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 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

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

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

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:

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
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

JavaScript merupakan bahasa pemrograman berorientasi objek berdasarkan model objek berbasis prototipe[3]. Bahasa ini terkenal karena penggunaannya sebagai bahasa scripting di web. JavaScript dianggap sebagai blok pembangun utama HTML yang Dinamis dimana kumpulan teknologi yang disertakan pada hampir semua browser web dalam mendukung pembuatan situs web animasi dan interaktif. Saat terintegrasi di dalam browser web, implementasi JavaScript menyertakan sekumpulan pustaka yang secara kolektif disebut sebagai "*Client-Side JavaScript*" Sebaliknya, bahasa JavaScript dan pustaka inti JavaScript (yaitu, pustaka yang mandiri dari web) biasanya disebut sebagai "*Core JavaScript*". Berikut ini merupakan kelebihan yang dimiliki oleh *JavaScript* [1] :

- Kecepatan
  JavaScript merupakan bahasa yang "ditafsirkan" dan hal ini dapat mengurangi waktu yang dibutuhkan oleh bahasa pemrograman lain seperti java untuk kompilasi
- *Versatility*
  JavaScript sekarang mampu pengembangan front-end serta back-end. Pengembangan back-end menggunakan NodeJS sementara banyak *library* membantu dalam pengembangan front-end seperti AngularJS, ReactJS, dll.
- *Server Load*
  Karena JavaScript beroperasi di sisi klien, validasi data dapat dilakukan pada browser tersebut tanpa mengirimkannya lagi ke server. Jika ada perbedaan, seluruh situs web tidak perlu dimuat ulang. Browser hanya perlu memperbarui segmen halaman yang dipilih.
- Popularitas
  Browser modern mendukung JavaScript dan banyak perusahaan terkenal menggunakan JavaScript termasuk Google, Amazon, PayPal, dll.

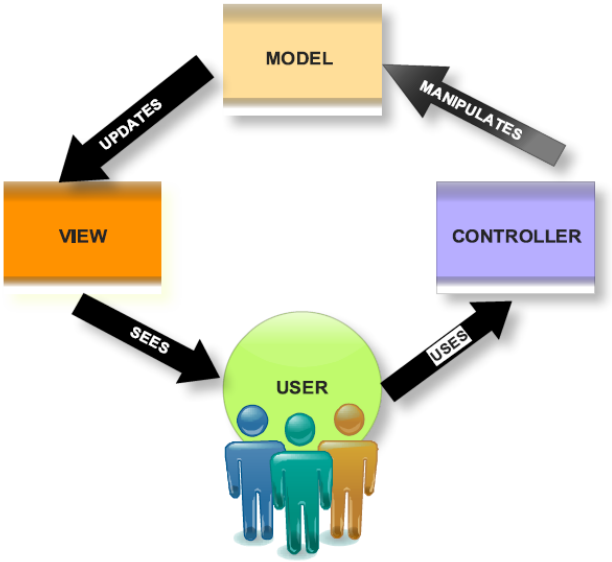
2.2 CodeIgniter

CodeIgniter merupakan sebuah framework bagi pengguna yang ingin membangun web application dengan menggunakan bahasa pemrograman berbasis PHP[4]. 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. CodeIgniter versi publik pertama kali dirilis pada 28 Februari 2006, dan versi stabil terbaru v4.1.9 yang dirilis pada 24 Februari 2020 yaitu CodeIgniter 4

[1] https://data-flair.training/blogs/advantages-disadvantages-javascript/

2.2.1 *Model-View-Controller*

Model-View-Controller (*MVC*) merupakan pola yang digunakan oleh CodeIgniter[2] . *MVC* adalah pendekatan perangkat lunak yang memisahkan logika aplikasi dari presentasi. Dalam praktiknya, ini memungkinkan web halaman pengguna berisi skrip yang minimal karena presentasinya terpisah dari skrip PHP. Pengoperasian metode Model View Controller (MVC) memiliki tujuan untuk menumpahkan atau memisahkan bagian kode yang berbeda ke dalam lapisan seperti tampilan, akses data, mengontrol permintaan pengguna dan meneruskan permintaan ke lapisan yang relevan[4]. MVC merupakan kumpulan dari tiga bagian: Model, View, dan Controller.



Gambar 2.1: Perputaran *Model-View-Controller*

Gambar 2.1 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.

2.2.1.1 *Model*

Model bertujuan untuk memberikan penyimpanan permanen data yang digunakan dalam desain keseluruhan [4]. Hal tersebut mewajibkan model untuk dapat mengakses data untuk dilihat, atau dikumpulkan dan ditulis, dan merupakan jembatan antara komponen View dan komponen Controller. Salah satu aspek penting dari Model adalah bahwa secara teknis "buta". Dengan ini, *Model* tidak memiliki koneksi atau pengetahuan tentang apa yang terjadi pada data ketika diteruskan ke komponen *View* atau *Controller*. *Model* tidak memanggil atau mencari tanggapan dari bagian lain dari komponen. Tujuan utamanya adalah untuk mengolah data menjadi penyimpanan permanennya, mencari dan menyiapkan data untuk diteruskan ke bagian lain.

Model tidak dapat dianggap hanya sebagai perangkat database saja, atau pintu gerbang ke sistem lain yang menangani proses data. *Model* mewakili penjaga gerbang ke data itu sendiri, tidak mengajukan pertanyaan tetapi menerima semua permintaan yang datang. Seringkali bagian paling kompleks dari sistem MVC ini, komponen *Model* juga merupakan puncak dari keseluruhan sistem karena tanpanya tidak akan ada koneksi antara *Controller* dan *View* .

[2] https://www.codeigniter.com/userguide3/overview/mvc.html

2.2.1.2 *View*

View merupakan informasi yang diminta dari *Model* dan ditampilkan kepada pengguna [4]. Secara tradisional, dalam aplikasi web menggunakan MVC untuk pengembangan, View adalah bagian dari sistem dimana HTML dihasilkan dan ditampilkan. View juga memicu reaksi dari pengguna yang kemudian berinteraksi dengan *controller*. Contoh dasar dari ini adalah tombol yang dihasilkan oleh *view*, yang di klik oleh pengguna dan memicu tindakan pada *controller*.

Namun terkadang beberapa orang memiliki konsep yang salah terhadap *View*, terutama oleh pengguna yang mengembangkan web dengan menggunakan pola MVC untuk membangun aplikasi mereka. Sebagai contoh, banyak yang salah mengira bahwa *View* tidak memiliki koneksi apa pun dengan *Model* dan bahwa semua data yang ditampilkan oleh *View* dilewatkan dari *Controller*. Pada kenyataannya, aliran ini mengabaikan teori di balik pola MVC sepenuhnya.Untuk menerapkan arsitektur MVC dengan benar, tidak boleh ada interaksi antara *Model* dengan *View*. Semua logika hanya ditangani oleh *Controller*. Selain itu, deskripsi *View* sebagai file template pun tidak tepat. *View* merupakan sesuatu yang lebih dari sekadar template. Kerangka kerja yang terinspirasi MVC modern telah merusak tampilan hampir ke titik di mana kurangnya kepedulian apakah kerangka kerja benar-benar mematuhi pola MVC yang benar atau tidak. Penting juga untuk menyebutkan bahwa bagian *View* tidak pernah diberikan data oleh *Controller*. Tidak ada hubungan langsung antara *View* dan *Controller* tanpa *Model* di antara keduanya.

2.2.1.3 *Controller*

Komponen ketiga dari MVC adalah *Controller*. Tugasnya adalah menangani data yang dikirimkan pengguna serta memperbarui *Model* yang sesuai [4]. *Controller* dapat disimpulkan sebagai pengumpul informasi, yang kemudian diteruskan ke *Model* untuk diatur penyimpanannya, dan tidak mengandung logika apa pun selain mengumpulkan masukan dari pengguna. *Controller* juga hanya terhubung ke *View* tunggal dan *Model* tunggal, menjadikannya sistem aliran data satu arah, dengan jabat tangan dan tanda tangan di setiap titik pertukaran data. *Controller* hanya diberikan tugas untuk dilakukan saat pengguna sedang berinteraksi dengan *View* dan setiap fungsi *Controller* adalah sebuah pemicu, yang dipicu oleh interaksi antara pengguna dengan *View*. Kesalahan paling umum yang dibuat oleh pengembang adalah menganggap *Controller* sebagai *gateway*, dan akhirnya menetapkan fungsi dan tanggung jawab yang seharusnya dilakukan oleh *View* (ini biasanya merupakan hasil dari pengembang yang sama yang mengacaukan komponen *View* sebagai template). Selain itu, adalah kesalahan umum untuk menetapkan fungsi *Controller* yang memberikan tanggung jawab tunggal untuk mengolah, meneruskan, dan memproses data dari *Model* ke *View*. Meskipun demikian, hubungan pola MVC harus dijaga antara *Model* dan *View*.

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)[3]. 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

[3] https://www.gnu.org/software/bash/

2.4 SharIF-Judge

Sharif Judge adalah online judge gratis untuk bahasa pemrograman C, C++, Java dan Python [4]. 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. erikut 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 2.1: Tabel *role*

Role	Level
Admin	3
Head Instructor	2
Instructor	1
Student	0

Pada Tabel 2.1 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 Tabel 2.2.

Tabel 2.2: Tabel *Action*

Action	*Admin*	*Head Instructor*	*Instructor*	*Student*
Mengubah *Setting*	O	X	X	X
Menambah/Menghapus *User*	O	X	X	X
Mengubah *role users*	O	X	X	X
Menambah/Menghapus/Mengubah *Assignment*	O	O	X	X
Mengunduh *Test*	O	O	X	X
Menambah/Menghapus/Mengubah Notifikasi	O	O	X	X
Rejudge	O	O	X	X
Melihat/Pause/Melanjutkan/Submission Queue	O	O	X	X
Mendeteksi Kode yang Mirip	O	O	X	X
Melihat Semua Kode	O	O	O	X
Mengunduh Kode Final	O	O	O	X
Memilih *Assignment*	O	O	O	O
Submit	O	O	O	O

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
```

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

```
    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*
  Sandboxing adalah 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*
  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[5].

- 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.

[5] https://codequiry.com/moss/measure-of-software-similarity

- 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.

2.4.1 Instalasi

Untuk dapat menjalankan Sharif Judge, diwajibkan untuk memiliki server Linux dan mengikuti tahapan sebagai berikut[6]:

- 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 databse*.
- 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
/*******************************************/
```

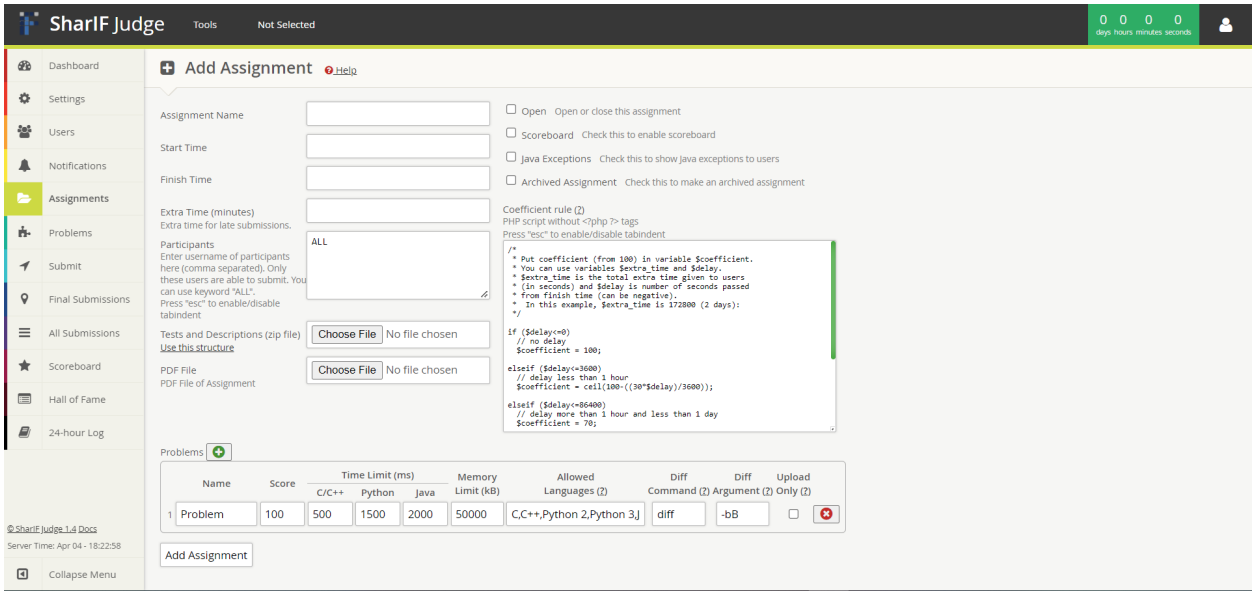
- Membuat direktori application/cache/Twig agar dapat ditulis oleh PHP
- Membuka halaman utama Sharif Judge pada web browser
- *Log in* menggunakan akun *admin*

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

- 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.

2.4.2 *Add Assignment*

Pengguna dapat menambahkan *assignment* dengan cara mengklik menu *Assignments* lalu klik *add* pada halaman tersebut[6]. Pada Gambar 2.2 merupakan halaman *add assignment*. Berikut ini



Gambar 2.2: 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
```

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

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

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

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

Perhatikan bawa berbeda dengan penempatan judul gambar gambar, keterangan tabel harus diletakkan di atas tabel!! Lihat Tabel 2.3 berikut ini:

Tabel 2.3: Tabel contoh

	v_{start}	$\mathcal{S}_1$	v_{end}
τ_1	1	12	20
τ_2	1		20
τ_3	1	9	20
τ_4	1		20

Tabel 2.4 dan Tabel 2.5 berikut ini adalah tabel dengan sel yang berwarna dan ada dua tabel yang bersebelahan.

Tabel 2.4: Tabel bewarna(1)

	v_{start}	$\mathcal{S}_2$	$\mathcal{S}_1$	v_{end}
τ_1	1	5	12	20
τ_2	1	8		20
τ_3	1	2/8/17	9	20
τ_4	1			20

Tabel 2.5: Tabel bewarna(2)

	v_{start}	$\mathcal{S}_1$	$\mathcal{S}_2$	v_{end}
τ_1	1	12	5	20
τ_2	1		8	20
τ_3	1	9	2/8/17	20
τ_4	1			20

2.4.3 Kutipan

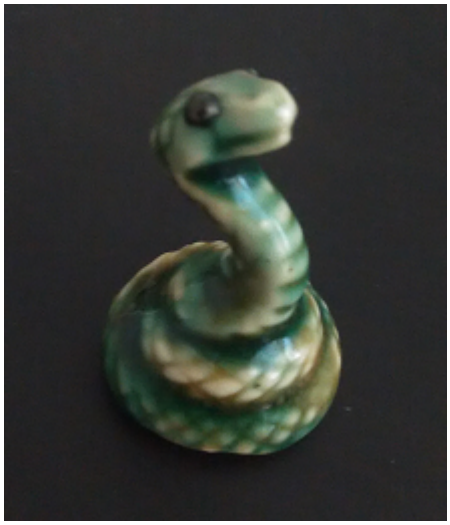
Berikut contoh kutipan dari berbagai sumber, untuk keterangan lebih lengkap, silahkan membaca file referensi.bib yang disediakan juga di template ini. Contoh kutipan:

- Buku: [5]
- Bab dalam buku: [6]
- Artikel dari Jurnal: [**?**]
- Artikel dari prosiding seminar/konferensi: [7]
- Skripsi/Thesis/Disertasi: [8] [9] [10]
- Technical/Scientific Report: [11]
- RFC (Request For Comments): [12]
- Technical Documentation/Technical Manual: [13] [14] [15]
- Paten: [16]
- Tidak dipublikasikan: [17] [18]
- Laman web: [19]
- Lain-lain: [20]

2.4.4 Gambar

Pada hampir semua editor, penempatan gambar di dalam dokumen LaTeX tidak dapat dilakukan melalui proses *drag and drop*. Perhatikan contoh pada file bab2.tex untuk melihat bagaimana cara menempatkan gambar. Beberapa hal yang harus diperhatikan pada saat menempatkan gambar:

- Setiap gambar **harus** diacu di dalam teks (gunakan *field* LABEL)
- *Field* CAPTION digunakan untuk teks pengantar pada gambar. Terdapat dua bagian yaitu yang ada di antara tanda [dan] dan yang ada di antara tanda { dan }. Yang pertama akan muncul di Daftar Gambar, sedangkan yang kedua akan muncul di teks pengantar gambar. Untuk skripsi ini, samakan isi keduanya.
- Jenis file yang dapat digunakan sebagai gambar cukup banyak, tetapi yang paling populer adalah tipe PNG (lihat Gambar 2.3), tipe JPG (Gambar 2.4) dan tipe PDF (Gambar 2.5)
- Besarnya gambar dapat diatur dengan *field* SCALE.



Gambar 2.3: Gambar *Serpentes* dalam format png

- Penempatan gambar diatur menggunakan *placement specifier* (di antara tanda [dan] setelah deklarasi gambar. Yang umum digunakan adalah **H** untuk menempatkan gambar **sesuai** penempatannya di file .tex atau **h** yang berarti "kira-kira" di sini.
  Jika tidak menggunakan *placement specifier*, LaTeX akan menempatkan gambar secara otomatis untuk menghindari bagian kosong pada dokumen anda. Walaupun cara ini sangat mudah, hindarkan terjadinya penempatan dua gambar secara berurutan.
  - Gambar 2.3 ditempatkan di bagian atas halaman, walaupun penempatannya dilakukan setelah penulisan 3 paragraf setelah penjelasan ini.
  - Gambar 2.4 dengan skala 0.5 ditempatkan di antara dua buah paragraf. Perhatikan penulisannya di dalam file bab2.tex!
  - Gambar 2.5 ditempatkan menggunakan *specifier* **h**.

Curabitur tellus magna, porttitor a, commodo a, commodo in, tortor. Donec interdum. Praesent scelerisque. Maecenas posuere sodales odio. Vivamus metus lacus, varius quis, imperdiet quis, rhoncus a, turpis. Etiam ligula arcu, elementum a, venenatis quis, sollicitudin sed, metus. Donec nunc pede, tincidunt in, venenatis vitae, faucibus vel, nibh. Pellentesque wisi. Nullam malesuada. Morbi ut tellus ut pede tincidunt porta. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam congue neque id dolor.

Donec et nisl at wisi luctus bibendum. Nam interdum tellus ac libero. Sed sem justo, laoreet vitae, fringilla at, adipiscing ut, nibh. Maecenas non sem quis tortor eleifend fermentum. Etiam id tortor ac mauris porta vulputate. Integer porta neque vitae massa. Maecenas tempus libero a libero posuere dictum. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aenean quis mauris sed elit commodo placerat. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Vivamus rhoncus tincidunt libero. Etiam elementum pretium justo. Vivamus est. Morbi a tellus eget pede tristique commodo. Nulla nisl. Vestibulum sed nisl eu sapien cursus rutrum.

Nulla non mauris vitae wisi posuere convallis. Sed eu nulla nec eros scelerisque pharetra. Nullam varius. Etiam dignissim elementum metus. Vestibulum faucibus, metus sit amet mattis rhoncus, sapien dui laoreet odio, nec ultricies nibh augue a enim. Fusce in ligula. Quisque at magna et nulla commodo consequat. Proin accumsan imperdiet sem. Nunc porta. Donec feugiat mi at justo. Phasellus facilisis ipsum quis ante. In ac elit eget ipsum pharetra faucibus. Maecenas viverra nulla in massa.

Nulla ac nisl. Nullam urna nulla, ullamcorper in, interdum sit amet, gravida ut, risus. Aenean ac enim. In luctus. Phasellus eu quam vitae turpis viverra pellentesque. Duis feugiat felis ut enim. Phasellus pharetra, sem id porttitor sodales, magna nunc aliquet nibh, nec blandit nisl mauris

at pede. Suspendisse risus risus, lobortis eget, semper at, imperdiet sit amet, quam. Quisque scelerisque dapibus nibh. Nam enim. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nunc ut metus. Ut metus justo, auctor at, ultrices eu, sagittis ut, purus. Aliquam aliquam.

Gambar 2.4: Ular kecil

Etiam pede massa, dapibus vitae, rhoncus in, placerat posuere, odio. Vestibulum luctus commodo lacus. Morbi lacus dui, tempor sed, euismod eget, condimentum at, tortor. Phasellus aliquet odio ac lacus tempor faucibus. Praesent sed sem. Praesent iaculis. Cras rhoncus tellus sed justo ullamcorper sagittis. Donec quis orci. Sed ut tortor quis tellus euismod tincidunt. Suspendisse congue nisl eu elit. Aliquam tortor diam, tempus id, tristique eget, sodales vel, nulla. Praesent tellus mi, condimentum sed, viverra at, consectetuer quis, lectus. In auctor vehicula orci. Sed pede sapien, euismod in, suscipit in, pharetra placerat, metus. Vivamus commodo dui non odio. Donec et felis.

Etiam suscipit aliquam arcu. Aliquam sit amet est ac purus bibendum congue. Sed in eros. Morbi non orci. Pellentesque mattis lacinia elit. Fusce molestie velit in ligula. Nullam et orci vitae nibh vulputate auctor. Aliquam eget purus. Nulla auctor wisi sed ipsum. Morbi porttitor tellus ac enim. Fusce ornare. Proin ipsum enim, tincidunt in, ornare venenatis, molestie a, augue. Donec vel pede in lacus sagittis porta. Sed hendrerit ipsum quis nisl. Suspendisse quis massa ac nibh pretium cursus. Sed sodales. Nam eu neque quis pede dignissim ornare. Maecenas eu purus ac urna tincidunt congue.

Gambar 2.5: *Serpentes* jantan

2.4.5 Kode Program

Kode program dalam bahasa tertentu seringkali harus ditulis di dalam bab, bukan hanya dilampirkan di bagian Lampiran. Kode 2.1 menampilkan penggunaan karakter-karakter yang umum digunakan dalam sebuah program yang ditulis dengan bahasa C.

Kode 2.1: Kode untuk menampilkan karakter-karakter aneh

```
// 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
```

2.4.6 Notasi

Simbol-simbol (matematika) yang sering digunakan sepanjang penulisan skripsi, dapat dimasukkan ke dalam "Daftar Notasi". Daftar ini ada di halaman depan sebelum Bab 1. Cara memasukkan sebuah simbol ke dalam Daftar Notasi adalah menggunakan perintah `\nomenclature`. Contoh:

```
\nomenclature[]{$A$}{luas kandang ular}
```

Argumen opsional digunakan untuk mengurutkan notasi. Silahkan lihat sendiri dokumentasi package `nomencl`

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] Mikkonen, T. dan Taivalsaari, A. (2007) Using javascript as a real programming language.

[4] Olanrewaju, R. F., Islam, T., dan Ali, N. (2015) An empirical study of the evolution of php mvc framework. *Advanced Computer and Communication Engineering Technology*, pp. 399–410. Springer, Malaysia.

[5] de Berg, M., Cheong, O., van Kreveld, M. J., dan Overmars, M. (2008) *Computational Geometry: Algorithms and Applications*, 3rd edition. Springer-Verlag, Berlin.

[6] van Kreveld, M. J. (2004) Geographic information systems. Bagian dari Goodman, J. E. dan O'Rourke, J. (ed.), *Handbook of Discrete and Computational Geometry*. Chapman & Hall/CRC, Boca Raton.

[7] van Kreveld, M. J. dan Wiratma, L. (2011) Median trajectories using well-visited regions and shortest paths. *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Chicago, USA, 1-4 November, pp. 241–250. ACM, New York.

[8] Lionov (2002) Animasi algoritma sweepline untuk membangun diagram voronoi. Skripsi. Universitas Katolik Parahyangan, Indonesia.

[9] Wiratma, L. (2010) Following the majority: a new algorithm for computing a median trajectory. Thesis. Utrecht University, The Netherlands.

[10] Wiratma, L. (2022) Coming Not Too Soon, Later, Delay, Someday, Hopefully. Disertasi. Utrecht University, The Netherlands.

[11] van kreveld, M., van Lankveld, T., dan Veltkamp, R. (2013) Watertight scenes from urban lidar and planar surfaces. Technical Report UU-CS-2013-007. Utrecht University, The Netherlands.

[12] Rekhter, Y. dan Li, T. (1994) A border gateway protocol 4 (bgp-4). RFC 1654. RFC Editor, http://www.rfc-editor.org.

[13] ITU-T Z.500 (1997) *Framework on formal methods in conformance testing*. International Telecommunications Union. Geneva, Switzerland.

[14] Version 9.0.0 (2016) *The Unicode Standard*. The Unicode Consortium. Mountain View, USA.

[15] Version 7.0 Nougat (2016) *Android API Reference Manual*. Google dan Open Handset Alliance. Mountain View, USA.

[16] Webb, R., Daruca, O., dan Alfadian, P. (2012) *Method of optimizing a text massage communication between a server and a secure element.* Paten no. EP2479956 (A1). European Patent Organisation. Munich, Germany.

[17] Wiratma, L. (2009) Median trajectory. Report for GMT Experimentation Project at Utrecht University.

[18] Lionov (2011) Polymorphism pada C++. Catatan kuliah AKS341 Pemrograman Sistem di Universitas Katolik Parahyangan, Bandung. http://tinyurl.com/lionov. 30 September 2016.

[19] Erickson, J. (2003) CG models of computation? http://www.computational-geometry.org/mailing-lists/compgeom-announce/2003-December/000852.html. 30 September 2016.

[20] AGUNG (2012) Menjajal tango 12. Majalah HAI no 02, Januari 2012.

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
```

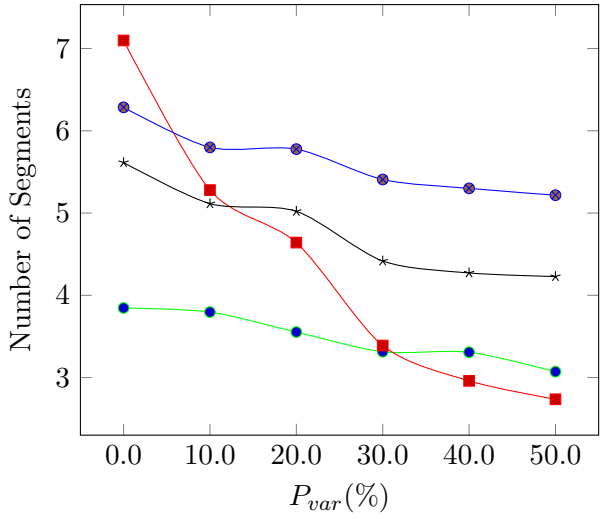
Kode A.2: MyCode.java

```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);
        }
    }

}
```

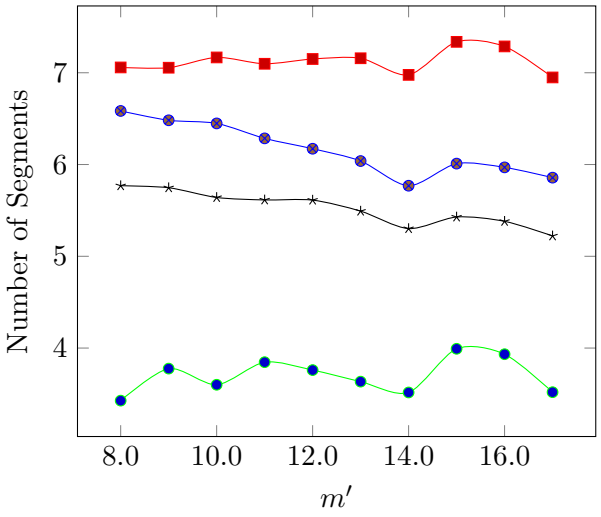
LAMPIRAN B

HASIL EKSPERIMEN

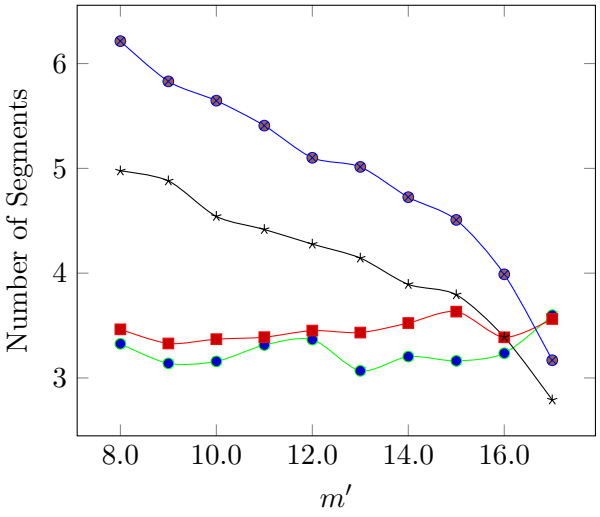
Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



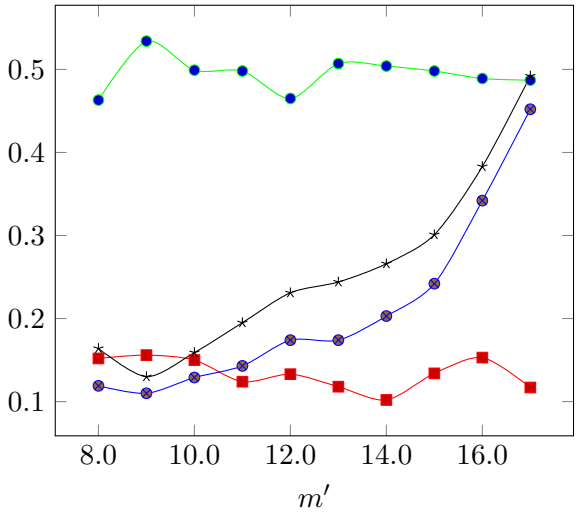
Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4