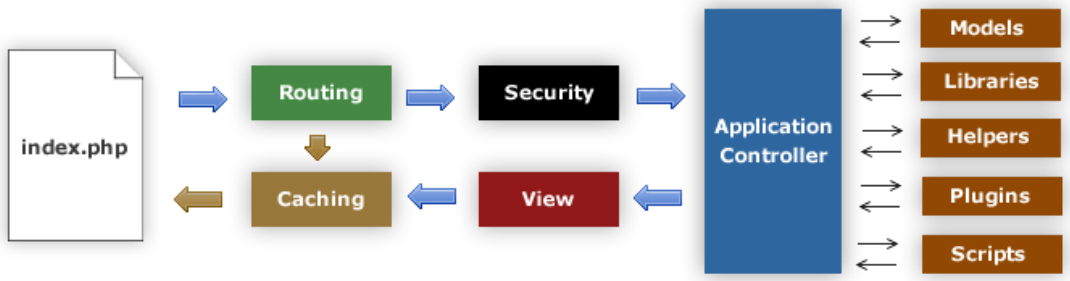
2.2 CodeIgniter 3

CodeIgniter merupakan sebuah *framework* bagi pengguna yang ingin membangun *web application* dengan menggunakan bahasa pemrograman berbasis PHP[5]. Tujuan utamanya adalah untuk mempercepat pengerjaan para pengguna agar tidak perlu menuliskan kode dari awal. Dengan menyediakan kumpulan *libraries* yang kaya untuk tugas-tugas umum yang dibutuhkan, serta antarmuka sederhana dan struktur logis untuk mengakses pustaka ini. SharIF-Judge menggunakan CodeIgniter 3. Untuk dapat menggunakan CodeIgniter 3 direkomendasikan PHP versi 5.6 atau yang lebih baru dari itu dengan alasan agar tidak memiliki permasalahan terhadap bagian keamanan, permasalahan perfoma, dan adanya fitur yang hilang. Database yang dapat didukung oleh CodeIgniter 3 adalah sebagai berikut:

- MySQL (5.1+) via the mysql (deprecated), mysqli and pdo drivers
- Oracle via the oci8 and pdo drivers
- PostgreSQL via the postgre and pdo drivers
- MS SQL via the mssql, sqlsrv (version 2005 and above only) and pdo drivers
- SQLite via the sqlite (version 2), sqlite3 (version 3) and pdo drivers
- CUBRID via cubrid dan pdo *drivers*
- Interbase/Firebird via ibase dan pdo *drivers*
- ODBC via the odbc dan pdo *drivers*

2.2.1 *Application Flow Chart*

Gambar 2.1 merupakan ilustrasi bagaimana data mengalir ke seluruh sistem[5].



Gambar 2.1: *Flow Chart*Aplikasi

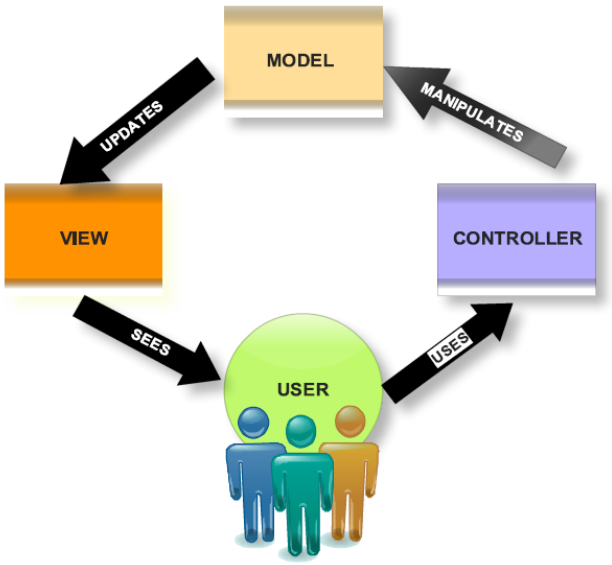
Berikut ini merupakan penjelasan secara urut dari awal sampai akhir bagaimana sistem mengalir:

1. *File index.php* berfungsi sebagai pengontrol depan, menginisialisasi sumber daya dasar yang diperlukan untuk menjalankan CodeIgniter.
2. Router memeriksa permintaan HTTP untuk menentukan apa yang harus dilakukan.
3. Jika terdapat *file cache*, maka akan langsung dikirimkan ke *browser*. *File cache* adalah data yang pernah diakses oleh pengguna. Data tersebut dapat berupa teks, gambar atau file.
4. *HTTP request* dan data pengguna yang dikirim akan disaring terlebih dahulu untuk keamanan. *Application controller* akan dimuat setelah proses penyaringan selesai.
5. *Controller* akan memuat kelas Model, *library* utama dan kelas bantuan.
6. *View* akan memuat tampilan akhir dan dikirim ke *web browser*. Jika *caching* diaktifkan, maka tampilan dimasukan ke dalam *cache* terlebih dahulu sehingga pada permintaan selanjutnya tampilan tersebut dapat diakses lebih cepat.

2.2.2 *Model-View-Controller*

CodeIgniter didasarkan pada pola pengembangan Model-View-Controller[5]. MVC adalah pendekatan perangkat lunak yang memisahkan logika aplikasi dari presentasi. Dalam praktiknya, ini memungkinkan halaman web Anda berisi skrip minimal karena presentasinya terpisah dari skrip PHP. MVC merupakan kumpulan dari tiga bagian: Model, View, dan Controller.

Gambar 2.2 menunjukkan pola dan interaksi dengan pengguna dan aplikasi itu sendiri. Ini adalah tata letak aliran tunggal data, bagaimana data itu dilewatkan di antara setiap komponen, dan akhirnya bagaimana hubungan antara setiap komponen bekerja.



Gambar 2.2: Perputaran *Model-View-Controller*

2.2.2.1 *Model*

Model adalah kelas PHP yang dirancang untuk bekerja dengan informasi dalam database[5]. Berikut adalah contoh tampilan kelas *model*:

```
class Blog_model extends CI_Model {

        public $title;
        public $content;
        public $date;

        public function get_last_ten_entries()
        {
                $query = $this->db->get('entries', 10);
                return $query->result();
        }

        public function insert_entry()
        {
                $this->title = $_POST['title']; // please read the below note
                $this->content = $_POST['content'];
                $this->date = time();

                $this->db->insert('entries', $this);
        }

        public function update_entry()
```

```
        {
                $this->title = $_POST['title'];
                $this->content = $_POST['content'];
                $this->date = time();

                $this->db->update('entries', $this, array('id' => $_POST['id']));
        }

}
```

model biasanya akan dimuat dan dipanggil dari dalam *control*. Untuk memuat model, dapat digunakan metode berikut:

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

jika model terletak di sub-direktori, sertakan jalur relatif dari direktori model. Misalnya, jika memiliki model yang terletak di application/models/blog/Queries.php, dapat dimuat menggunakan:

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

Ketika sebuah model dimuat itu tidak terhubung secara otomatis ke database. Berikut merupakan opsi alternatif untuk menghubungkan:

- Terhubung menggunakan metode database standar yang dijelaskan pada dokumentasi, baik dari dalam kelas *Controller* atau kelas *Model*.
- Memberi tahu metode pemuatan *model* untuk terhubung secara otomatis dengan meneruskan *TRUE* (boolean) melalui parameter ketiga, dan pengaturan konektivitas, seperti yang ditentukan dalam *file* konfigurasi database yang akan digunakan, berikut cara menuliskannya:

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

- melewati pengaturan konektivitas basis data secara manual melalui parameter ketiga:

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

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

2.2.2.2 *View*

View tidak pernah dipanggil secara langsung, *view* harus dimuat oleh *controller*. Dalam kerangka MVC, Controller bertindak sebagai polisi lalu lintas, sehingga bertanggung jawab untuk mengambil

tampilan tertentu. berikut merupakan contoh kode program pada file *view*:

```
<html>
<head>
	<title>My Blog</title>
</head>
<body>
	<h1>Welcome to my Blog!</h1>
</body>
</html>
```

Untuk memuat file tampilan tertentu, dapat digunakan metode berikut:

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

Jika lebih dari satu panggilan terjadi *view* akan ditambahkan bersama-sama. Misalnya, ingin dimiliki tampilan header, tampilan menu, tampilan konten, dan tampilan footer. akan terlihat sebagai berikut:

```
<?php

class Page extends CI_Controller {

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

}
```

File juga dapat disimpan dalam sub-direktori saat melakukannya, harus disertakan nama direktori yang memuat tampilan. Contoh:

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

View juga dapat mengeluarkan tampilan lebih dari satu dengan menggunakan array. Berikut adalah contoh sederhana. Tambahkan ini ke *Controller*:

```
<?php
class Blog extends CI_Controller {

	public function index()
	{
```

```
                $data['todo_list'] = array('Clean House', 'Call Mom', 'Run Errands
                    ');

                $data['title'] = "My Real Title";
                $data['heading'] = "My Real Heading";

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

lalu pada *View* buatlah kode seperti ini:

```
<html>
<head>
        <title><?php echo $title;?></title>
</head>
<body>
        <h1><?php echo $heading;?></h1>

        <h3>My Todo List</h3>

        <ul>
        <?php foreach ($todo_list as $item):?>

                <li><?php echo $item;?></li>

        <?php endforeach;?>
        </ul>

</body>
</html>
```

2.2.2.3 *Controller*

Komponen ketiga dari MVC adalah *Controller*. Tugasnya adalah menangani data yang dikirimkan pengguna serta memperbarui *Model* yang sesuai [5]. CodeIgniter dapat diminta untuk memuat *controller default* ketika URI tidak ada, seperti halnya ketika hanya URL root situs yang diminta. Untuk menentukan pengontrol default, buka file application/config/routes.php dan atur variabel ini:

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

'blog' adalah nama kelas *controller* yang ingin digunakan. Jika sekarang dimuat file index.php utama tanpa menentukan segmen URI apa pun, akan terlihat pesan "Hello World" secara default. CodeIgniter memiliki kelas keluaran yang menangani pengiriman data akhir ke *browser web* secara

otomatis. Informasi lebih lanjut tentang ini dapat ditemukan di halaman *View* dan kelas *Output*. CodeIgniter mengizinkan penambahan metode bernama _output() ke *controller* yang akan menerima data keluaran akhir. Contohnya sebagai berikut:

```
public function _output($output)
{
        echo $output;
}
```

2.3 BASH

BASH adalah penerjemah default pada banyak sistem GNU/Linux yang merupakan singkatan dari *Bourne Again SHell*, Bash adalah shell yang kompatibel yang menggabungkan fitur berguna dari *Korn shell* (ksh) dan *C Shell*(csh)[6]. BASH menawarkan peningkatan fungsional atas sh untuk pemrograman dan penggunaan interaktif. Selain itu, sebagian besar skrip shell dapat dijalankan oleh Bash tanpa modifikasi. Kelebihan yang dapat diberikan oleh Bash antara lain sebagai berikut

- Edit pada *command-line*
- Riwayat penulisan perintah yang tidak dibatasi
- Kontrol pekerjaan
- Fungsi dari shell
- array dengan index tidak terbatas
- bilangan integer dari hanya 2 bit sampai 64 bit

2.4 Node.js

Node.js adalah lingkungan runtime JavaScript *open-source* dan lintas platform[7]. Ini adalah alat yang populer untuk hampir semua jenis proyek Node.js menjalankan *engine* JavaScript V8, inti dari Google Chrome, di luar browser. Ini memungkinkan Node.js menjadi sangat berkinerja.

Aplikasi Node.js berjalan dalam satu proses, tanpa membuat thread baru untuk setiap permintaan. Saat Node.js melakukan operasi *Input/Output* (I/O), seperti membaca dari jaringan, mengakses database atau sistem file, alih-alih memblokir *thread* dan membuang siklus CPU menunggu, Node.js akan melanjutkan operasi saat respons kembali.

Hal ini memungkinkan Node.js untuk menangani ribuan koneksi bersamaan dengan satu server tanpa menimbulkan beban mengelola konkurensi *thread*, yang dapat menjadi sumber *bug* yang signifikan. Di Node.js, standar ECMAScript baru dapat digunakan tanpa masalah, karena tidak perlu menunggu semua pengguna memperbarui browser user dan bertanggung jawab untuk memutuskan versi ECMAScript mana yang akan digunakan dengan mengubah versi Node.js, dan juga dapat mengaktifkan fitur eksperimental tertentu dengan menjalankan Node.js dengan *flag*.

Node.js dapat diinstal dengan berbagai cara. Paket resmi untuk semua platform utama tersedia di https://nodejs.dev/download/. Untuk menjalankan program Node.js terdapat perintah node yang tersedia secara global dan meneruskan nama file yang ingin dijalankan. Jika file aplikasi Node.js utama adalah app.js, dapat dipanggil dengan menuliskan:

```
node app.js
```

Skrip tersebut secara eksplisit memberi tahu *Shell* untuk menjalankan skrip dengan node.js

2.4.1 *Input* dan *Output*

Node.js menyediakan modul konsol yang menyediakan banyak cara yang sangat berguna untuk berinteraksi dengan baris perintah. Pada dasarnya konsol tersebut merupakan konsol yang sama dengan objek konsol yang ditemukan pada browser. Metode yang paling dasar dan paling sering digunakan adalah *console.log()*, yang mencetak string yang diberikan pada konsol. *Console.log()* dapat digunakan seperti sebagai berikut:

```
const x = 'x';
const y = 'y';
console.log(x, y);
```

berdasarkan kode diatas, node.js akan mencetak keduanya. Node.js juga memiliki format spesifik yaitu sebagai berikut:

```
console.log('My %s has %d ears', 'cat', 2);
```

- %s memformat variabel sebagai string
- %d memformat variabel sebagai angka
- %i memformat variabel sebagai bagian integernya saja
- %o memformat variabel sebagai objek

Node.js juga menyediakan modul readline untuk melakukan hal ini: dapatkan input dari aliran yang dapat dibaca seperti aliran *process.stdin*, yang selama eksekusi program Node.js adalah input terminal, satu baris pada satu waktu.

```
const readline = require('readline').createInterface({
  input: process.stdin,
  output: process.stdout,
});

readline.question(`What's your name?`, name => {
  console.log(`Hi ${name}!`);
  readline.close();
});
```

Potongan kode ini menanyakan nama pengguna, dan setelah teks dimasukkan dan pengguna menekan enter. Metode *question()* menunjukkan parameter pertama (pertanyaan) dan menunggu input pengguna. Ini memanggil fungsi panggilan balik setelah enter ditekan. Dalam fungsi panggilan balik ini, menutup antarmuka readline.

2.4.2 *Package Manager*

npm merupakan *package manager* yang standar untuk Node.js. npm mengelola unduhan dependensi pada proyek. Jika sebuah proyek memiliki file package.json, dengan menjalankan:

```
npm install
```

kode program tersebut akan menginstal semua yang dibutuhkan proyek, di folder *node_modules*, dan akan membuatnya jika belum ada. Modul dapat menginstal paket tertentu dengan menjalankan:

```
npm install <package-name>
```

Sejak npm 5, perintah ini menambahkan <package-name> ke dependensi file package.json. Sebelum versi 5, diperlukan tanda –save. Seringkali tanda akan ditambahkan ke perintah sebagai berikut:

- *–save-dev* menginstal dan menambahkan entri ke file *package.json devDependencies*
- –no-save menginstal tetapi tidak menambahkan entri ke dependensi file *package.json*
- –save-optional menginstal dan menambahkan entri ke file *package.json optionalDependencies*
- –no-optional akan mencegah dependensi opsional diinstal

Singkatan dari tanda juga dapat digunakan:

- -S: –save
- -D: –save-dev
- -O: –save-optional

Perbedaan antara devDependencies dan dependensi adalah bahwa yang pertama berisi alat pengembangan, seperti perpustakaan pengujian, sedangkan yang terakhir dibundel dengan aplikasi dalam produksi. Adapun OptionalDependencies perbedaannya adalah bahwa kegagalan build dari dependensi tidak akan menyebabkan instalasi gagal. Tapi itu adalah tanggung jawab program untuk menangani kurangnya ketergantungan.

Jika ingin melakukan update pada *package* dapat dilakukan dengan menggunakan perintah berikut:

```
npm update
```

npm akan memeriksa semua paket untuk versi yang lebih baru yang memenuhi batasan versi.

npm dapat menentukan secara spesifik paket mana yang ingin diperbaharui dengan menuliskan perinta seperti berikut:

```
npm update <package-name>
```

File *package.json* mendukung format untuk menentukan file yang akan dijalankan dengan menggunakan perintah sebagai berikut:

```
npm run <task-name>
```

2.5 *Twig*

Pada perangkat lunak SharIF-Judge, kelas *view* menggunakan *framework* aplikasi yaitu *Twig*. *Twig* merupakan sebuah *template engine modern* untuk PHP[8]. Sebuah *template Twig* dapat

mengandung variables atau *expression* dan *tags*. *Variables* atau *expression* akan diubah pada saat *template Twig* dievaluasi dan *tags* yang akan mengontrol logika dari *template* tersebut. Berikut merupakan *template* kode yang menunjukan beberapa hal mendasar.

```
<!DOCTYPE html>
<html>
    <head>
        <title>My Webpage</title>
    </head>
    <body>
        <ul id="navigation">
        {% for item in navigation %}
        <li><a href="{{ item.href }}">{{ item.caption }}</a></li>
        {% endfor %}
        </ul>
        <h1>My Webpage</h1>
        {{ a_variable }}
    </body>
</html>
```

Ada dua jenis delimiters pada kode di atas, yaitu % ... % dan ... . *Delimiters* % ... % digunakan untuk mengeksekusi sebuah *statements*. Pada kode di atas, *delimiters* % for item in navigation % akan mengeksekusi *statements for-loops* atau pengulangan. *Delimiters* ... digunakan untuk menampilkan nilai. Pada kode di atas, *delimiters* item.href akan menampilkan nilai item.href, *delimiters* item.caption akan menampilkan nilai item.caption dan *delimiters* a_variable akan menampilkan nilai a_variable.

BAB 3

ANALISIS

Bab ini membahas tentang analisis dari SharIF-Judge yang digunakan oleh Teknik Informatika. SharIF-Judge memiliki fungsi utama yaitu untuk mengevaluasi kode program yang telah dikumpulkan oleh pengguna secara otomatis. SharIF-Judge digunakan pada beberapa matak kuliah pemrograman Teknik Informatika Unpar untuk mempermudah evaluasi yang akan dilakukan oleh pengajar.

3.1 Analisis Sistem Kini

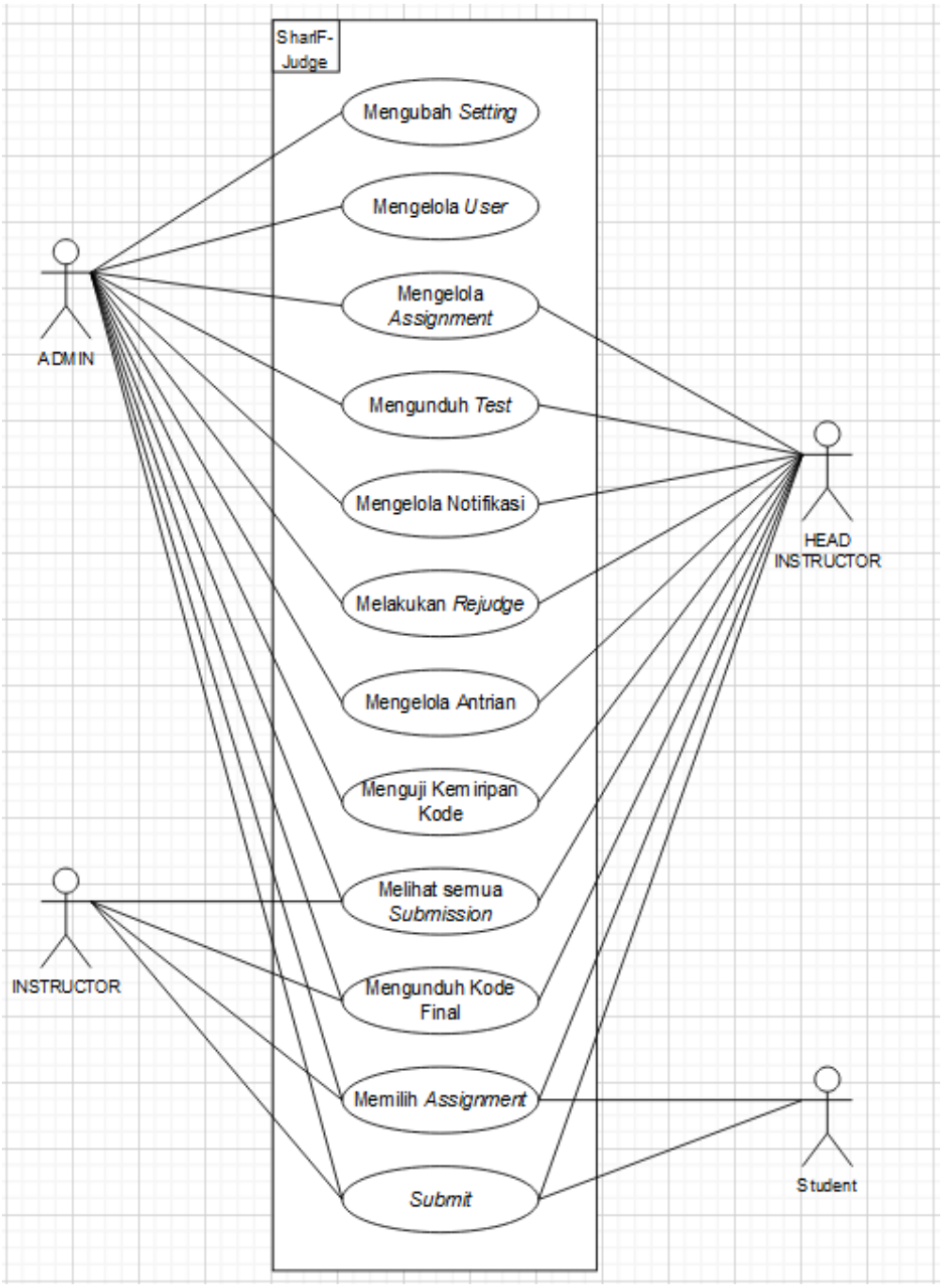
SharIF-Judge adalah *online judge* gratis dan open source untuk bahasa pemrograman C, C++, Java dan Python. SharIF-Judge merupakan hasil *fork* dari Sharif Judge yang dibuat oleh Mohammad Javad Naderi. Versi bercabang ini mengandung banyak perbaikan, sebagian besar karena kebutuhan fakultas Informatika UNPAR. Antarmuka web ditulis dalam PHP (kerangka CodeIgniter) dan backend utama ditulis dalam BASH.-Python di SharIF Judge belum di-sandbox. Hanya tingkat keamanan rendah yang disediakan untuk python. Berikut ini merupakan fitur-fitur yang dimiliki oleh SharIF-Judge antara lain yaitu:

- ***Multiple Role***. Pada SharIF-Judge, *User* memiliki 4 jenis *role* yang dibedakan berdasarkan level, dimana level tersebut digunakan untuk memisahkan aksi yang dapat dilakukan setiap *role*. 4 *role* tersebut adalah *Admin*, *Head Instructor*, *Instructor*, dan *Student*.

Tabel 3.1: Tabel *role*

Role	Level
Admin	3
Head Instructor	2
Instructor	1
Student	0

Pada Tabel 3.1 menunjukan level role yang ada pada SharIF-Judge. Setiap *role* tersebut, dapat melakukan aksi yang berbeda-beda yang disesuaikan dengan level role. Untuk melihat aksi apa saja yang dapat dilakukan setiap *role* dapat dilihat pada Gambar 3.1.



Gambar 3.1: Use Case Diagram SharIF-Judge

Pengguna dapat menambahkan *user* dengan menggunakan fitur *Add User* pada halaman *User*, Pengguna harus mengisi semua informasi yang ada pada text area. Baris dimulai dengan komentar #. Setiap baris lainnya mewakili pengguna dengan sintaks berikut:

```
USERNAME EMAIL PASSWORD ROLE
-Username dapat berisikan huruf kecil atau nomor dan harus terdiri antara 3
  sampai 20 karakter.
-Password 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 yang disebutkan sebagai berikut: ‘admin’,
    ‘head_instructor’, ‘instructor’, dan ‘student’
```

Berikut ini adalah contoh penggunaan sintaks untuk *add user*:

```
# 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
```

- ***Sandboxing***. Sebuah mekanisme dimana sebuah aplikasi yang dikirimkan oleh pengguna, dapat dijalankan dalam lingkungan virtual yang aman dan menghindari adanya serangan keluar dari Sharif Judge.

- ***Cheat Detection***. Berguna sebagai pendeteksi adanya kode yang mirip dan sebagai pendeteksi kecurangan. Untuk pengecekan digunakan *Moss*(*Measure Of Software Similarity*) Moss adalah perangkat lunak yang digunakan oleh guru dan penerbit untuk menemukan plagiarisme perangkat lunak. Perangkat lunak ini dikembangkan oleh Stanford, dan telah menjadi alat utama untuk memeriksa plagiarisme kode. MOSS diciptakan untuk menghentikan kelaziman pada siswa untuk menyalin kode dan langsung selesai tanpa adanya pengertian dari siswa terhadap pelajaran tersebut[9]
- **Penilaian berbeda untuk keterlambatan pengumpulan**.
  Hal ini berguna agar jika pelajar ada yang membutuhkan pengertian khusus dan memiliki alasan yang baik, pengajar dapat memberikan pengecualian dan memberikan hukuman ringan.
- **Antrian pengiriman**. Hal ini berguna agar Sharif Judge tidak *down* akibat banyaknya pengiriman.
- **Mengunduh nilai dalam bentuk *excel***. Hal ini berguna agar pengajar tidak mendapatkan kesulitan untuk menyimpan data nilai pelajar.

- **Mengunduh kode yang dikumpulkan dalam bentuk zip**. Hal ini berguna agar ketika kode yang harus dikumpulkan ada banyak dan ketika pengajar ingin memeriksa, file tersedia dengan rapih.
- **Metode "Output Comparison" dan "Tester Code" untuk memeriksa *output***. Hal ini berguna agar pelajar mengetahui apakah pekerjaan yang dikirimkan tersebut sudah benar atau masih kurang tepat.
- **Penilaian ulang** . Hal ini berguna agar ketika pengajar melakukan kesalahan penilaian, pengajar dapat melakukan edit.
- **Papan nilai**. Hal ini berguna agar pelajar dapat melihat nilai dari pelajar-pelajar yang lain dan bisa menjadi motivasi bagi pelajar tersebut.
- **Notifikasi** . hal ini berguna agar ketika ada tugas yang harus dikumpulkan, pelajar mengetahuinya dan tidak terlewat.

3.1.1 Instalasi

Untuk dapat menjalankan SharIF-Judge, diwajibkan untuk memiliki server Linux dan mengikuti kebutuhannya sebagai berikut:

- Menjalankan PHP versi 5.3 atau versi yang lebih baru.
- Pengguna dapat menjalankan PHP dari command line dan pengguna perlu menginstall paket PHP CLI.
- Memiliki *Mysql* atau *PostgreSql database*.
- PHP memiliki akses untuk menjalankan perintah *shell* terutama untuk fungsi *shell_exec*. contohnya seperti *command* di bawah ini:

```
echo shell_exec(‘‘php -v’’);
```

- Untuk melakukan proses kompilasi dan menjalankan kode yang dikumpulkan adalah (*gcc, g++, javac, java, python2, python3 commands*)
- Disarankan untuk melakukan instalasi *Perl* dengan alasan agar memiliki ketepatan waktu, penggunaan *memory* yang terbatas dan memaksimalkan batas ukuran pada *output* kode yang dikirimkan.

Jika persyaratan diatas telah selesai dilakukan, dapat melakukan instalasi sebagai berikut:

- Mengunduh versi terakhir dari Sharif Judge dan unpack file yang berhasil diunduh, letakan pada direktori html publik
- Untuk mempermudah pindahkan folder *system* dan *application* keluar dari direktori publik dan masukan *path* lengkap pada file index.php

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

- Membuat sebuah *Mysql* atau *PostgreSql database* untuk Sharif Judge.
- Mengatur koneksi database di file application/config/database.php.

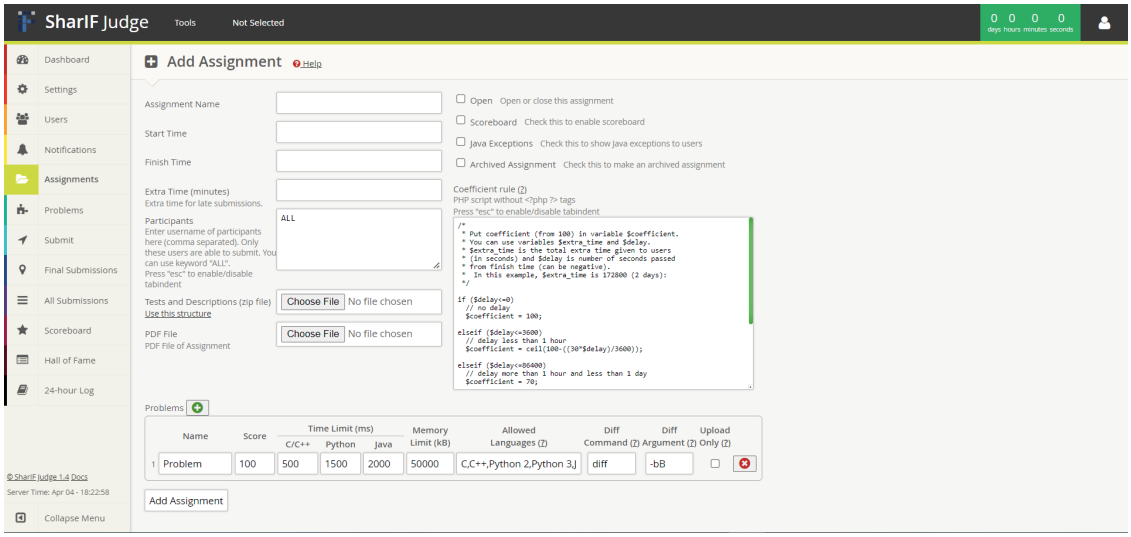
```
/* Enter database connection settings here: */
‘dbdriver’ => ‘postgre’, // database driver (mysqli, postgre)
‘hostname’ => ‘localhost’, // database host
```

```
'username' => '', // database username
'password' => '', // database password
'database' => '', // database name
'dbprefix' => 'shj_', // table prefix
/*****************************************/
```

- Membuat direktori application/cache/Twig agar dapat ditulis oleh PHP
- Membuka halaman utama SharIF-Judge pada web browser
- *Log in* menggunakan akun *admin*
- Memindahkan folder *tester* dan *assigments* di luar direktori publik lalu simpan *path* lengkap pada halaman *Settings* . Dua folder tersebut harus dapat ditulis oleh PHP. File-file yang diunggah akan disimpan di folder *assigments* sehingga tidak dapat diakses publik.

3.1.2 *Add Assignment*

Pengguna dapat menambahkan *assignment* dengan cara mengklik menu *Assignments* lalu klik *add* pada halaman tersebut. Pada Gambar 3.2 merupakan halaman *add assignment*. Berikut ini



Gambar 3.2: Halaman *Assignment*

merupakan pengaturan yang terdapat pada *add assignment*:

- *Assignment Name*
  Memberikan nama pada *assignment* yang akan dibuat

- *Start Time*
  Menentukan dimulainya waktu dari *assignment* tersebut dan peserta tidak dapat mengumpulkan *assignment* tersebut lebih cepat dari *start time*. Format yang digunakan untuk *start time* adalah *MM/DD/YYYY HH:MM:SS*. Contoh penulisannya adalah 08/31/2013 12:00:00

- *Finish Time and Extra Time*
  Peserta tidak dapat melakukan aksi *submit* setelah *Finish time + Extra time*. *Assignment* yang telat akan dikalikan dengan koefisien yang sudah ditentukan. Pengguna harus menulis

script dalam bahasa PHP untuk menghitung koefisien pada bidang "*Coefficient Rule*". Format yang digunakan dalam pengaturan *finish time* adalah *MM/DD/YYYY HH:MM:SS*. Contoh penulisannya adalah 08/31/2013 23:59:59. "Extra Time" akan terhitung dalam satuan menit. Pengguna juga dapat menggunakan operator aritmatika seperti *, -, +, /. Contoh 120 (2 jam) atau 48*60 (2 hari).

- *Participants*
  Pengaturan ini berfungsi untuk membatasi peserta yang dapat mengumpulkan assignment. Pengguna dapat menggunakan kata kunci *ALL* pada kolom *Participants* untuk mengijinkan seluruh peserta agar dapat mengumpulkan *assignment*. Untuk membatasi peserta tertentu, pengguna dapat memasukan username peserta pada kolom *Participants*. Setiap username dapat dipisahkan menggunakan tanda koma. Contoh: admin, instructor1, instructor2, student1.

- *Tests*
  Pengguna dapat mengirim tes kasus dalam *file* zip dengan syarat saat menambahkan tugas, Pengguna harus menyediakan file zip yang berisi kasus uji. File zip ini harus berisi folder untuk setiap masalah (masalah Upload-Only tidak memerlukan folder apa pun). Nama folder harus p1, p2, p3, ... Metode yang dapat digunakan untuk memeriksa keluaran setiap masalah adalah dengan metode "Perbandingan *Input/Output*" dan metode "*Tester*".
  - Perbandingan *Input/Output*
    Dalam metode ini, Pengguna harus meletakkan beberapa file *input* dan *output* di folder masalah. SharIF-Judge memberikan setiap file input tes ke kode pengguna dan membandingkan output pengguna dengan output tes. File input harus di folder in dengan nama input1.txt, input2.txt, ... dan file output harus di folder out dengan nama output1.txt, output2.txt, ...
  - *Tester*
    Dalam metode ini, Pengguna harus menyediakan beberapa file pengujian input dan file C++ (tester.cpp) dan (opsional) beberapa file pengujian output. SharIF-Judge memberikan file tes input ke kode pengguna dan mendapatkan output pengguna. Kemudian tester.cpp mendapatkan input tes, output tes, dan output pengguna. Jika output pengguna benar, mengembalikan 0, jika tidak mengembalikan 1. Pengguna dapat menggunakan template kode ini 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]); /* This stream reads from test's input
```

```
    file */
  ifstream test_out(argv[2]); /* This stream reads from test's output
    file */
  ifstream user_out(argv[3]); /* This stream reads from user's output
    file */

  /* Your code here */
  /* If user's output is correct, return 0, otherwise return 1 */
  ...
}
```

Diberikan contoh dari file tes kasus dengan struktur file sebagai berikut:

```
.
|-- 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
```

Berikut ini merupakan contoh dari *Tester*

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

- *Scoreboard*
  Pengguna dapat mengaktifkan atau mematikan papan nilai dengan menggunakan pilihan ini.
- *Java Exceptions*
  Pengguna dapat mengaktifkan dan mematikan java exceptions yang ditunjukan kepada *role students*. Perubahan pada pilihan ini tidak berdampak pada kode yang sebelumnya sudah dinilai. Nama *exception* akan muncul ketika pada file pathtester/java_exceptions_list berisikan nama *exception* tersebut. Berikut hasil exception yang ditunjukan jika pengguna mengaktifkan pengaturan *Java Exceptions*:

```
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 skrip PHP di sini yang menghitung koefisien dikalikan dengan skor. Skrip pengguna harus memasukkan koefisien (dari 100) ke dalam variabel $koefisien. Pengguna dapat menggunakan variabel $extra_time dan $delay. $extra_time adalah total waktu tambahan yang diberikan kepada pengguna dalam detik (waktu tambahan yang Anda masukkan di bidang Extra Time) dan $delay adalah jumlah detik yang berlalu dari waktu selesai (bisa negatif). Skrip PHP ini tidak boleh mengandung tag <?php , <? , ?>. Dalam contoh ini, $extra_time adalah 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 dalam menjalankan kode dalam satuan milisekon. Program yang ditulis menggunakan Python dan Java biasanya lebih lambat dari C/C++. Oleh karena itu Python dan Java membutuhkan waktu yang lebih lama.

- *Memory Limit*
  Pengguna dapat mengatur batas memori dalam satuan kilobyte, namun penggunaan *Memory Limit* tidak terlalu akurat.

- *Allowed Languages*
  Melakukan pengaturan bahasa untuk setiap kasus yang dipisahkan menggunakan koma. Ba-

hasa yang tersedia seperti C, C++, Java, Python 2, Python 3, zip, PDF. Pengguna dapat menggunakan zip atau PDF jika mengaktifkan pilihan Upload Only. Contoh: C, C++ , zip atau Python 2,Python 3 atau Java ,C.

- *Diff Command*
  Command ini digunakan untuk membandingkan keluaran dengan keluaran yang benar. Secara default SharIF-Judge menggunakan *diff*, namun pengguna dapat mengubah *command* pada bagian ini.

- *Diff Arguments*
  Pengguna dapat mengatur argumen dari *Diff Command* disini. Untuk melihat daftar lengkap *diff argumen*, pengguna dapat melihat *man diff*. SharIF-Judge menambahkan dua pilihan baru yaitu *ignore* dan *identical*. *Ignore* akan menghiraukan semua baris baru dan spasi. *Identical* tidak akan menghiraukan apapun namun keluaran dari file yang dikumpulkan harus identik dengan keluaran test case agar dapat diterima.

- *Upload Only*
  Jika pengguna mengatur problem sebagai *Upload-Only*, maka SharIF-Judge tidak akan menilai *assignment* pada kasus tersebut. Pengguna dapat menggunakan zip dan PDF pada *allowed languages* jika mengaktifkan pilihan ini.

3.1.2.1 Aturan *Submission*

Untuk melakukan pengumpulan jawaban pada sebuah *assignment*, terdapat beberapa aturan yang harus dipatuhi yaitu:

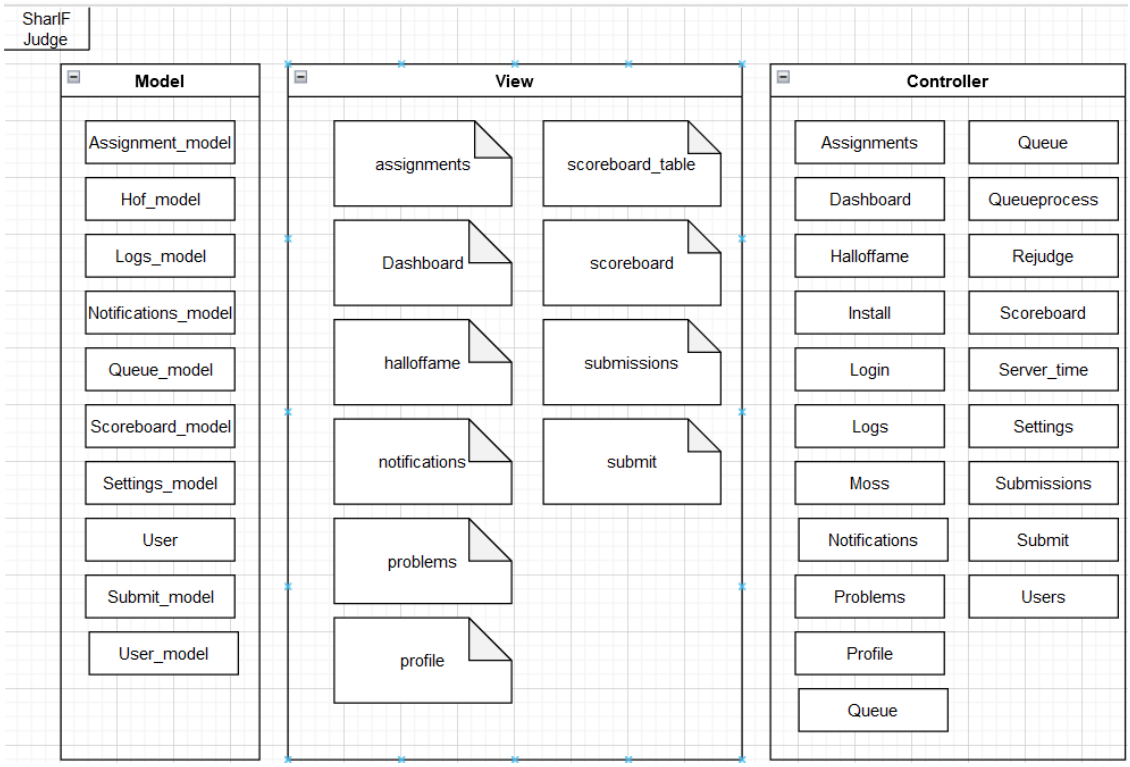
1. Memilih *assignment* yang ingin dikumpulkan pada halaman *Assignment*.
2. Jawaban tidak dapat dikumpulkan jika peserta tidak terdaftar pada *assignment* yang dipilih.
3. Jawaban tidak dapat dikumpulkan jika waktu sekarang belum melewati '*Start time*' pada *assignment* yang dipilih.
4. Jawaban tidak dapat dikumpulkan jika waktu sekarang telah melewati '*Finish time*' pada *assignment* yang dipilih.
5. Jawaban tidak dapat dikumpulkan jika *assignment* yang dipilih berstatus *close*.
6. Pada halaman *Submit*, para peserta dapat memilih *problem* yang ingin dikumpulkan, bahasa pemrograman yang digunakan dan file jawaban.
7. Menekan tombol Submit untuk mengumpulkan jawaban.

Setelah menekan tombol *Submit*, pengguna akan diarahkan ke halaman *All Submission*. Pada tahap ini, jawaban para peserta telah berhasil dikumpulkan ke SharIF-Judge dan peserta dapat memilih jawaban yang mana yang ingin dikumpulkan dengan mencentang salah satu jawaban yang telah dikumpulkan.

3.1.3 Mode, View, Controller

SharIF Judge menggunakan *framework* CodeIgniter 3. Seperti yang telah dibahas pada 2.2.2, SharIF Judge menggunakan pola arsitektur MVC. Diagram kelas untuk SharIF Judge dapat dilihat pada

gambar 3.3. Pada komponen *View* digunakan simbol komentar dengan alasan *view* tersebut tidak berbentuk kelas.



Gambar 3.3: Kelas Diagram SharIF Judge

3.1.3.1 Model

Berikut ini merupakan *model* pada SharIF Judge

- Assignment_model
  Model ini bertujuan untuk menangani tabel shj_assignments Fungsi yang dimiliki:
  - add_assignment ($id, $edit)
    Menambah atau memperbaharui *assignment*.
  - delete_assignment ($assignment_id)
    Menghapus sebuah assignment.
  - all_assignments()
    Mengambil seluruh assignment.
  - new_assignment_id()
    Menentukan integer terkecil yang dapat digunakan sebagai id assignment baru.
  - all_problems ($assignment_id)
    Mengambil seluruh problem dari assignment.
  - problem_info ($assignment_id, $problem_id)
    Mengambil sebuah problem.
  - assignment_info ($assignment_id)
    Mengambil sebuah assignment.
  - is_participant ($participants, $username)

Mengembalikan nilai TRUE jika $username terdapat dalam $participants
  - increase_total_submits ($assignment_id) Menambah jumlah total submit sebuah assignment sebanyak satu.
  - set_moss_time ($assignment_id) Memperbarui "*Moss Update Time*" untuk sebuah *assignment*.
  - get_moss_time ($assignment_id) Mengambil "*Moss Update Time*" untuk sebuah *assignment*.
  - save_problem_description ($assignment_id, $problem_id, $text, $type)
    Menambah atau memperbarui deskripsi sebuah *problem*.
  - _update_coefficients($a_id, $extra_time, $finish_time, $new_late_rule)
    Memperbarui koefisien pada seluruh submission untuk sebuah assignment.
- Hof_model
  Model dengan tujuan untuk menangani informasi pada *hall of fame*. Fungsi yang dimiliki:
  - get_all_final_submission()
    Mengambil seluruh final submission.
  - get_all_user_assignments($username)
    Mengambil seluruh assignment dan problem untuk user tertentu.
- Logs_model
  Model dengan tujuan untuk menangani tabel shj_logins. Fungsi yang dimiliki:
  - insert_to_logs ($username, $ip_adrress)
    Menambah sebuah catatan login dan menghapus catatan yang sudah melewati 24 jam.
  - get_all_logs ()
    Mengambil seluruh catatan login.
- Notifications_model
  Model yang bertujuan untuk menangani tabel shj_notifications. Fungsi yang dimiliki:
  - get_all_notifications ()
    Mengambil seluruh notifikasi.
  - get_latest_notifications ()
    Mengambil 10 notifikasi terbaru.
  - add_notification ($title, $text)
    Menambah notifikasi baru.
  - update_notification ($id, $title, $text)
    Memperbarui sebuah notifikasi.
  - delete_notification ($id)
    Menghapus sebuah notifikasi.
  - get_notification ($notif_id)
    Mengambil sebuah notifikasi.
  - have_new_notification ($time)
    Mengembalikan TRUE jika terdapat notifikasi setelah $time.
- Queue_model
  Model yang bertujuan untuk menangani tabel shj_queue. Fungsi yang dimiliki:
  - in_queue($username, $assignment, $problem)

Mengembalikan nilai TRUE jika sebuah submission sudah berada dalam antrian.
  - get_queue ()
    Mengambil seluruh antrian.
  - empty_queue ()
    Mengosongkan antrian.
  - add_to_queue ($submit_info)
    Menambahkan sebuah *submission* ke dalam antrian.
  - rejudge ($assignment_id, $problem_id)
    Menambahkan seluruh *submission* dari sebuah *problem* ke dalam antrian untuk dinilai ulang.
  - rejudge_single ($submission)
    Menambahkan sebuah *submission* ke dalam antrian untuk dinilai ulang.
  - get_first_item ()
    Mengambil baris pertama dari antrian.
  - remove_item ($username, $assignment, $problem, $submit_id) Menghapus sebuah baris dari antrian.
  - save_judge_result_in_db ($submission, $type)
    Menyimpan hasil dari penilaian ke dalam *database*.
- Scoreboard_model
  Model yang bertujuan untuk menangani tabel shj_scoreboard. Fungsi yang dimiliki:
  - _generate_scoreboard($assignment_id)
    Membuat *scoreboard* untuk sebuah *assignment*.
  - update_scoreboards ()
    Memperbarui *scoreboard* untuk seluruh *assignment*.
  - update_scoreboard ($assignment_id)
    Memperbarui *scoreboard* untuk sebuah *assignment*.
  - get_scoreboard($assignment_id)
    Mengambil *scoreboard* untuk sebuah *assignment*.
- Settings_model
  Model yang bertujuan untuk menangani tabel shj_settings. Fungsi yang dimiliki:
  - get_setting($key)
    Mengambil sebuah pengaturan.
  - set_setting ($key, $value)
    Memperbarui sebuah pengaturan.
  - get_all_settings ()
    Mengambil seluruh pengaturan.
  - set_settings ($settings)
    Memperbarui beberapa pengaturan.
- Submit_model
  Model yang bertujuan untuk menangani tabel shj_submissions. Fungsi yang dimiliki:
  - get_submission ($uname, $assignment, $problem, $submit_id)
    Mengambil sebuah *submission*.

- get_final_submissions ($a_id, $u_lv, $uname, $p_num, $f_user, $f_prblm)
  Mengambil seluruh *final submission* untuk sebuah *assignment*.
- get_all_submissions ($a_id, $u_lv, $uname, $p_num, $f_user, $f_prblm)
  Mengambil seluruh *submission* untuk sebuah *assignment*.
- count_final_submissions ($a_id, $u_lv, $uname, $f_user, $f_prblm)
  Menghitung jumlah *final submission* dari *user* tertentu.
- count_all_submissions ($a_id, $u_lv, $uname, $f_user, $f_prblm)
  Menghitung jumlah *submission* dari *user* tertentu.
- set_final_submission ($uname, $assignment, $problem, $submit_id)
  Memperbarui sebuah *submission* menjadi *final submission*.
- add_upload_only ($submit_info)
  Menambahkan *submission upload only* ke dalam database.

• User

Model yang bertujuan untuk menangani informasi preferensi setiap *user*. Fungsi yang dimiliki:

- select_assignment ($assignment_id)
  Menetapkan *assignment* yang dipilih.
- save_widget_positions ($positions)
  Memperbarui posisi *widget*.
- get_widget_positions ()
  Mengambil posisi *widget*.

• User_model

Model yang bertujuan untuk menangani tabel shj_users. Fungsi yang dimiliki:

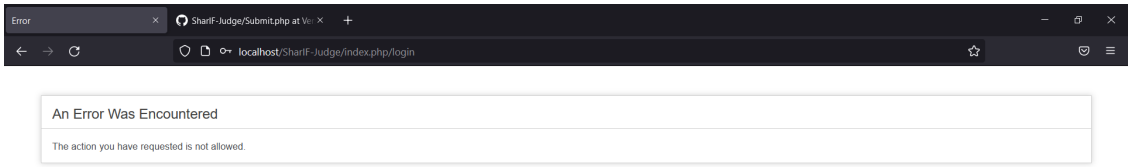
- have_user ($username)
  Mengembalikan nilai TRUE jika terdapat *user* dengan nama $username.
- user_id_to_username ($user_id)
  Mengembalikan *username* dari *user* dengan id tertentu.
- username_to_user_id ($username)
  Mengembalikan id dari *user* dengan *username* tertentu.
- have_email ($email, $username)
  Mengembalikan nilai TRUE jika terdapat *user* selain $username dengan email $email.
- add_user ($username, $email, $display_name, $password, $role)
  Menambahkan sebuah *user* baru.
- add_users($text, $send_mail, $delay)
  Menambahkan *user* baru.
- delete_user($user_id)
  Menghapus sebuah *user*.
- delete_submissions($user_id)
  Menghapus seluruh *submission* dari sebuah *user*.
- validate_user($username, $password)
  Mengembalikan nilai TRUE jika $username dan $password valid untuk login.
- selected_assignment ($username)
  Mengembalikan *assignment* yang telah dipilih oleh *user*.

- get_names ()
  Mengembalikan nama dari *user*.
- update_profile ($user_id)
  Memperbarui sebuah *user*.
- send_password_reset_mail ($email)
  Mengirim email untuk *reset password*.
- passchange_is_valid ($passchange_key)
  Mengembalikan nilai TRUE jika kunci untuk *reset password valid*.
- reset_password ($passchange_key, $newpassword)
  Memperbarui *password* menjadi kunci *reset password*.
- get_all_users ()
  Mengambil seluruh *user*.
- get_user($user_id)
  Mengambil sebuah *user*.
- update_login_time ($username)
  Memperbarui catatan login sebuah *user*.

3.1.3.2 View

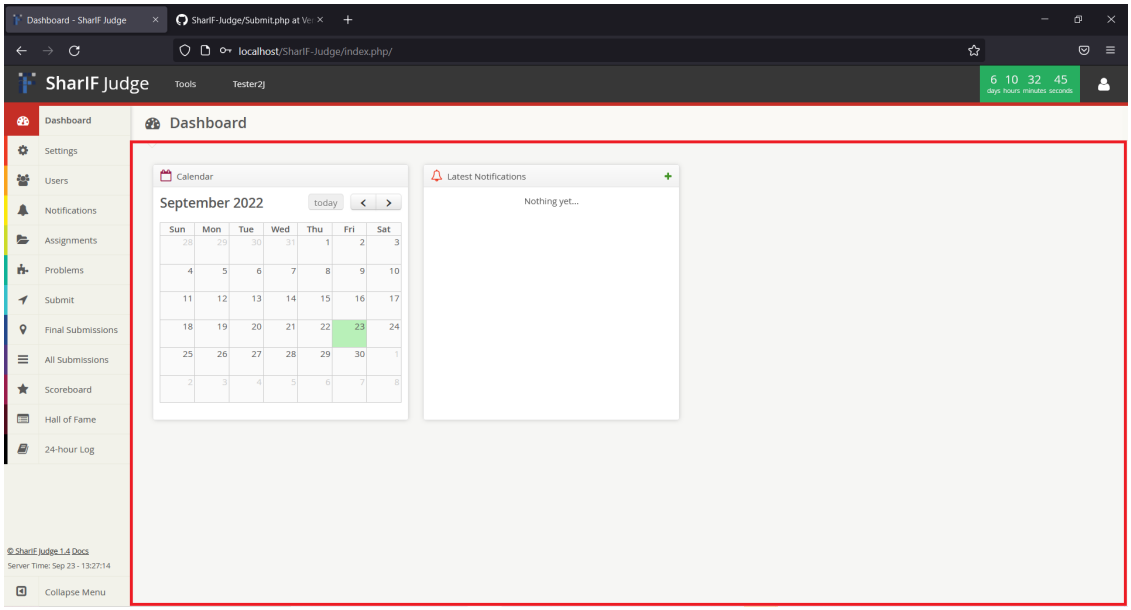
Berikut ini merupakan folder di dalam *View* yang terdapat pada SharIF-Judge:

- errors
  Menyimpan tampilan halaman error. Gambar 3.4 adalah contoh halaman error 404.



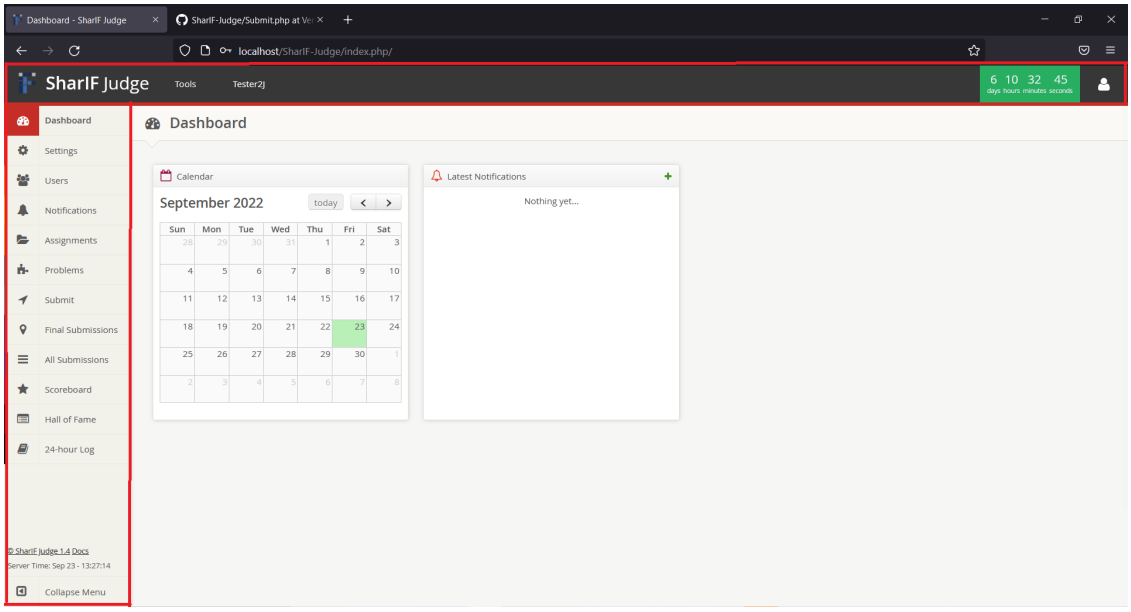
Gambar 3.4: Tampilan *error_404.php*

- pages
  Menyimpan komponen utama halaman. Pada gambar 3.5,yang telah ditandai warna merah merupakan komponen utama halaman *Dashboard*.



Gambar 3.5: Tampilan *dashboard.twig*

- templates
  Menyimpan komponen dasar halaman. Seperti pada gambar 3.6 yang telah ditandai warna merah menunjukan komponen dasar *side bar* dan *top bar*.



Gambar 3.6: Tampilan *side_bar.twig* dan *top_bar.twig*

3.1.3.3 Controller

Berikut ini merupakan controller pada SharIF Judge:

- Assignments
  Controller yang bertujuan untuk menangani *assignments*. Fungsi yang dimiliki:
  - select ()

Memilih *assignment* yang sedang ditampilkan.
- pdf ($assignment_id, $problem_id)
  Mengunduh *file* PDF dari sebuah *assignment*.
- downloadtestsdesc ($assignment_id)
  Mengunduh *file test case* dari sebuah *assignment*.
- download_submissions($type, $assignment_id)
  Mengunduh seluruh *file final submission* dari sebuah *assignment*.
- delete ($assignment_id)
  Menghapus sebuah *assignment*.
- add ()
  Menambah atau memperbarui *assignment*.
- edit($assignment_id)
  Memperbarui *assignment*.

- Dashboard
  Controller yang memiliki tujuan untuk menangani halaman Dashboard. Fungsi yang dimiliki:
  - widget_positions ()
    Menyimpan posisi *widget* dari *user*.
- Halloffame
  Controller yang bertujuan untuk menangani halaman Hall of Fame . Fungsi yang dimiliki:
  - hof_details ()
    Mengambil data yang diperlukan untuk *hall of fame*.
- Install
  Controller yang bertujuan untuk menangani instalasi SharIF Judge.
- Login
  Controller yang bertujuan untuk menangani halaman-halaman *login*. Fungsi yang dimiliki:
  - register ()
    Registrasi user baru dan menampilkan halaman *register*.
  - logout ()
    Log out user saat ini dan mengalihkan ke halaman *login*.
  - lost ()
    Menangani email dan menampilkan halaman untuk *reset password*.
  - reset($passchange_key)
    Memproses dan menampilkan halaman untuk ubah *reset password*.
- Logs
  Controller yang bertujuan untuk menangani halaman *24-hour Log*.
  - index ()
    Mengambil data yang diperlukan dan menampilkan halaman *24-hour Log*.
- Moss
  Controller yang bertujuan untuk menangani halaman *Detect Similar Codes*. Fungsi yang dimiliki:
  - update ($assignment_id) Memperbarui informasi pada halaman *Detect Similar Codes*.
  - _detect ($assignment_id) Menjalankan Moss untuk mendeteksi kesamaan kode.

- Notifications Controller yang bertujuan untuk menangani halaman *Notifications*. Fungsinya yaitu:
  - add ()
    Menambahkan notifikasi baru dan menampilkan halaman *New Notification*.
  - edit ($notif_id) Memperbarui sebuah notifikasi.
  - delete ()
    Menghapus sebuah notifikasi.
  - check () Memeriksa adanya notifikasi baru.
- Problems
  Controller yang bertujuan untuk menangani halaman Problems. Fungsinya yaitu:
  - index ($assignment_id, $problem_id = 1)
    Mengambil data yang diperlukan dan menampilkan halaman *Problems*.
  - edit ($type = 'md', $assignment_id, $problem_id = 1)
    Memperbarui deskripsi *problem* dan menampilkan halaman *Edit Problem Description*.
- Profile
  Controller yang bertujuan untuk menangani halaman *Profile*. Fungsi yang dimiliki:
  - index($user_id) Mengambil data yang diperlukan dan menampilkan halaman *Profile*.
  - _password_check ($str)
    Memeriksa apakah *password* sesuai dengan syarat.
  - _password_again_check ($str) // Memeriksa apakah *password again* sama dengan *password* yang dimasukkan.
  - _email_check ($email) Memeriksa apakah terdapat *user* dengan alamat email tertentu.
  - _role_check ($role) Memeriksa *role* yang dimiliki *user*.
- Queue
  Controller yang bertujuan untuk menangani halaman *Queue*. Fungsi yang dimiliki:
  - index ()
    Mengambil data yang diperlukan dan menampilkan halaman *Queue*.
  - pause ()
    Memberhentikan antrian.
  - resume ()
    Melanjutkan antrian.
  - empty_queue()
    Mengosongkan antrian.
- Queueprocess
  Controller yang bertujuan untuk menangani proses penilaian kode. Fungsinya yaitu:
  - run ()
    Menilai setiap kode satu demi satu dari antrian.
- Rejudge
  Controller yang bertujuan untuk menangani halaman *Rejudge*. Fungsinya yaitu:
  - index ()
    Mengambil data yang diperlukan dan menampilkan halaman *Rejudge*.
  - rejudge_single ()

Melakukan penilaian ulang untuk sebuah submission.

- Server_time Controller yang bertujuan untuk menangani sinkronisasi waktu server. Fungsinya yaitu:
  - index ()
    Mengembalikan waktu *server*.
- Submissions
  Controller yang bertujuan untuk menangani unduh *submissions* menjadi file excel. Fungsinya yaitu:
  - _download_excel ($view)
    Menggunakan *library* PHPExcel untuk membuat file excel.
  - final_excel ()
    Mengunduh data *final submissions* sebagai file excel.
  - all_excel ()
    Mengunduh data *final submissions* sebagai file excel.
  - the_final ()
    Mengambil dan menampilkan data *final submissions* yang akan diunduh.
  - all ()
    Mengambil dan menampilkan data *submissions* yang akan diunduh.
  - select ()
    Memilih *final submission*.
  - view_code ()
    Menampilkan kode, *result*, atau *log* dari *submission*.
  - download_file ()
    Mengunduh file excel.
- Submit
  Controller yang bertujuan untuk menangani submissions. Fungsinya yaitu:
  - _language_to_type ($language)
    Mengembalikan kode singkatan dari bahasa pemrograman.
  - _match ($type, $extension)
    Memeriksa apakah bahasa pemrograman dan tipe file sesuai.
  - _check_language ($str)
    Memeriksa apakah bahasa pemrograman yang dipilih *valid*.
  - index ()
    Mengambil data yang diperlukan dan menampilkan halaman *Submit*.
  - _upload ()
    Menyimpan file yang diunggah dan menambahkannya ke dalam antrian.
- Users
  Controller yang bertujuan untuk menangani halaman *Users*. Fungsinya yaitu:
  - index ()
    Mengambil data yang diperlukan dan menampilkan halaman *Users*.
  - add ()
    Menambah *user* baru dan menampilkan halaman *Add Users*.

- delete ()
  Menghapus *user*.
- delete_submissions ()
  Menghapus seluruh *submission* dari sebuah *user*.
- list_excel ()
  Menggunakan *library* PHPExcel untuk membuat file excel dari *list user*.

3.1.4 Clean URLs

Secara default, index.php merupakan bagian dari seluruh URLs yang ada pada Sharif Judge. Berikut merupakan contoh URLs yang dihasilkan oleh SharIF-Judge:

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

Pengguna dapat menghilangkan index.php di atas dan memiliki URLs yang baik jika sistem pengguna mendukung URL rewriting. URL rewriting adalah proses mengubah parameter yang terdapat pada URL. Berikut contoh URL yang telah melewati proses URL rewriting:

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

Untuk menggunakan clean urls, pengguna perlu melakukan beberapa perubahan terlebih dahulu:

- Mengubah nama file .htaccess2 menjadi .htaccess yang berlokasi di direktori utama SharIF-Judge.
- Mengubah $config[‘index_page’] = ‘index.php’; menjadi $config[‘index_page’] = ”; pada file application/config/config.php.

3.1.5 Deteksi Kecurangan

SharIF-Judge menggunakan *Moss* (*Measure of Software Similarity*) untuk mendeteksi kode yang mirip dengan menggunakan sistem otomatis untuk menentukan kemiripan program. Pada saat ini, aplikasi utama *Moss* telah digunakan untuk mendeteksi plagiarisme pada kelas *programming*. Pengguna dapat menggunakan deteksi kecurangan ini dengan cara mengirimkan kode yang dikumpulkan oleh peserta ke server *Moss*. 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 mendaftar pada halaman http: //theory.stanford.edu/ aiken/moss/. Pengguna akan mendapatkan sebuah email yang berisikan script perl dan *Moss user id* berada pada script tersebut. Berikut merupakan potongan skrip dengan bahasa pemrograman *perl* yang berisikan *user id*.

```
...

$server = 'moss.stanford.edu';
$port = '7690';
```

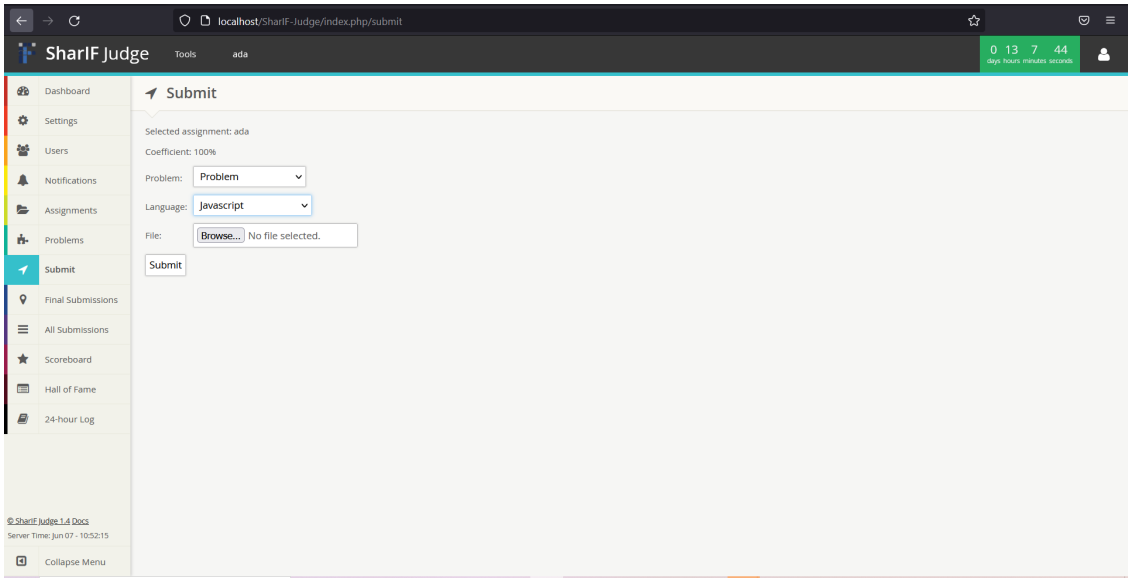
```
$noreq = "Request not sent.";
$usage = "usage: moss [-x] [-l language] [-d] [-b basefile1] ... [-b
    basefilen] [-m #] [-c \"string\"] file1 file2 file3 ...";

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

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

...
```

User id yang terdapat pada potongan perl script di atas, dapat digunakan pada SharIF-Judge untuk mendeteksi kecurangan. Pengguna dapat menyimpan *user id* di SharIF-Judge pada halaman *Moss*. SharIF-Judge akan menggunakan *user id* tersebut di *Moss perl script*. *Perl* harus terinstal pada server agar dapat menggunakan *Moss*.

- Pengguna dianjurkan untuk mendeteksi kode yang mirip setelah waktu *assignment* berakhir, karena para peserta masih dapat mengubah *Final Submissions* masing-masing sebelum waktu habis. Dengan cara tersebut, SharIF-Judge dapat mengirimkan *Final submissions* para peserta ke server *Moss*.

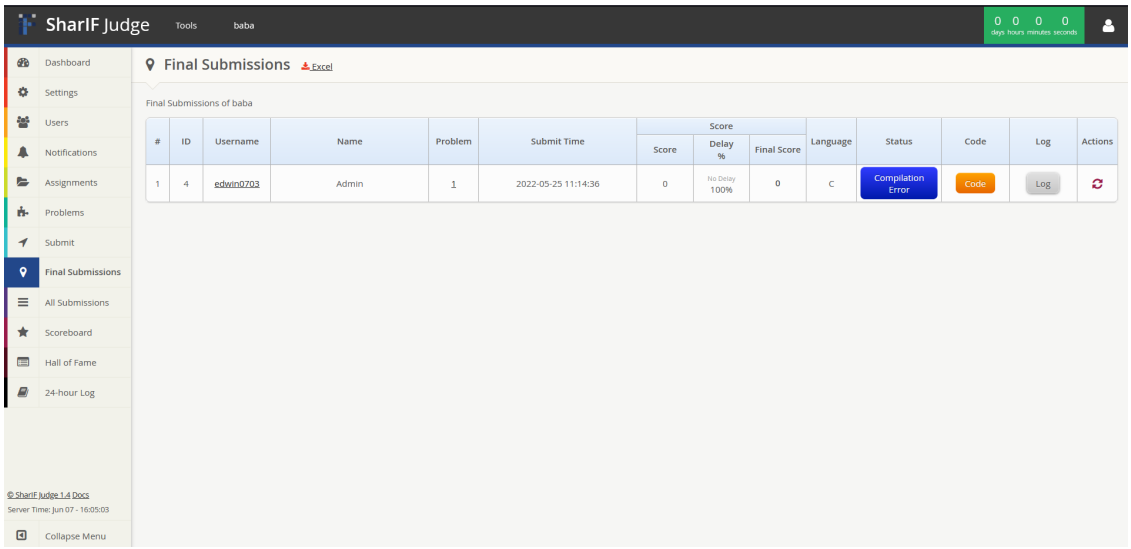
3.1.6 *Submit*

Pada Gambar 3.7, terdapat 4 fungsi yaitu *problem*, *Language*, *Upload File*, dan tombol *submit*. *Problem* berfungsi untuk memilih permasalahan mana yang ingin dinilai. *Language* berfungsi untuk memilih dengan bahasa pemrograman apa penilaian *Problem* akan diperiksa. *Upload File* berfungsi untuk mengirimkan file yang akan diperiksa oleh SharIF-Judge. *Submit* berfungsi untuk mengirimkan permintaan untuk diperiksa oleh SharIF-Judge sesuai dengan ketentuan *problem*, *language*, dan file yang dimasukan peserta.



Gambar 3.7: Halaman *Submit*

3.1.7 *Final Submission*

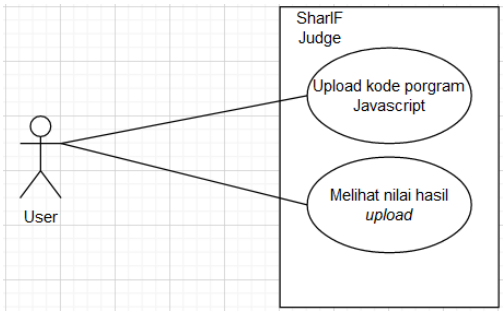


Gambar 3.8: Halaman *Final Submission*

Pada Gambar 3.8 Menunjukan halaman *Final Submission*. Pada halaman ini, terdapat daftar seluruh *final submission* pada *assignment* yang telah dipilih oleh peserta. Pengguna juga dapat melihat file atau kode program yang diunggah oleh peserta dan bahkan melihat nilai yang didapatkannya.

3.2 Analisis Sistem Usulan

Agar SharIF Judge dapat menerima, membaca, dan mengevaluasi *Javascript* maka diperlukan adanya kostumisasi pada beberapa fungsi yang ada di SharIF-Judge. Pada gambar 3.9, merupakan *Use Case* Diagram sebagai tambahan fungsi yang dapat dilakukan oleh *user*. Fitur-fitur yang akan ditambahkan, dapat digunakan oleh setiap *role*.



Gambar 3.9: *Use Case Diagram Fitur Usulan*

Berikut ini merupakan beberapa penjelasan untuk fitur tambahannya:

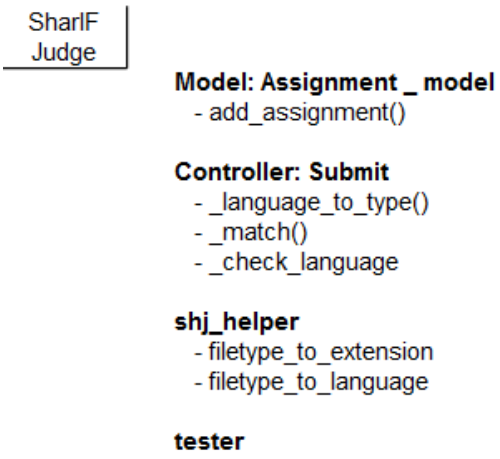
- Upload kode program *javascript*
  Adanya kostumisasi fitur agar pengguna dapat memilih bahasa javascript, sehingga pengguna dapat melakukan upload kode program javascript.
  - Aktor: Pengguna
  - Syarat:
    1. Memilih javascript sebagai bahasa pilihan
    2. memiliki file javascript
  - Skenario:
    1. Pada halaman *Add Assignment* bagian *Allowed Language*, *Role Admin* dan *Head Instructor* menambahkan *javascript*
    2. Pada halaman *submit* bagian *Language*, *User* memilih *javascript*
    3. pada bagian file, *User* memasukan file *javascript*
    4. *User* click tombol submit.
- Melihat nilai hasil *upload*
  Adanya kostumisasi fitur agar file yang telah *user upload* dapat dinilai sehingga nilai dapat dilihat oleh *User*.
  - Aktor : Pengguna
  - Syarat:
    1. Memilih *javascript* sebagai bahasa pilihan.
    2. Memiliki file *javascript*
    3. Sudah *upload* file *javascript*.
    4. Upload file *javascript* berhasil.
  - Skenario:
    1. Pada halaman *Add Assignment* bagian *Allowed Language*, *Role Admin* dan *Head Instructor* menambahkan *javascript*
    2. Pada halaman *submit* bagian *Language*, *User* memilih *javascript*
    3. pada bagian file, *User* memasukan file *javascript*
    4. *User* click tombol submit.
    5. *User* akan dipindah ke halaman *All Submission* secara otomatis dan melihat apakah file yang dipilih telah berhasil diunggah.
    6. *refresh* halaman *All submission* dan *User* akan dapat melihat nilainya.

BAB 4

PERANCANGAN

Bab ini akan membahas perancangan perubahan kode program untuk fitur yang diimplementasi pada perangkat lunak SharIF Judge.

4.1 Rancangan Perubahan Kode Program



Gambar 4.1: *List* yang akan dikostumisasi pada SharIF-Judge

Agar fitur-fitur tambahan dapat digunakan, diperlukan adanya kostumisasi kode pada SharIF Judge. Pada gambar 4.1, merupakan fungsi-fungsi yang akan mengalami perubahan.

4.1.1 Memilih bahasa JavaScript

SharIF-Judge sudah tersedia berbagai bahasa seperti Java, Python 2, Python 3, C, dan C++ dan *user* dapat memilihnya dengan kondisi dimana pada saat melakukan *add assignment* pada bagian *allowed language*, bahasa tersebut disertakan. Namun ketika javascript ditambahkan pada *allowed language*, *user* belum dapat memilihnya pada saat ingin melakukan *submit*. Agar *user* dapat memilih javascript, dilakukan perubahan sebagai berikut:

- *Model* Assignment_model
  - Fungsi add_assignment()
    Penambahan *script* ditujukan agar ketika dilakukan *add assignment* dan memasukan JavaScript pada *allowed language*, dan ketika *user* ingin memilih language pada halaman

submit, *dropdown language* memiliki pilihan JavaScript dengan kostumisasi kode program sebagai berikut

```
Sebelum


if ( !in_array($item2, array('c','c++','python 2','python 3','java','zip
    ','pdf','txt')))
        continue;

// If the problem is not Upload-Only, its language should be one of {C,C
    ++,Python 2, Python 3,Java}

if ( !in_array($i, $uo) && ! in_array($item2, array('c','c++','python
    2','python 3','java')) )
        continue;
```

```
Sesudah

if ( !in_array($item2, array('c','c++','python 2','python 3','java','zip
    ','pdf','txt','javascript')))
        continue;

// If the problem is not Upload-Only, its language should be one of {C,C
    ++,Python 2, Python 3,Java, JavaScript}

if ( !in_array($i, $uo) && ! in_array($item2, array('c','c++','python
    2','python 3','java','javascript')) )
        continue;
```

4.1.2 Unggah file JavaScript

Ketika *user* memilih *language* JavaScript dan mengunggah file .js, *user* akan dikirmkan ke *error page*. Hal tersebut terjadi karena SharIF Judge belum memahami adanya file .js sehingga diperlukan adanya kostumisasi sebagai berikut:

- *Controller* Submit
  - Fungsi _language_to_type()

    Penambahan *script* ditujukan agar ketika *user* memilih *language* JavaScript, SharIF Judge akan memahami bahwa tipe file JavaScript adalah .js. dengan kostumisasi kode program sebagai berikut:

```
Sebelum


    public function _language_to_type($language)
    {
```

```
        $language = strtolower ($language);
        switch ($language) {
            case 'c': return 'c';
            case 'c++': return 'cpp';
            case 'python 2': return 'py2';
            case 'python 3': return 'py3';
            case 'java': return 'java';
            case 'zip': return 'zip';
            case 'pdf': return 'pdf';
            case 'txt': return 'txt';
            default: return FALSE;
        }
    }
```

Sesudah

```
    public function _language_to_type($language)
    {
        $language = strtolower ($language);
        switch ($language) {
            case 'c': return 'c';
            case 'c++': return 'cpp';
            case 'python 2': return 'py2';
            case 'python 3': return 'py3';
            case 'java': return 'java';
            case 'zip': return 'zip';
            case 'pdf': return 'pdf';
            case 'txt': return 'txt';
            case 'javascript': return 'js';
            default: return FALSE;
        }
    }
```

– Fungsi _check_language()

Penambahan *script* ditujukan agar ketika memilih bahasa JavaScript dan mengunggah kode program JavaScript, SharIF-Judge memahami bahwa file yang diunggah sudah sesuai dengan *language* yang dipilih dengan kostumisasi kode program sebagai berikut:

Sebelum

```
    public function _check_language($str)
    {
        if ($str=='0')
            return FALSE;
```

```
        if (in_array( strtolower($str),array('c', 'c++', 'python 2', '
            python 3', 'java', 'zip', 'pdf', 'txt')))
            return TRUE;
        return FALSE;
    }
```

```
Sesudah

    public function _check_language($str)
    {
        if ($str=='0')
            return FALSE;
        if (in_array( strtolower($str),array('c', 'c++', 'python 2', '
            python 3', 'java', 'zip', 'pdf', 'txt', 'javascript')))
            return TRUE;
        return FALSE;
    }
```

– Fungsi _match()

Penambahan *script* ditujukan untuk memastikan bahwa ketika *user* memilih *language* JavaScript, file .js dapat diterima dan dapat diunggah dengan kostumisasi kode program sebagai berikut:

```
Sebelum

public function _match($type, $extension)
    {
        switch ($type) {
            case 'c': return ($extension==='c'?TRUE:FALSE);
            case 'cpp': return ($extension==='cpp'?TRUE:FALSE);
            case 'py2': return ($extension==='py'?TRUE:FALSE);
            case 'py3': return ($extension==='py'?TRUE:FALSE);
            case 'java': return ($extension==='java'?TRUE:FALSE);
            case 'zip': return ($extension==='zip'?TRUE:FALSE);
            case 'pdf': return ($extension==='pdf'?TRUE:FALSE);
            case 'txt': return ($extension==='txt'?TRUE:FALSE);
        }
    }
```

```
Sesudah

public function _match($type, $extension)
    {
        switch ($type) {
```

```
            case 'c': return ($extension==='c'?TRUE:FALSE);
            case 'cpp': return ($extension==='cpp'?TRUE:FALSE);
            case 'py2': return ($extension==='py'?TRUE:FALSE);
            case 'py3': return ($extension==='py'?TRUE:FALSE);
            case 'java': return ($extension==='java'?TRUE:FALSE);
            case 'zip': return ($extension==='zip'?TRUE:FALSE);
            case 'pdf': return ($extension==='pdf'?TRUE:FALSE);
            case 'txt': return ($extension==='txt'?TRUE:FALSE);
            case 'js': return ($extension==='js'?TRUE:FALSE);
        }
    }
```

4.1.3 Melihat bahasa yang diunggah

Ketika *user* selesai mengunggah file dan menekan tombol *submit*, *user* akan dikirimkan ke halaman *All Submission*. Namun jika *user* mengunggah file javascript, pada kolom *Language* akan kosong sehingga diperlukan adanya kostumisasi sebagai berikut:

- *helpers* shj_helper
  - Fungsi filetype_to_language()

    Penambahan *script* ditujukan agar pada kolom bagian *Language* muncul bahwa bahasa yang digunakan adalah JavaScript dengan kostumisasi kode program sebagai berikut:

```
Sebelum

function filetype_to_language($file_type)
    {
        $file_type = strtolower($file_type);
        switch ($file_type) {
            case 'c': return 'C';
            case 'cpp': return 'C++';
            case 'py2': return 'Py 2';
            case 'py3': return 'Py 3';
            case 'java': return 'Java';
            case 'zip': return 'Zip';
            case 'pdf': return 'PDF';
            case 'txt': return 'TXT';
            default: return FALSE;
        }
    }
```

```
Sesudah

function filetype_to_language($file_type)
```

```
    {
        $file_type = strtolower($file_type);
        switch ($file_type) {
            case 'c': return 'C';
            case 'cpp': return 'C++';
            case 'py2': return 'Py 2';
            case 'py3': return 'Py 3';
            case 'java': return 'Java';
            case 'zip': return 'Zip';
            case 'pdf': return 'PDF';
            case 'txt': return 'TXT';
            case 'js': return 'Js';
            default: return FALSE;
        }
    }
```

4.1.4 Melihat kode program yang diunggah

Ketika *user* selesai mengunggah file dan menekan tombol *submit*, *user* akan dikirimkan ke halaman *All Submission*. pada kolom *Code* jika *user* mengunggah file javascript, *user* tidak dapat melihat kode program yang diunggah sehingga diperlukan adanya kostumisasi sebagai berikut:

- *helpers* shj_helper
  - Fungsi filetype_to_extension()

    Penambahan *script* ditujukan agar ketika menekan tombol *Code* pada kolom *Code* akan muncul kode program yang diunggah dengan kostumisasi kode program sebagai berikut.

```
Sebelum


    function filetype_to_extension($file_type)
    {
        $file_type = strtolower($file_type);
        switch ($file_type) {
            case 'c': return 'c';
            case 'cpp': return 'cpp';
            case 'py2': return 'py';
            case 'py3': return 'py';
            case 'java': return 'java';
            case 'zip': return 'zip';
            case 'pdf': return 'pdf';
            case 'txt': return 'txt';
            default: return FALSE;
        }
    }
```

Sesudah

```
    function filetype_to_extension($file_type)
    {
        $file_type = strtolower($file_type);
        switch ($file_type) {
            case 'c': return 'c';
            case 'cpp': return 'cpp';
            case 'py2': return 'py';
            case 'py3': return 'py';
            case 'java': return 'java';
            case 'zip': return 'zip';
            case 'pdf': return 'pdf';
            case 'txt': return 'txt';
            case 'js': return 'js';
            default: return FALSE;
        }
    }
```

4.1.5 Pengecekan kode program

Ketika user selesai mengunggah file dan menekan tombol submit, kode program yang dikirim akan diperiksa untuk dilihat apakah kode tersebut terdapat kesalahan dalam penulisannya secara JavaScript atau tidak. Jika terdapat kesalahan, akan diperlihatkan dimana kesalahan penulisannya. namun jika tidak akan dicek input dan outputnya. sehingga diperlukan adanya penambahan kode program sebagai berikut:

```
if [ "$EXT" = "js" ]; then
    cp ../javascript.policy javascript.policy
    cp $PROBLEMPATH/$UN/$FILENAME.js $MAINFILENAME.js
    shj_log "Compiling as Javascript"
    node --check $MAINFILENAME.js >/dev/null 2>cerr
    EXITCODE=$?
    COMPILE_END_TIME=$(($(date +%s%N)/1000000));
    shj_log "Compiled. Exit Code=$EXITCODE Execution Time: $((
        COMPILE_END_TIME-COMPILE_BEGIN_TIME)) ms"
    if [ $EXITCODE -ne 0 ]; then
        shj_log "Compile Error"
        shj_log "$(cat cerr|head -10)"
        echo '<span class="shj_b">Compile Error</span>' >$PROBLEMPATH/$UN/
            result.html
        echo '<span class="shj_r">' >> $PROBLEMPATH/$UN/result.html
```

```
        #filepath="$(echo "${JAIL}/${FILENAME}.${EXT}" | sed 's///\//g')" #
            replacing / with /
        (cat cerr | head -10 | sed 's/&/&amp;/g' | sed 's/</&lt;/g' | sed 's
            />/&gt;/g' | sed 's/"/&quot;/g') >> $PROBLEMPATH/$UN/result.html
        #(cat $JAIL/cerr) >> $PROBLEMPATH/$UN/result.html
        echo "</span>" >> $PROBLEMPATH/$UN/result.html
        cd ..
        rm -r $JAIL >/dev/null 2>/dev/null
        shj_finish "Compilation Error"
    fi
fi
```

4.1.6 Penilaian Kode Program

Ketika user selesai mengunggah file dan menekan tombol submit, dan setelah kode program yang dikirim tersebut telah diperiksa dan tidak terdapat kesalahan dalam penulisannya secara JavaScript. Kode program akan dicek apakah *input* dan *output*nya sudah sesuai. Jika sudah maka akan muncul nilai dari user tersebut sehingga diperlukan adanya penambahan kode program sebagai berikut:

```
elif [ "$EXT" = "js" ]; then
        if $PERL_EXISTS; then
            ./runcode.sh $EXT $MEMLIMIT $TIMELIMIT $TIMELIMITINT $PROBLEMPATH/
                in/input$i.txt "./timeout --just-kill -nosandbox -l $OUTLIMIT -
                t $TIMELIMIT -m $MEMLIMIT javascript -O $FILENAME.js"
        else
            ./runcode.sh $EXT $MEMLIMIT $TIMELIMIT $TIMELIMITINT $PROBLEMPATH/
                in/input$i.txt "javascript -O $FILENAME.js"
        fi
        EXITCODE=$?
```

DAFTAR REFERENSI

[1] Kurnia, A., Lim, A., dan Cheang, B. (2001) Online judge. *Computers & Education*, **36**, 299–315.

[2] Crockford, D. (2008) *JavaScript: The Good Parts: The Good Parts*. " O'Reilly Media, Inc.", Gravenstein Highway North.

[3] Pluralsight (2016) Learn javascript. https://www.javascript.com/learn. 10 January 2022.

[4] for Geeks, G. (2021) Node.js readline() module. https://www.geeksforgeeks.org/node-js-readline-module/. 08 June 2022.

[5] Foundation, C. (2019) Codeigniter3 user guide. https://codeigniter.com/userguide3/. 10 January 2022.

[6] GNU (2021) Gnu bash. https://www.gnu.org/software/software.html. 10 Januari 2022.

[7] Foundation, O. (2011) Node.js. https://nodejs.org/en/. 22 April 2022.

[8] Hub, S. (2010) The flexible, fast, and secure template engine for php. https://twig.symfony.com/. 09 May 2022.

[9] Codequiry (2019) Moss. https://codequiry.com/moss/measure-of-software-similarity. 07 May 2022.

LAMPIRAN A

KODE PROGRAM

Kode A.1: MyCode.c

```c

// This does not make algorithmic sense,
// but it shows off significant programming characters.

#include<stdio.h>

void myFunction( int input, float* output ) {
    switch ( array[i] ) {
        case 1: // This is silly code
            if ( a >= 0 || b <= 3 && c != x )
                *output += 0.005 + 20050;
            char = 'g';
            b = 2^n + ~right_size - leftSize * MAX_SIZE;
            c = (--aaa + &daa) / (bbb++ - ccc % 2 );
            strcpy(a,"hello $@?");
    }
    count = ~mask | 0x00FF00AA;
}

// Fonts for Displaying Program Code in LATEX
// Adrian P. Robson, nepsweb.co.uk
// 8 October 2012
// http://nepsweb.co.uk/docs/progfonts.pdf
```

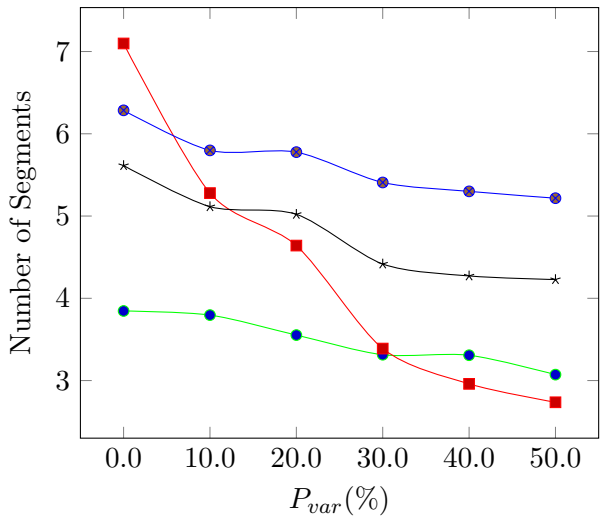
Kode A.2: MyCode.java

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;

//class for set of vertices close to furthest edge
public class MyFurSet {
    protected int id;                                   //id of the set
    protected MyEdge FurthestEdge;                      //the furthest edge
    protected HashSet<MyVertex> set;                    //set of vertices close to furthest edge
    protected ArrayList<ArrayList<Integer>> ordered;    //list of all vertices in the set for each trajectory
    protected ArrayList<Integer> closeID;               //store the ID of all vertices
    protected ArrayList<Double> closeDist;              //store the distance of all vertices
    protected int totaltrj;                             //total trajectories in the set

    /*
     * Constructor
     * @param id : id of the set
     * @param totaltrj : total number of trajectories in the set
     * @param FurthestEdge : the furthest edge
     */
    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
        this.id = id;
        this.totaltrj = totaltrj;
        this.FurthestEdge = FurthestEdge;
        set = new HashSet<MyVertex>();
        ordered = new ArrayList<ArrayList<Integer>>();
        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
        closeID = new ArrayList<Integer>(totaltrj);
        closeDist = new ArrayList<Double>(totaltrj);
        for (int i = 0;i <totaltrj;i++) {
            closeID.add(-1);
            closeDist.add(Double.MAX_VALUE);
        }
    }

}
```

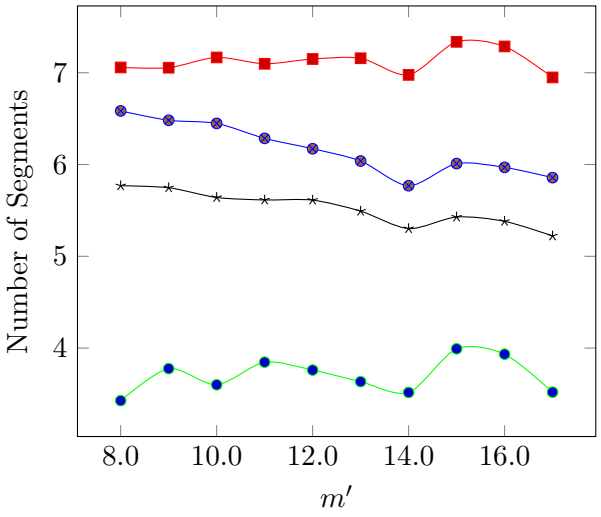
LAMPIRAN B

HASIL EKSPERIMEN

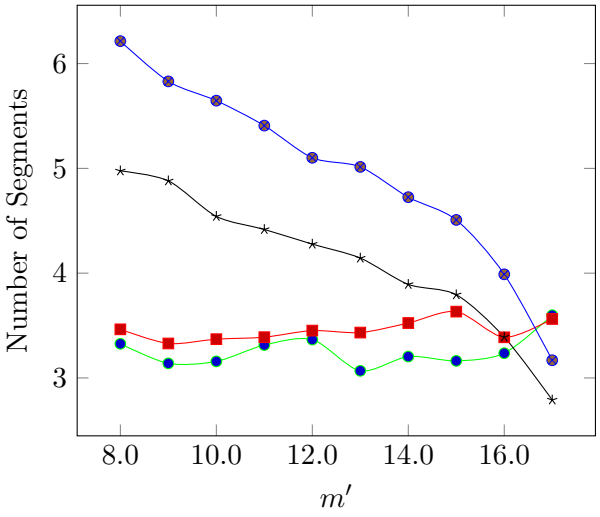
Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



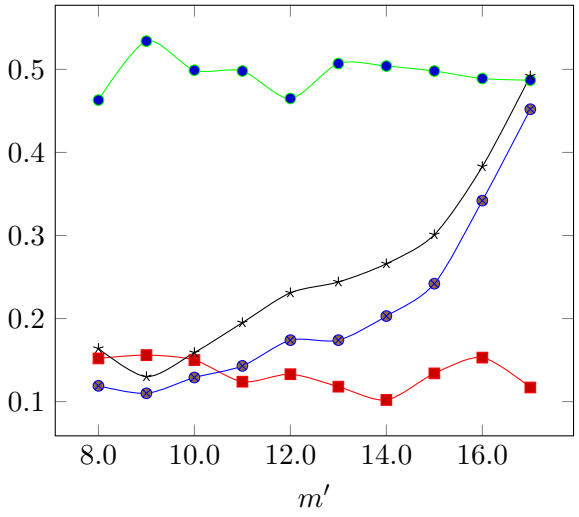
Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4