SKRIPSI

IMPLEMENTASI EDITOR KODE PADA SHARIF JUDGE

Nicholas Aditya Halim

NPM: 2017730018

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

«tahun»

UNDERGRADUATE THESIS

«JUDUL BAHASA INGGRIS»

Nicholas Aditya Halim

NPM: 2017730018

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY

«tahun»

LEMBAR PENGESAHAN

IMPLEMENTASI EDITOR KODE PADA SHARIF JUDGE

Nicholas Aditya Halim

NPM: 2017730018

Bandung, «tanggal» «bulan» «tahun»

Menyetujui,

Pembimbing Utama	Pembimbing Pendamping
Pascal Alfadian, Nugroho, M.Comp.	**«pembimbing pendamping/2»**

Ketua Tim Penguji	Anggota Tim Penguji
«penguji 1»	**«penguji 2»**

Mengetahui,

Ketua Program Studi

Mariskha Tri Adithia, P.D.Eng

PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

IMPLEMENTASI EDITOR KODE PADA SHARIF JUDGE

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,
Tanggal «tanggal» «bulan» «tahun»

Nicholas Aditya Halim
NPM: 2017730018

ABSTRAK

«Tuliskan abstrak anda di sini, dalam bahasa Indonesia»

Kata-kata kunci: «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Indonesia»

ABSTRACT

«Tuliskan abstrak anda di sini, dalam bahasa Inggris»

Keywords: «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Inggris»

«kepada siapa anda mempersembahkan skripsi ini. . . ?»

KATA PENGANTAR

«Tuliskan kata pengantar dari anda di sini . . . »

Bandung, «bulan» «tahun»

Penulis

DAFTAR ISI

DAFTAR GAMBAR

BAB 1

PENDAHULUAN

1.1 Latar Belakang

SharIF Judge (dengan IF kapital) adalah modifikasi dari aplikasi Sharif Judge buatan Mohammad Javad Naderi yang berfungsi untuk menilai kode program yang diunggah secara otomatis berdasarkan kunci jawaban yang disediakan. SharIF Judge digunakan pada beberapa kuliah di Informatika Unpar untuk mempermudah proses pengumpulan dan penilaian kode program, terutama saat ujian.

Dengan adanya situasi pandemi, seluruh kegiatan kuliah wajib dilaksanakan secara daring. Hal ini menyebabkan berbagai kesulitan, terutama dalam pelaksanaan ujian. Saat ujian sedang berlangsung, umumnya terdapat pengawas yang mengawasi mahasiswa secara fisik untuk mencegah kecurangan. Namun, pengawasan saat ujian menjadi sangat sulit untuk dilakukan saat kuliah dilaksanakan secara daring. Diperlukan sebuah cara untuk merekam tindakan-tindakan mahasiswa selama ujian daring berlangsung.

Maka, pada skripsi ini akan diimplementasikan editor kode pada SharIF Judge, yang sudah memiliki kemampuan untuk mengompilasi dan menjalankan kode. Dengan demikian, SharIF Judge dapat menjadi sebuah *Integrated Development Environment* yang mampu memfasilitasi seluruh proses pembuatan kode serta merekamnya.

Integrated Development Environment (IDE) adalah sebuah aplikasi yang menyediakan fasilitas untuk pembangunan perangkat lunak. Sebuah IDE memiliki kemampuan untuk mengedit, mengompilasi, dan menjalankan kode program. Pada umumnya, mahasiswa menggunakan aplikasi IDE seperti Netbeans untuk membuat kode program yang kemudian diunggah ke SharIF Judge untuk dinilai.

Dengan mengimplementasikan IDE berbasis web pada SharIF Judge, pengawasan terhadap mahasiswa saat ujian dapat dipermudah dengan merekam ketikan pada editor kode dan mendeteksi bila mahasiswa sedang melihat aplikasi selain SharIF judge.

1.2 Rumusan Masalah

Rumusan masalah yang akan dibahas pada skripsi ini adalah sebagai berikut:

- Bagaimana mengimplementasikan *Integrated Development Environment* sehingga mahasiswa dapat mengetik dan menjalankan kode dalam SharIF Judge?

1.3 Tujuan

Tujuan yang ingin dicapai skripsi ini adalah sebagai berikut:

- Mengimplementasikan *Integrated Development Environment* sehingga mahasiswa dapat mengetik dan menjalankan kode dalam SharIF Judge.

1.4 Batasan Masalah

Perangkat lunak diuji pada kuliah Dasar-dasar Pemrograman semester 51 Informatika Unpar. Pada kuliah ini terdapat 2 alamat *judge* yang digunakan, yaitu `http://daspro.labftis.net` untuk latihan, dan `http://daspro-quiz.labftis.net` untuk kuis. Perangkat lunak skripsi ini hanya akan diuji pada *judge* latihan.

1.5 Metodologi

Metodologi pengerjaan skripsi ini adalah sebagai berikut:

1. Melakukan studi mengenai komponen yang diperlukan untuk membuat IDE berbasis web.
2. Mempelajari struktur SharIF Judge.
3. Merancang IDE berbasis web untuk SharIF Judge.
4. Mengimplementasikan IDE pada SharIF Judge.
5. Melakukan pengujian dan eksperimen.
6. Menulis dokumen skripsi.

1.6 Sistematika Pembahasan

Sistematika pembahasan skripsi ini adalah sebagai berikut:

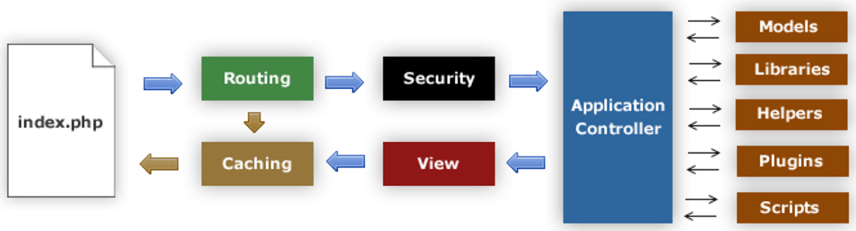
- Bab 1 Pendahuluan membahas mengenai latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi, dan sistematika pembahasan.
- Bab 2 Landasan Teori

BAB 2

LANDASAN TEORI

2.1 CodeIgniter

CodeIgniter adalah sebuah *framework* untuk membangun situs web menggunakan PHP. Tujuan utamanya adalah untuk mempercepat pembuatan proyek dengan menyediakan *library* yang lengkap untuk fungsi-fungsi yang umum digunakan, serta antarmuka yang sederhana dan struktur yang logis untuk mengakses *library* tersebut [1]. Versi CodeIgniter yang digunakan pada SharIF Judge adalah CodeIgniter 3.



Gambar 2.1: *Flow Chart* CodeIgniter

Gambar 2.1 mengilustrasikan bagaimana data mengalir pada sistem CodeIgniter.

1. *File* index.php berfungsi sebagai *front controller*, menginisialisasi *resource* utama untuk menjalankan CodeIgniter.
2. Router meneliti *request* HTTP dan menentukan apa yang harus dilakukan.
3. Jika terdapat *file cache*, maka langsung dikirimkan ke *browser*.
4. Sebelum *controller* dimuat, seluruh *request* HTTP dan data dari user disaring terlebih dahulu untuk keamanan.
5. *Controller* memuat *model*, *library* utama, dan *resource* lainnya yang diperlukan.
6. *View* akhir lalu dikirim ke browser untuk dilihat. *Cache* akan dibuat terlebih dahulu bila diaktifkan.

2.1.1 Model-View-Controller

CodeIgniter menggunakan pola arsitektur MVC (*Model-View-Controller*) sebagai dasarnya. MVC memisahkan proses logika aplikasi dari presentasi. Dengan demikian, halaman web dapat memuat sedikit *script* karena presentasinya terpisah dari *scripting* PHP.

Model

Model merepresentasikan struktur data. Biasanya *model* memiliki fungsi-fungsi yang membantu dalam mengambil, memasukkan, dan memperbarui informasi pada *database*. Pada CodeIgniter, *model* adalah sebuah kelas yang mengekstensi `CI_Model` dan terletak di direktori `application/models/`.

Kode 2.1: Contoh *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']));
        }

}
```

Kode 2.1 merupakan contoh sebuah kelas *model* pada CodeIgniter. Kelas tersebut mengekstensi `CI_Model` dan memiliki fungsi untuk mengambil, memasukkan, dan memperbarui *database*.

View

View adalah informasi yang ditampilkan kepada pengguna. Pada CodeIgniter, *view* merupakan sebuah halaman web atau sebagian dari halaman web yang terletak di direktori `application/view/`.

Kode 2.2: Contoh *view*

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

Kode 2.2 merupakan contoh sebuah *view*. *View* pada CodeIgniter harus dipanggil melalui *Controller* dan tidak pernah dipanggil secara langsung.

Controller

Controller adalah perantara dari *model* dan *view*, serta *resource* lainnya yang diperlukan untuk memproses *request* HTTP dan menghasilkan sebuah halaman web. Pada CodeIgniter, *controller* adalah sebuah kelas yang mengekstensi `CI_Controller` dan terletak di direktori `application/controllers/`.

Kode 2.3: Contoh *controller*

```
<?php
class Blog extends CI_Controller {

        public function index()
        {
                echo 'Hello World!';
        }

        public function comments()
        {
                echo 'Look at this!';
        }
}
```

Kode 2.1 merupakan contoh sebuah kelas *controller* pada CodeIgniter. Kelas tersebut mengekstensi `CI_Controller` dan memiliki fungsi `index()` dan `comments()`. Fungsi `index()` akan dipanggil secara otomatis jika tidak ada fungsi lain yang dipanggil.

Kode 2.4: Contoh memuat *model* dan menampilkan *view*

```php
<?php
class Blog_controller extends CI_Controller {

        public function blog()
        {
                $this->load->model('blog');

                $data['query'] = $this->blog->get_last_ten_entries();

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

Pada CodeIgniter, *model* dan *view* hanya dapat dimuat melalui controller. Pada contoh kode 2.4, fungsi `blog()` pada *controller* memuat *model* untuk mengambil data dari *database*, lalu menampilkan *view* yang memuat data tersebut.

2.1.2 URL CodeIgniter

URL pada CodeIgniter menggunakan *segment-based approach* yang dirancang untuk lebih mudah dibaca oleh *search engine* dan manusia. Berikut ini adalah contoh sebuah URL pada CodeIgniter:

`example.com/class/function/ID`

- Bagian pertama, `class` merepresentasikan kelas *controller* yang akan dipanggil.
- Bagian kedua, `function` merepresentasikan fungsi yang akan dipanggil.
- Bagian ketiga dan seterusnya, `ID` merepresentasikan variabel yang akan digunakan.

2.2 Twig

Twig adalah sebuah *template engine* untuk PHP. Sebuah *template* Twig memuat *variable* atau *expression* yang nantinya akan diubah menjadi *value* saat template dievaluasi, serta *tag* yang mengontrol logika template [2].

Kode 2.5: Contoh template Twig

```twig
<!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>
```

Kode 2.5 merupakan contoh sebuah template Twig. Terdapat dua jenis *delimiter*, yaitu `{% ... %}` dan `{{ ... }}`. *Delimiter* `{% ... %}` digunakan untuk menjalankan *statement* seperti `for` dan `if`, sementara *delimiter* `{{ ... }}` digunakan untuk menampilkan nilai dari *variable* atau *expression*.

2.3 PDF.js

PDF.js adalah sebuah library JavaScript yang berfungsi untuk menampilkan *file* Portable Document Format (PDF) menggunakan HTML5 *Canvas* [3]. PDF.js terdiri dari 3 *layer*:

- ***Core*** merupakan bagian dimana proses *parse* dan *interpret* dilakukan terhadap *binary* PDF.
- ***Display*** mengambil *layer core* sebagai API yang lebih mudah digunakan untuk menampilkan PDF dan mengambil informasi lainnya dari sebuah dokumen.
- ***Viewer*** membangun *layer display* sebagai halaman website dengan *user interface* yang dapat ditampilkan di browser.

Kode 2.6: Contoh kode untuk menggunakan PDF.js

```
<!DOCTYPE html>
<html>
    <iframe src="/web/viewer.html?file=sample.pdf"></iframe>
</html>
```

Salah satu cara untuk menampilkan *file* PDF menggunakan PDF.js adalah dengan *embed* layer *viewer* yang sudah tersedia melalui `web/viewer.js` pada sebuah `iframe`. Kode 2.6 merupakan contoh kode *embed* PDF.js untuk menampilkan sebuah file PDF contoh `sample.pdf`.

2.4 Ace

Ace adalah sebuah library JavaScript yang berfungsi sebagai *code editor*. Ace memiliki fitur-fitur yang dapat ditemukan di *code editor* pada umumnya [4]. Berikut ini merupakan beberapa fitur utama dari Ace:

- *Syntax highlighting* untuk lebih dari 110 bahasa pemrograman.
- *Indent* dan *outdent* otomatis.
- Kemampuan *cut*, *copy*, dan *paste*.
- *Drag and drop* teks menggunakan mouse.

Berikut ini adalah beberapa kelas yang terdapat pada Ace:

- `Ace`
  Kelas utama yang digunakan mempersiapkan Ace pada browser. Salah satu fungsi yang dimiliki:
  - `edit(String` DOMElement el)|
    Embed Ace pada elemen yang disediakan.
- `Anchor`
  Menangani posisi *pointer* pada dokumen.
- `BackgroundTokenizer`
  Bekerja di latar belakang untuk melakukan tokenisasi pada dokumen saat ini dan menyimpan baris yang sudah ditokenisasi sebagai *cache*.
- `Document`
  Menyimpan teks dari dokumen.
- `EditSession`
  Menyimpan seluruh *state* untuk `Editor` dan menyediakan cara untuk mengubahnya dengan mudah. Beberapa fungsi yang dimiliki:
  - `getMode()`
    Mengembalikan mode *syntax highlighting* editor yang sedang digunakan.
  - `setMode()`
    Mengubah mode *syntax highlighting* editor.
- `Editor`
  Entry point utama untuk seluruh kegunaan Ace. Beberapa fungsi yang dimiliki:
  - `getReadOnly()`
    Mengembalikan `true` jika editor sedang menggunakan pengaturan *read-only*.
  - `getTheme()`
    Mengembalikan alamat tema editor yang sedang digunakan.
  - `getValue()`
    Mengembalikan isi teks editor.
  - `setReadOnly(Boolean readOnly)`
    Mengubah pengaturan *read-only*.
  - `setTheme(String style)`
    Mengubah tema editor.
  - `setValue(String val, Number cursorPos)`
    Mengubah isi teks editor.

- `Range`
  Mengindikasi sebuah daerah pada editor.
- `Scrollbar`
  Menangani *scrollbar* editor.
- `Search`
  Menangani seluruh operasi pencarian teks pada dokumen.
- `Selection`
  Menyimpan posisi kursor dan seleksi teks pada editor.
- `TokenIterator`
  Menyediakan fungsi untuk membaca dokumen sebagai aliran token.
- `Tokenizer`
  Menerima sejumlah aturan dan membuat `Tokenizer`.
- `UndoManager`
  Menangani fungsi *undo* pada editor.
- `VirtualRenderer`
  Menggambar tampilan yang terlihat di layar.

Kode 2.7: Contoh kode untuk menggunakan Ace

```
<!DOCTYPE html>
<html>
<head>
<title>ACE in Action</title>
</head>
<body>

<div id="editor">
function foo(items) {
    var x = "All␣this␣is␣syntax␣highlighted";
    return x;
}
</div>

<script src="/ace-builds/src-noconflict/ace.js" type="text/javascript" charset="utf-8"></script>
<script>
    var editor = ace.edit("editor");
    editor.setTheme("ace/theme/monokai");
    editor.session.setMode("ace/mode/javascript");
</script>
</body>
</html>
```

Kode 2.7 merupakan contoh kode untuk menempatkan editor Ace pada sebuah elemen `div` dengan id `editor`. Terdapat berbagai konfigurasi pada Ace, pada contoh ini digunakan tema *monokai* dan mode *syntax highlighting* untuk JavaScript.

BAB 3

ANALISIS

3.1 Analisis Sistem Kini

SharIF Judge menggunakan *framework* CodeIgniter. Seperti yang dibahas pada bagian 2.1.1, *framework* CodeIgniter menerapkan pola arsitektur MVC, dengan komponen-komponen *model*, *view*, dan *controller* yang terdapat pada `\application`.

3.1.1 Model

Berikut ini adalah seluruh *model* pada SharIF Judge yang terletak pada `\application\models`:

- `Assignment_model`
  Model untuk menangani *database* `assignments`. Fungsi yang dimiliki:
  - `add_assignment($id, $edit = FALSE)`
    Menambah atau memperbarui sebuah *assignment*.
  - `delete_assignment($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, $username)`
    Mengembalikan `TRUE` jika `$username` terdapat dalam `$participants`.
  - `increase_total_submits($assignment_id)`
    Meningkatkan 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, $type)`
    Menambah atau memperbarui deskripsi sebuah *problem*.
  - `_update_coefficients($assignment_id, $extra_time, $finish_time, $new_late_rule)`
    Memperbarui koefisien seluruh *submission* pada sebuah *assignment*. Fungsi ini dipanggil oleh `add_assignment($id, TRUE)`.
- `Hof_model`
  Model untuk menangani informasi *hall of fame*. Fungsi yang dimiliki:

  - `get_all_final_submission()`
    Mengambil seluruh *final submission*.
  - `get_all_user_assignments($username)`
    Mengambil seluruh *assignment* dan *problem* untuk *user* tertentu.
- `Logs_model`
  Model untuk menangani *database* `logins`. Fungsi yang dimiliki:
  - `insert_to_logs($username, $ip_adrress)`
    Menambah sebuah catatan *login* dan menghapus catatan yang sudah melebihi 24 jam.
  - `get_all_logs()`
    Mengambil seluruh catatan *login*.
- `Notifications_model`
  Model untuk menangani *database* `notifications`. Fungsi yang dimiliki:
  - `get_all_notifications()`
    Mengambil seluruh notifikasi.
  - `get_latest_notifications()`
    Mengambil 10 notifikasi terbaru.
  - `add_notification($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 `$time`.
- `Queue_model`
  Model untuk menangani *database* `queue`. Fungsi yang dimiliki:
  - `in_queue($username, $assignment, $problem)`
    Mengembalikan `TRUE` jika sebuah *submission* sudah berada dalam antrian.
  - `get_queue()`
    Mengambil seluruh antrian.
  - `empty_queue()`
    Mengosongkan antrian.
  - `add_to_queue($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 *entry* pertama dari antrian.
  - `remove_item($username, $assignment, $problem, $submit_id)`
    Menghapus sebuah *entry* dari antrian.
  - `save_judge_result_in_db ($submission, $type)`
    Menyimpan hasil penilaian ke dalam *database*. Fungsi ini dipanggil oleh *controller* `Queueprocess`.
- `Scoreboard_model`
  Model untuk menangani *database* `scoreboard`. Fungsi yang dimiliki:
  - `_generate_scoreboard($assignment_id)`

Membuat *scoreboard* untuk sebuah *assignment*. Fungsi ini dipanggil oleh `update_scoreboard($assig`
  - `update_scoreboards()`
    Memperbarui *scoreboard* untuk seluruh *assignment*.
  - `update_scoreboard($assignment_id)`
    Memperbarui *scoreboard* untuk sebuah *assignment*.
  - `get_scoreboard($assignment_id)`
    Mengambil *scoreboard* untuk sebuah *assignment*.
- `Settings_model`
  Model untuk menangani *database* `settings`. Fungsi yang dimiliki:
  - `get_setting($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 untuk menangani *database* `submissions`. Fungsi yang dimiliki:
  - `get_submission($username, $assignment, $problem, $submit_id)`
    Mengambil sebuah *submission*.
  - `get_final_submissions($assignment_id, $user_level, $username, $page_number = NULL,`
    Mengambil seluruh *final submission* untuk sebuah *assignment*.
  - `get_all_submissions($assignment_id, $user_level, $username, $page_number = NULL, $f`
    Mengambil seluruh *submission* untuk sebuah *assignment*.
  - `count_final_submissions($assignment_id, $user_level, $username, $filter_user = NULL`
    Menghitung jumlah *final submission* dari *user* tertentu.
  - `count_all_submissions($assignment_id, $user_level, $username, $filter_user = NULL,`
    Menghitung jumlah *submission* dari *user* tertentu.
  - `set_final_submission($username, $assignment, $problem, $submit_id)`
    Memperbarui sebuah *submission* menjadi *final*.
  - `add_upload_only($submit_info)`
    Menambahkan hasil dari *submission upload only* ke dalam *database*.
- `User`
  Model 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 untuk menangani *database* `users`. Fungsi yang dimiliki:
  - `have_user($username)`
    Mengembalikan `TRUE` jika terdapat *user* dengan nama `$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, $username = FALSE)`
    Mengembalikan `TRUE` jika terdapat *user* selain `$username` dengan email `$email`.

  - `add_user($username, $email, $display_name, $password, $role)`
    Menambahkan sebuah *user* baru.
  - `add_users($text, $send_mail, $delay)`
    Menambahkan beberapa *user* baru.
  - `delete_user($user_id)`
    Menghapus sebuah *user*.
  - `delete_submissions($user_id)`
    Menghapus seluruh *submission* dari sebuah *user*.
  - `validate_user($username, $password)`
    Mengembalikan `TRUE` jika `$username` dan `$password` valid untuk login.
  - `selected_assignment($username)`
    Mengembalikan *assignment* yang dipilih sebuah *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($passchange_key)`
    Mengembalikan `TRUE` jika kunci untuk *reset password* valid.
  - `reset_password($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*.

3.1.2 View

View pada SharIF Judge yang terletak pada `\application\views` terbagi menjadi beberapa kategori:

- `errors`
  Menyimpan tampilan halaman error.
- `pages`
  Menyimpan tampilan utama halaman.
- `templates`
  Menyimpan komponen-komponen dasar halaman.

3.1.3 Controller

Berikut ini adalah seluruh *controller* pada SharIF Judge yang terletak pada `\application\controllers`:

- `Assignments`
  Controller untuk menangani *assignments*. Fungsi yang dimiliki:
  - `select()`
    Memilih *assignment* yang sedang ditampilkan.
  - `pdf($assignment_id, $problem_id = NULL)`
    Mengunduh *file* PDF dari sebuah *assignment*.
  - `downloadtestsdesc($assignment_id = FALSE)`
    Mengunduh *file test case* dari sebuah *assignment*.
  - `download_submissions($type = FALSE, $assignment_id = FALSE)`
    Mengunduh seluruh *file final submission* dari sebuah *assignment*.

  - `delete($assignment_id = FALSE)`
    Menghapus sebuah *assignment*.
  - `add()`
    Menambah atau memperbarui *assignment*.
  - `edit($assignment_id)`
    Memperbarui *assignment*.
- `Dashboard`
  Controller untuk menangani halaman *Dashboard*. Fungsi yang dimiliki:
  - `widget_positions()`
    Menyimpan posisi *widget* dari *user*.
- `Halloffame`
  Controller untuk menangani halaman *Hall of Fame* . Fungsi yang dimiliki:
  - `hof_details()`
    Mengambil data yang diperlukan untuk *hall of fame*.
- `Install`
  Controller untuk menangani instalasi SharIF Judge.
- `Login`
  Controller 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 meminta *reset password*.
  - `reset($passchange_key = FALSE)`
    Memproses dan menampilkan halaman untuk ubah *reset password*.
- `Logs`
  Controller untuk menangani halaman *24-hour Log*.
  - `index()` Mengambil data yang diperlukan dan menampilkan halaman *24-hour Log*.
- `Moss`
  Controller untuk menangani halaman *Detect Similar Codes* . Fungsi yang dimiliki:
  - `update($assignment_id = FALSE)`
    Memperbarui informasi pada halaman *Detect Similar Codes*.
  - `_detect($assignment_id = FALSE)`
    Menjalankan Moss untuk mendeteksi kesamaan kode.
- `Notifications`
  Controller untuk menangani halaman *Notifications*. Fungsi yang dimiliki:
  - `add()`
    Menambahkan notifikasi baru dan menampilkan halaman *New Notification*.
  - `edit($notif_id = FALSE)`
    Memperbarui sebuah notifikasi.
  - `delete()`
    Menghapus sebuah notifikasi.
  - `check()`
    Memeriksa adanya notifikasi baru.
- `Problems`
  Controller untuk menangani halaman *Problems*. Fungsi yang dimiliki:
  - `index($assignment_id = NULL, $problem_id = 1)`
    Mengambil data yang diperlukan dan menampilkan halaman *Problems*.
  - `edit($type = 'md', $assignment_id = NULL, $problem_id = 1)`
    Memperbarui deskripsi *problem* dan menampilkan halaman *Edit Problem Description*.

- `Profile`
  Controller untuk menangani halaman *Profile*. Fungsi yang dimiliki:
  - `index($user_id = FALSE)`
    Mengambil data yang diperlukan dan menampilkan halaman *Profile*.
  - `_password_check($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($role)`
    Memeriksa *role* yang dimiliki *user*.
- `Queue`
  Controller untuk menangani halaman *Queue*. Fungsi yang dimiliki:
  - `index()`
    Mengambil data yang diperlukan dan menampilkan halaman *Queue*.
  - `pause()`
    Memberhentikan antrian.
  - `resume()`
    Melanjutkan antrian.
  - `empty_queue()`
    Mengosongkan antrian.
- `Queueprocess`
  Controller untuk menangani proses penilaian kode. Fungsi yang dimiliki:
  - `run()`
    Menilai kode satu per satu dari antrian.
- `Rejudge`
  Controller untuk menangani halaman *Rejudge*. Fungsi yang dimiliki:
  - `index()`
    Mengambil data yang diperlukan dan menampilkan halaman *Rejudge*.
  - `rejudge_single()`
    Melakukan penilaian ulang untuk sebuah *submission*.
- `Server_time`
  Controller untuk menangani sinkronisasi waktu server. Fungsi yang dimiliki:
  - `index()`
    Mengembalikan waktu server.
- `Submissions`
  Controller untuk menangani unduh *submissions* menjadi file Excel. Fungsi yang dimiliki:
  - `_download_excel($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 untuk menangani *submissions*. Fungsi yang dimiliki:
  - `_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 untuk menangani halaman *Users*. Fungsi yang dimiliki:
  - `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*.

DAFTAR REFERENSI

[1] Version 3.1.11 (2019) *CodeIgniter User Guide*. British Columbia Institute of Technology. Burnaby, Canada.

[2] Version 1.44.5 (2021) *Twig Documentation*. Symfony SAS. Clichy, France.

[3] Version 2.7.570 (2021) *PDF.js*. Mozilla Corporation. Mountain View, United States.

[4] Version 1.4.13 (2021) *Ace API Reference*. Ajax.org B.V. Amsterdam, The Netherlands.

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
```

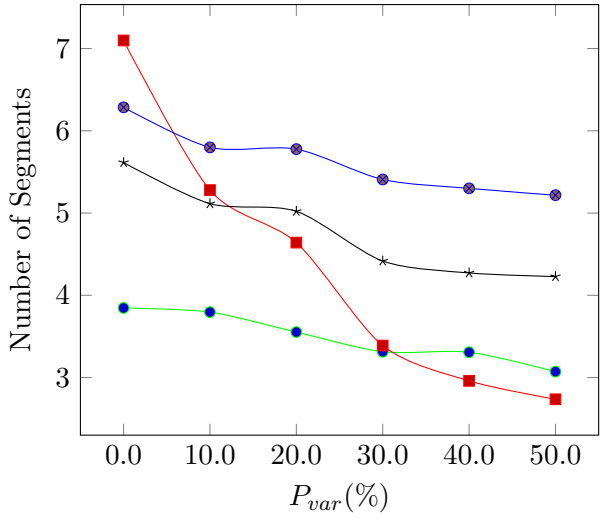
Kode A.2: MyCode.java

```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);
        }
    }

}
```

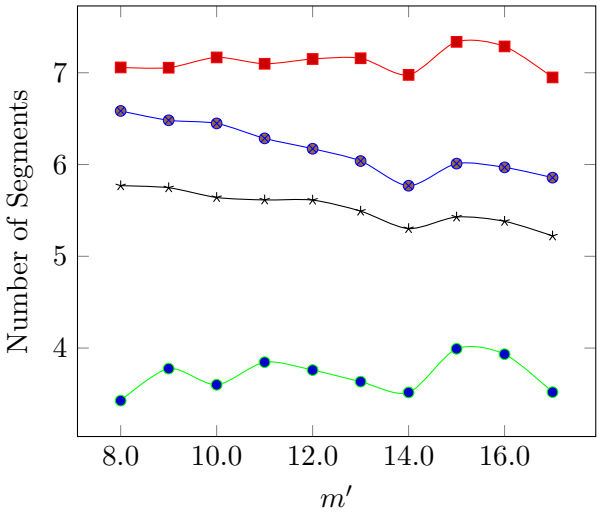
LAMPIRAN B

HASIL EKSPERIMEN

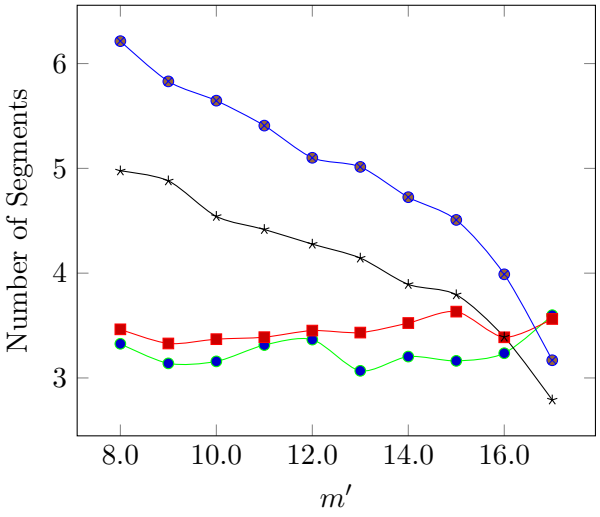
Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



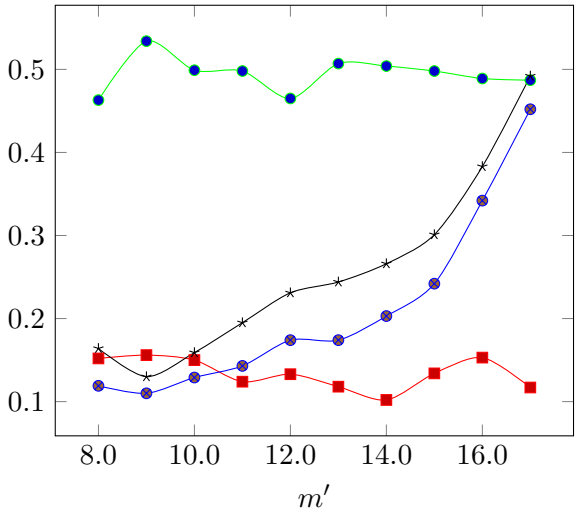
Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4