

SKRIPSI

PEMUTARAN ULANG KETIKAN MAHASISWA PADA SHARIF JUDGE



Andreas Ronaldi

NPM: 6182101026

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2025

UNDERGRADUATE THESIS

TYPING REPLAY ON SHARIF JUDGE



Andreas Ronaldi

NPM: 6182101026

**DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2025**

LEMBAR PENGESAHAN

PEMUTARAN ULANG KETIKAN MAHASISWA PADA SHARIF JUDGE

Andreas Ronaldi

NPM: 6182101026

Bandung, «**tanggal**» «**bulan**» 2025

Menyetujui,

Pembimbing

Pascal Alfadian, Nugroho, M.Comp.

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:

PEMUTARAN ULANG KETIKAN MAHASISWA 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 dijatuhan 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**» 2025



Andreas Ronaldi
NPM: 6182101026

ABSTRAK

Pada Universitas Katolik Parahyangan, digunakan SharIF Judge sebagai Online Judge yang telah dimodifikasi untuk mendukung kebutuhan pembelajaran. Namun, sistem ini masih memiliki keterbatasan dalam pengawasan, terutama saat ujian online, di mana potensi kecurangan seperti *copy-paste* meningkat. Untuk mempermudah pengawasan, tugas akhir sebelumnya telah mengintegrasikan Integrated Development Environment atau IDE ke dalam SharIF Judge, memungkinkan seluruh proses pengerjaan tugas dilakukan dalam satu platform.

Meskipun demikian, IDE tersebut tidak dapat mengawasi aktivitas pengguna jika terjadinya kecurangan. Oleh karena itu, dalam tugas akhir ini, dikembangkan fitur perekaman ketikan dan pemutaran ulang ketikan pada IDE SharIF Judge. Fitur ini bertujuan memudahkan pengawasan, mendeteksi kecurangan, dan memberikan bukti jika terjadi pelanggaran.

Kata-kata kunci: *Online Judge, Integrated Development Environment, Perekaman Penekanan Tombol*

ABSTRACT

At Parahyangan Catholic University, SharIF Judge is used as an Online Judge that has been modified to support learning needs. However, this system still has limitations in monitoring, especially during online exams, where the potential for cheating, such as copy-pasting, increases. To facilitate supervision, the previous final project integrated an Integrated Development Environment or IDE into SharIF Judge, allowing the entire task completion process to be conducted within a single platform.

Nevertheless, the IDE could not monitor user activity in cases of cheating. Therefore, in this final project, a keystroke recording and playback feature was developed for the SharIF Judge IDE. This feature aims to simplify monitoring, detect cheating, and provide evidence in case of violations.

Keywords: Online Judge, Integrated Development Environment, Keystroke Logging

*Tugas akhir ini dipersembahkan kepada Tuhan Yang Maha Esa,
Keluarga, dan rekan-rekan sesama mahasiswa*

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas berkat dan penyertaan-Nya, penulis dapat menyelesaikan tugas akhir ini dengan judul "Pemutaran Ulang Ketikan Mahasiswa Pada SharIF Judge". Penulis juga ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada berbagai pihak yang telah memberikan dukungan, bimbingan, dan bantuan selama proses pengerjaan. Secara khusus, penulis berterima kasih kepada:

1. Kedua orang tua dan keluarga tercinta yang senantiasa memberikan doa, dukungan, dan motivasi tanpa henti.
2. Bapak Pascal Alfadian, Nugroho, M.Comp., selaku dosen pembimbing yang telah meluangkan waktu, memberikan bimbingan, saran, dan arahan berharga selama penyusunan skripsi ini.
3. Kepada seluruh teman-teman yang telah menemani, membantu, dan memberi dukungan selama pengerjaan tugas akhir ini.
4. Semua pihak yang tidak dapat disebutkan satu per satu, yang telah berkontribusi baik secara langsung maupun tidak langsung.

Akhir kata, penulis ingin mengucapkan permohonan maaf atas kekurangan-kekurangan yang terdapat pada tugas akhir ini. Penulis berharap agar tugas akhir ini dapat bermanfaat bagi pembaca yang hendak melakukan penelitian serupa.

Bandung, «bulan» 2025

Penulis

DAFTAR ISI

KATA PENGANTAR	xv
DAFTAR ISI	xvii
DAFTAR GAMBAR	xix
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Tujuan	3
1.4 Batasan Masalah	3
1.5 Metodologi	4
1.6 Sistematika Pembahasan	4
2 LANDASAN TEORI	5
2.1 SharIF Judge	5
2.1.1 Instalasi	5
2.1.2 Users	6
2.2 CodeIgniter 3	6
2.2.1 Model-View-Controller	7
2.2.2 CodeIgniter URLs	9
2.2.3 <i>Helpers</i>	9
2.3 Twig	9
2.4 Integrated Development Environment	10
2.5 Ace	10
2.5.1 Perekaman Event	12
2.6 Chart.js	13
3 ANALISIS	17
3.1 Analisis Sistem Kini	17
3.1.1 Model, View, Controller	17
3.1.2 Assets	39
3.1.3 Penyimpanan Kode Submission	41
3.1.4 Antrean Penilaian Kode	41
3.2 Analisis Sistem Usulan	41
3.2.1 Fitur perekaman perubahan atau event	42
3.2.2 Fitur penyimpanan rekaman perubahan	43
3.2.3 Fitur melihat daftar rekaman	43
3.2.4 Fitur pemutaran ulang rekaman	44
4 PERANCANGAN	47
4.1 Rancangan Antarmuka	47
4.1.1 Sistem Rekaman	47

4.1.2	Sistem Pemutaran ulang	47
4.2	Rancangan Penyimpanan Rekaman	48
4.3	Rancangan Perubahan Kode	49
4.3.1	Merekam perubahan atau event	49
4.3.2	Menyimpan rekaman	50
4.3.3	Melihat daftar rekaman	50
4.3.4	Pemutaran ulang rekaman	51
5	IMPLEMENTASI DAN PENGUJIAN	53
5.1	Implementasi	53
5.1.1	Merekam Peristiwa pada IDE	53
5.1.2	Menyimpan Rekaman pada Sistem	55
5.1.3	Melihat Daftar Rekaman	57
5.1.4	Pemutaran Ulang Rekaman	57
5.2	Pengujian Fungsional	59
5.3	Pengujian Eksperimental	59
5.3.1	Lingkungan pengujian	59
5.3.2	Eksperimen	60
6	KESIMPULAN DAN SARAN	65
6.1	Kesimpulan	65
6.2	Saran	65
DAFTAR REFERENSI		67
A	KODE PROGRAM	69
B	HASIL HALAMAN	97
C	FILE DOCKER EKSPERIMEN	99

DAFTAR GAMBAR

1.1	Sistem Tradisional Pemberian Tugas	1
1.2	Sistem Integrasi oleh <i>Online Judge</i>	2
1.3	Tampilan Awal SharIF Judge	2
2.1	<i>Flow Chart</i> CodeIgniter	6
2.2	Hasil Web Page <i>Library Ace</i>	12
2.3	Hasil Web Page <i>Library Chart.js</i>	15
3.1	Struktur MVC pada SharIF Judge	18
3.2	Struktur Kelas Model pada SharIF Judge	19
3.3	Struktur Direktori View pada SharIF Judge	23
3.4	Struktur Kelas Controller pada SharIF Judge	25
3.5	Halaman Assignments	26
3.6	Halaman Dashboard	27
3.7	Halaman Hall of Fame	27
3.8	Halaman Install	28
3.9	Halaman Login	29
3.10	Halaman 24-Hour Log	29
3.11	Halaman Moss	30
3.12	Halaman Notifications	31
3.13	Halaman Problems	32
3.14	Halaman Profile	33
3.15	Halaman Queue	34
3.16	Halaman Rejudge	35
3.17	Halaman Scoreboard	35
3.18	Halaman Settings	36
3.19	Halaman Final Submissions	37
3.20	Halaman All Submissions	37
3.21	Halaman Submit	38
3.22	Halaman Users	39
3.23	Usecase analisis sistem usulan	42
3.24	Sequence Diagram Fitur Perekaman Perubahan	42
3.25	Sequence Diagram Fitur Penyimpanan Rekaman	43
3.26	Sequence Diagram Membuka Halaman Rekaman	44
3.27	Sequence Diagram Membuka Halaman Rekaman	45
4.1	Halaman	47
4.2	Rancangan Antarmuka Halaman Daftar Rekaman	48
4.3	Rancangan Antarmuka Halaman Pemutaran Ulang	48
5.1	Bagan Histogram Perubahan Kode Program	61
5.2	Bagan Heatmap Perubahan Lokasi Kode Program	62
5.3	Bagan Histogram Perubahan Input dan Aksi <i>Execute</i>	62

5.4	Bagan Histogram Perubahan Navigasi	63
5.5	Bagan Histogram Perubahan	63
5.6	Bagan Histogram Perubahan	64
B.1	Halaman Daftar Rekaman	97
B.2	Halaman Rekaman	98

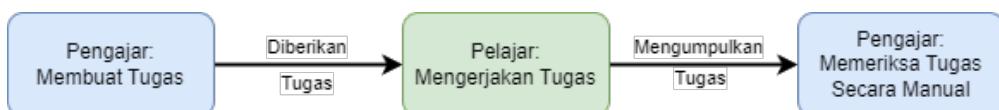
BAB 1

PENDAHULUAN

1.1 Latar Belakang

Institusi yang memberikan pendidikan, perlu memiliki cara untuk mengetahui pemahaman pelajarannya. Salah satu caranya adalah dengan memberikan tugas. Tugas merupakan sebuah bentuk penilaian dari pengajar kepada pelajarnya [1]. Tugas diberikan kepada pelajar untuk membantu pelajar mendalami materi yang sudah diberikan sebelumnya oleh pengajar dan juga untuk melihat seberapa jauh pemahaman pelajar terhadap materi yang sudah diberikan.

Pada bidang informatika, banyak materi pembelajaran yang dapat diberikan. Salah satu pembelajaran utama dalam bidang informatika adalah keterampilan pemrograman. Dikarenakan itu, perlu sebuah sistem untuk melatih keterampilan pemrograman yaitu dengan memberikan tugas menulis kode program sesuai dengan petunjuk yang diberikan dan program tersebut dapat berjalan sesuai dengan petunjuk [2]. Secara tradisional, tugas ini diberikan dengan cara pengajar menyiapkan dan mendistribusikan tugas tersebut kepada pelajar, kemudian dikumpulkan kembali hasil program pekerjaan pelajar, dan pengajar akan menilai kode program sesuai ketepatan dengan program yang diinginkan secara manual seperti Gambar 1.1. Karena menilaian kode program mencakup keluaran program dan juga analisis kode, maka proses tersebut memakan waktu yang cukup lama untuk dilakukan. Walaupun begitu, cara tradisional ini masih bekerja jika jumlah pelajarnya sedikit. Tetapi semakin banyak kode program yang harus di periksa maka semakin banyak waktu yang dibutuhkan dan semakin banyak pula kesalahan yang berhubungan dengan manusia. Salah satu masalah lain yang muncul juga adalah pelajar tidak dapat mengetahui apakah kode program berada pada jalur yang benar dalam menemukan solusi tugas tersebut.



Gambar 1.1: Sistem Tradisional Pemberian Tugas

Pemberian tugas menulis kode program memiliki banyak masalah. Oleh karena itu, dibutuhkannya sistem baru untuk memberikan tugas kepada pelajar bidang informatika. Sistem baru yang dimaksud tentunya untuk melakukan penilaian secara otomatis. Sebuah sistem yang mengambil kode program pelajar dan memberikan sebuah nilai numerik yang menandakan hasil dari kode program tersebut [3]. Suatu hal yang menarik, Tugas kode program dapat dibagi menjadi 2 jenis yaitu tugas individu dan tugas kelompok. Pada tugas kelompok merupakan tugas yang ditanggung oleh banyak pelajar, biasanya program yang dibuat memiliki antarmuka dan harus diperiksa oleh pengguna khusus yang mengetahui fitur-fitur yang dibutuhkan. Sedangkan tugas individu merupakan sebuah tugas yang diberikan untuk satu individu, biasanya program yang dibuat bersifat algoritmik dan tidak memerlukan antarmuka untuk dijalankan. Program algoritmik sebuah jenis program yang dibuat berdasarkan algoritma untuk menyelesaikan masalah tertentu. Algoritma sendiri adalah langkah-langkah dalam pemecahan masalah secara sistematis [4]. Algoritma itu

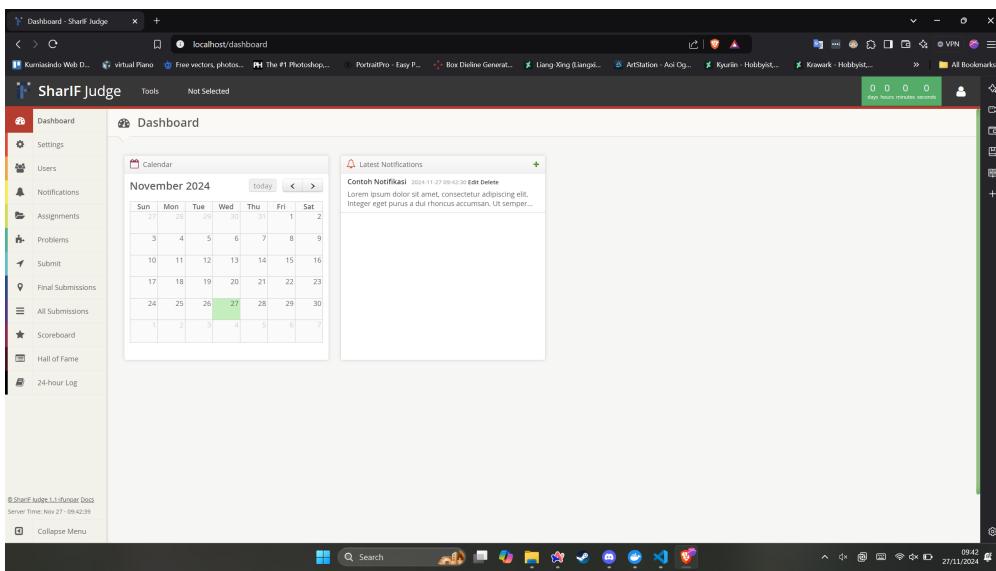
seperti resep makanan, dimana akan ada bahan-bahan yang dibutuhkan dan serangkaian langkah untuk membuat suatu makanan yang dijelaskan.

Sebagian besar program yang bersifat algoritmik hanya perlu mengambil *input* dari *input* standar seperti angka, huruf, dan sebuah kata atau kalimat dengan format yang sudah ditentukan, seolah-olah *input* ini merupakan *output* dari program lain. Kemudian program algoritmik akan memproses *input* tersebut dalam komputer dan mengeluarkan hasil komputasinya dalam format yang sudah ditentukan untuk dibaca oleh program lain dan memanfaatkan hasil komputasi tersebut. Singkatnya, program algoritmik itu seperti filter antar program. Dengan ini, sistem penilaian secara otomatis dapat dibuat dengan membuat sebuah program yang mengambil kode program, memasukkan *input* sesuai format ke dalam program tersebut, membaca hasil keluaran program, dan menilai hasil keluaran program tersebut [3]. Sistem penilaian otomatis ini diberikan nama *Online Judge*. Terlebih lagi sistem ini dapat dilakukan secara *offline* maupun *online*. Gambar 1.2 menunjukkan bagaimana *online judge* berintegrasi dengan sistem pemberian tugas yang sudah ada.



Gambar 1.2: Sistem Integrasi oleh *Online Judge*

Tugas pemrograman sudah menjadi keseharian dalam pembelajaran pada bidang informatika. Termasuk pada perguruan tinggi pada bidang informatika, maka *online judge* menjadi sebuah kebutuhan termasuk pada Universitas Katolik Parahyangan atau yang biasa disebut UNPAR. *Online Judge* yang digunakan oleh UNPAR dinamakan SharIF-Judge [5] yang merupakan hasil dimodifikasi oleh Stillmen Vallian terhadap Sharif-Judge [6] buatan Mohammad Javad Naderi yang dibuat menggunakan *framework* CodeIgniter dan Bash. Gambar 1.3 merupakan halaman utama setelah masuk ke dalam SharIF-Judge.



Gambar 1.3: Tampilan Awal SharIF Judge

Ujian juga merupakan sebuah bentuk penilaian dari pengajar kepada pelajarnya. Tentunya pelajar maupun mahasiswa ingin memperoleh nilai yang memuaskan dalam ujiannya. Banyak cara yang dilakukan oleh pelajar maupun mahasiswa untuk memperoleh nilai tersebut, salah satunya adalah dengan melakukan kecurangan yaitu *copy paste* atau menyalin jawaban teman atau rekan mereka [1]. Praktek ini diperparah jika ujian dilakukan secara *online*, dikarenakan pelajar dapat mengakses berbagai fasilitas di internet. Oleh karena itu, diperlukannya sebuah sistem pada sistem

online judge untuk mengawasi saat terjadinya ujian online.

Pada saat siswa mengerjakan tugas maupun ujian pembuatan kode program, umumnya pekerjaan kode tersebut dilakukan pada aplikasi eksternal seperti *visual studio code* atau *notepad*. Hal ini juga terjadi pada sistem dalam UNPAR dimana mahasiswa akan membuat kode program pada aplikasi eksternal. Ini membuat pengawasan saat pembuatan kode program lebih sulit untuk dilakukan, terlebih jika ujian dilakukan secara *online*. Maka dari itu, Nicholas Aditya Halim memodifikasi SharIF Judge agar semua sistem pemberian tugas seperti pada Gambar 1.2 dapat dilakukan dalam sistem yang sama yaitu pada SharIF Judge. Sistem yang bangun oleh Nicholas Aditya Halim adalah “Implementasi editor kode pada Sharif Judge” [7], dimana SharIF Judge ditambahkan sebuah *Integrated Development Environment* atau yang disebut dengan IDE. IDE merupakan sebuah sistem yang memiliki kemampuan untuk membuat kode dalam editor kode dan menjalankan kode program tersebut. Dengan adanya IDE, seluruh proses pembuatan kode program dapat dilakukan dalam SharIF Judge. Maka dari itu, seluruh proses sistem pemberian tugas dapat dilakukan dalam satu sistem saja, yaitu SharIF Judge.

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

1.2 Rumusan Masalah

Rumusan Masalah yang akan dibahas pada tugas akhir ini adalah:

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

1.3 Tujuan

Tujuan yang ingin dicapai skripsi ini adalah sebagai berikut:

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

1.4 Batasan Masalah

Pada pelaksanaan tugas akhir ini terhadap batasan sebagai berikut:

- Perangkat lunak SharIF Judge hanya digunakan pada lingkungan Teknik Informatika Unpar.
- Perangkat lunak hanya dapat diuji pada mata kuliah pemrograman di mana dosen pembimbing terlibat.

1.5 Metodologi

Metodologi penggerjaan tugas akhir ini adalah sebagai berikut:

1. Melakukan studi mengenai komponen yang diperlukan untuk membuat sistem perekaman dan pemutaran ulang ketikan pada IDE berbasis web.
2. Merancang sistem perekaman dan pemutaran ulang ketikan berbasis web untuk SharIF Judge
3. Mengimplementasikan IDE berbasis web pada SharIF Judge.
4. Melakukan pengujian dan eksperimen.
5. Menulis dokumen tugas akhir.

1.6 Sistematika Pembahasan

Sistematika pembahasan skripsi ini adalah sebagai berikut:

- **Bab 1:** Pendahuluan
Membahas latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi, dan sistematika pembahasan.
- **Bab 2:** Landasan Teori
Membahas teori-teori yang berhubungan dengan penelitian ini, yaitu SharIF Judge, CodeIgniter 3, Twig, IDE, dan Ace.
- **Bab 3:** Analisis
Membahas analisis terhadap perangkat lunak SharIF Judge dan IDE pada SharIF Judge.
- **Bab 4:** Perancangan
Membahas perancangan fitur yang diimplementasikan pada SharIF Judge.
- **Bab 5:** Implementasi dan Pengujian
Membahas implementasi fitur pada SharIF Judge dan pengujian yang dilakukan.
- **Bab 6:** Kesimpulan dan Saran
Membahas kesimpulan dari penelitian ini dan saran untuk penelitian berikutnya.

BAB 2

LANDASAN TEORI

2.1 SharIF Judge

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

2.1.1 Instalasi

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

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

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

1. Unduh versi terakhir dari Sharif Judge dan menempatkannya pada direktori publik.
2. Pindahkan folder `system` dan `application` ke luar direktori publik. Kemudian simpan alamatnya pada `index.php`.
3. Buat sebuah *Database MySQL* atau PostgreSQL.
4. Atur pengaturan koneksi `database` pada `application/config/database.php`.
5. Atur pengaturan koneksi RADIUS dan SMTP pada `application/config/secrets.php` jika dibutuhkan.
6. Atur agar direktori `application/cache/Twig` dapat ditulis oleh php.
7. Buka halaman utama SharIF Judge pada *browser* dan ikuti proses instalasi.
8. Log in dengan akun admin
9. Pindahkan folder `tester` dan `assignments` ke luar direktori publik. Kemudian simpan alamatnya pada halaman pengaturan.

2.1.2 Users

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

1. *admin*
2. *head instructor*
3. *instructor*
4. *student*

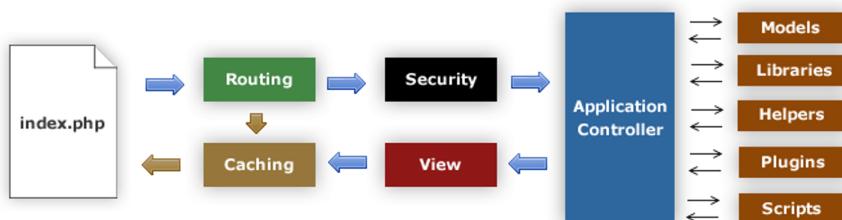
Setiap *role* memiliki akses pada aksi yang berbeda berdasarkan *role*-nya. Tabel 2.1 merupakan aksi-aksi yang dapat dilakukan untuk setiap pengguna pada SharIF Judge.

Tabel 2.1: *Tabel fitur untuk setiap role*

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

2.2 CodeIgniter 3

CodeIgniter 3 adalah sebuah *framework opensource* untuk mempermudah pengguna dalam membangun sebuah aplikasi *website* menggunakan bahasa PHP. CodeIgniter 3 bertujuan untuk membantu pengguna dalam membangun sebuah aplikasi *website* lebih cepat dengan menyediakan *library* yang beragam dengan fungsi yang umum digunakan dan tampilan dan *logic* yang simpel. Gambar 2.1 merupakan bagaimana data mengalir pada sistem CodeIgniter.



Gambar 2.1: *Flow Chart* CodeIgniter

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

1. *index.php* berfungsi sebagai *front controller* yang akan melakukan inisiasi *resource* utama untuk menjalankan CodeIgniter.
2. Router meneliti *request* HTTP dan menentukan apa yang harus dilakukan dengan *request* tersebut.
3. Jika terdapat *file cache*, maka langsung dikirimkan ke *browser* melewati eksekusi sistem yang biasanya.

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.

2.2.1 Model-View-Controller

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

Model

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

Kode 2.1: Contoh *model*

```

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

```

Kode 2.1 merupakan contoh model kelas bernama `Blog_model` pada CodeIgniter. *Model* `Blog_model` dapat mengambil, menambahkan, dan memperbarui *database* bernama ‘entries’. File *model* tersebut akan disimpan pada `application/models/Blog_model`. Selanjutnya, pengguna dapat memanggil *Model* tersebut pada *file controller* (akan dijelaskan pada bagian [Controller](#)) untuk memanggil model pada Kode 2.1 dengan menggunakan notasi sebagai berikut:

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

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

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

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

View

View adalah informasi yang akan di tunjukkan kepada user. Biasanya *view* merupakan sebuah halaman web, tetapi pada CodeIgniter, *view* dapat berupa pecahan halaman seperti *header*, *footer*, *sidebar*, dan lainnya. Pecahan halaman tersebut dapat dimasukkan secara fleksibel ke dalam *view* lainnya apabila dibutuhkan.

Kode 2.2: Contoh *view*

```

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

Kode 2.2 merupakan contoh dari *file view* pada CodeIgniter. File akan disimpan pada direktori `application/views/`. Untuk dapat diperlihatkan dibutuhkannya penggalian halaman pada *file controller* dengan cara sebagai berikut:

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

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

Controller

Controller adalah bagian utama dari aplikasi CodeIgniter, berfungsi sebagai perantara antara *model*, *view*, dan *resources* lainnya yang dibutuhkan untuk memproses HTTP *request* dan membuat sebuah web page. Kelas *Controller* akan mengekstensi `CI_Controller` dan disimpan pada `application/controllers/`. Contoh *controller* ditunjukkan pada Kode 2.3.

Kode 2.3: Contoh *controller*

```

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

Kode 2.3 berfungsi dalam mengembalikan string sesuai dengan fungsi *controller* yang dipanggil. Nama file *controller* pada direktori `application/controllers/blog.php` dan metode diatas akan dijadikan segmen pada URL seperti berikut:

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

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

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

```

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

Pada CodeIgniter, *model* dan *view* hanya dapat dimuat melalui controller. Seperti contoh, Kode 2.4 akan memuat *model* `blog` dan mengambil data dari *database*, lalu menampilkan *view* yang memuat data tersebut.

2.2.2 CodeIgniter URLs

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

```
example.com/news/article/my_article
```

Struktur URL pada CodeIgniter juga mengikuti pendekatan MVC (Referensi 2.2.1) dan biasanya memiliki struktur sebagai berikut:

```
example.com/class/function/ID
```

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

2.2.3 Helpers

Helpers merupakan sebuah kumpulan fungsi untuk membantu dalam sebuah kategori tertentu. *File helpers* terdapat pada direktori `system\helpers` atau `application\helpers`. Penggunaan *helpers* dalam *CodeIgniter* adalah dengan memuat file helpers dalam fungsi atau kelas *Controller* dengan cara seperti berikut ini:

```
$this->load->helper('name')
```

Setelah *helper* dimuat dalam fungsi, maka kumpulan fungsi dalam *file helper* dapat langsung dipanggil.

2.3 Twig

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

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

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

Kode 2.5: Contoh template Twig

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

Kode 2.5 merupakan contoh sebuah template Twig. Terdapat dua jenis *delimiter*, yaitu `{% ... %}` dan `{{ ... }}`. *Delimiter* `{% ... %}` digunakan untuk Menjalankan sebuah *statement* seperti *for-loops*, sedangkan *delimiter* `{{ ... }}` digunakan untuk mengubah sebuah *variable* atau *expression* menjadi nilai sesungguhnya.

2.4 Integrated Development Environment

Integrated Development Environment (IDE) merupakan sebuah aplikasi yang menyediakan berbagai peralatan yang diperlukan untuk membantu pengembangan perangkat lunak. Beberapa peralatan umum yang dimiliki oleh sebuah IDE adalah sebagai berikut:

- *Editor*
Editor teks sebagai tempat untuk mengetik kode, dapat dilengkapi dengan berbagai fitur seperti *syntax highlighting* (menampilkan teks dengan warna yang berbeda untuk menginkatkan keterbacaan kode) dan *word completion* (menampilkan prediksi kata yang sedang atau yang akan diketik pengguna).
- *Compiler*
Digunakan untuk menterjemahkan kode program yang dibuat pada editor teks ke dalam sebuah program yang dapat dijalankan oleh komputer.
- *Execution*
Menjalankan kode program yang sudah dikompilasi, dengan input jika dibutuhkan, dan mengembalikan hasilnya.

2.5 Ace

Ace merupakan *library* yang menyediakan sebuah editor kode yang dapat dimasukkan ke dalam sebuah web page dan dikembangkan menggunakan bahasa *Javascript*. Ace memiliki kemampuan yang sama seperti editor kode pada umumnya. Berikut merupakan beberapa fitur utama yang dimiliki oleh Ace:

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

Untuk mengintegrasikan *library* Ace dalam sebuah web page, Ace perlu ditanam dalam sebuah web page. Salah satu cara untuk menanam Ace ke dalam sebuah web page adalah dengan mem-*build library* atau menngunduh hasil dari *build* folder bernama *src* versi *pre-packaged* yang disediakan oleh Ace. Hasil dari *building library* Ace adalah sebuah folder yang dapat ditaruh dalam direktori lokal. Dalam folder tersebut, terdapat file *javascript* bernama *ace.js* yang dapat dipanggil dalam web page untuk menanam *library* Ace dalam web page. Cara untuk menanamkan file tersebut sama dengan cara untuk memasukkan file *javascript* pada umumnya yaitu dengan cara seperti berikut:

```
<script src="/ace-builds/ace.js" type="text/javascript"></script>
```

Setelah *library* Ace ditanam untuk mengakses berbagai macam fitur yang disediakan, maka kelas yang disediakan Ace dapat dipanggil. Berikut merupakan beberapa kelas penting yang terdapat pada *library* Ace adalah sebagai berikut:

- **Ace**
Kelas **Ace** merupakan kelas utama untuk menyiapkan editor kode Ace pada *browser*. Ace memiliki fungsi utama yang penting yaitu fungsi **edit** yang akan membuat sebuah editor dalam web page pada element beridentitas argumen yang diberikan saat dipanggil. Fungsi **edit** akan mengembalikan kelas **Editor**.
- **Editor**
Entri utama untuk fungsionalitas *library* Ace. Editor sendiri merepresentasikan editor kode yang dibuat pada web page. Editor juga menjadi kelas utama untuk mengakses kelas-kelas yang berhubungan dengan editor kode dengan mengakses kelas variable **Editor** seperti **session**

dalam editor kode. Kelas `Editor` sendiri dapat dikonfigurasikan sesuai dengan fungsi yang disediakan seperti `setTheme` yang mengubah warna editor kode sesuai dengan *theme* yang dipasang.

- **EditSession**

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

- **Anchor**

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

- **Document**

Menyimpan teks dokumen.

- **Range**

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

- **Selection**

Kelas ini menyimpan posisi yang dipilih oleh pengguna dalam editor.

- **Commands**

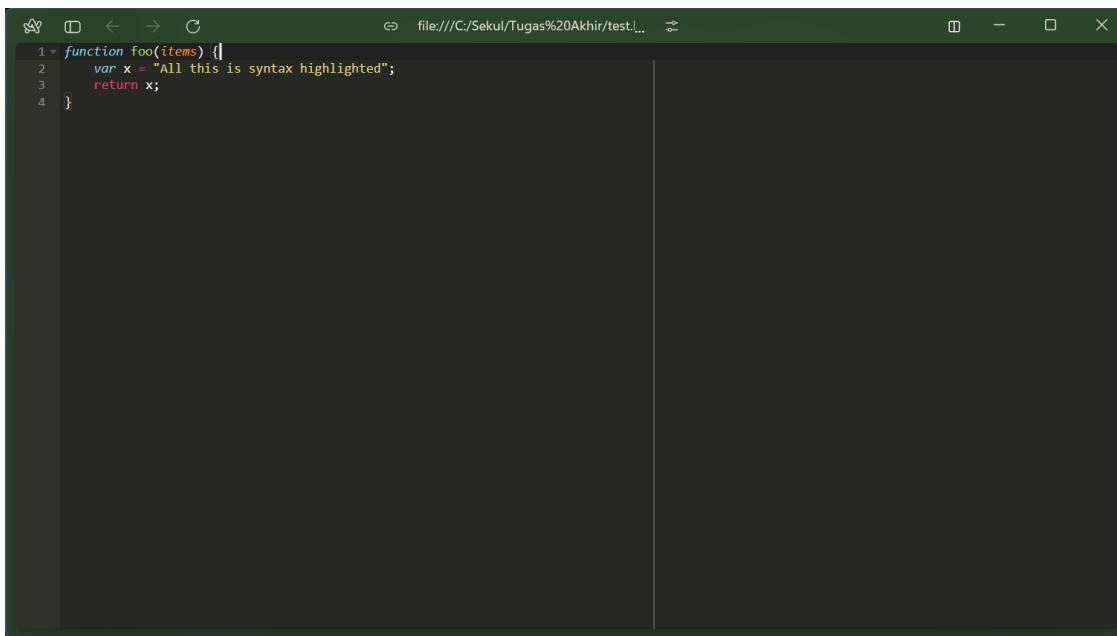
Kelas ini digunakan untuk menjalankan perintah pada sebuah editor. Contoh perintah yang sudah ada dalam editor yaitu *insert*, *copy*, *paste*.

Kode 2.6: Contoh kode penggunaan Ace

```

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

Kode 2.6 merupakan cara penggunaan Ace pada sebuah `div` dengan id `editor`. Ace juga memiliki beberapa konfigurasi, seperti contoh ini yaitu menggunakan tema *monokai* dan menggunakan *syntax highlighting* untuk bahasa pemrograman JavaScript. Gambar 2.2 menunjukkan hasil web page yang dibuka dalam *browser* menggunakan Kode 2.6.



Gambar 2.2: Hasil Web Page *Library Ace*

2.5.1 Perekaman Event

Pada editor kode Ace, disediakannya fungsi *event listener* atau pendengar *event* atau kejadian yang berhubungan dengan sebuah kelas. Pada *event listener* ini akan disediakannya sebuah fungsi *callback* yang akan dipanggil saat *event* tersebut terjadi. Berikut merupakan beberapa *event listener* dalam sebuah kelas:

- **Editor**

Pada editor sendiri disediakannya satu *event listener* yaitu `mouseup` yang akan mendengarkan saat melepaskan tombol pada tetikus atau *mouse*.

- **EditSession**

Pada kelas `session` ada satu *event listener* yaitu `change` yang akan mendengarkan perubahan pada isi atau kode pada editor kode. Pada fungsi *callback* yang akan dijalankan oleh *event listener* ini akan diberikan parameter `delta` yang menunjukkan perubahan apa yang terjadi pada editor kode.

- **Selection**

Pada kelas `selection` ada beberapa *event listener* yaitu sebagai berikut:

- `changeCursor` : Mendengarkan perubahan pada kursor atau *anchor* dalam editor kode.
- `changeSelection` : Mendengarkan perubahan pemilihan isi kode dalam editor kode.

- **Commands**

Pada kelas ini tersedia dua *event listener* yaitu `exec` dan `afterExec`. `exec` akan mendengarkan saat perintah akan dijalankan pada editor kode, sedangkan `afterExec` akan mendengarkan perintah yang sudah selesai dijalankan pada editor kode. Pada fungsi *callback* yang akan dijalankan oleh *event listener* ini akan diberikan perintah yang dijalankan oleh kelas `Commands`.

Untuk menggunakan fungsi *event listener* pada kelas yang diinginkan, dibutuhkan fungsi `on` pada kelas tersebut. Fungsi `on` memiliki dua parameter yaitu nama *event* ingin didengar (`exec` atau `change`) dan sebuah fungsi *callback* yang akan dijalankan saat *event* terjadi. Kode 2.7 merupakan perubahan kode yang dilakukan dalam tag `<script>` pada Kode 2.6 agar perubahan isi editor dapat didengar.

Kode 2.7: Contoh kode event listener

```

1 <script src="/ace-builds/ace.js" type="text/javascript"></script>
2 <script>
3     var editor = ace.edit("editor");

```

```

4 editor.setTheme("ace/theme/monokai");
5 editor.session.setMode("ace/mode/javascript");
6
7 editor.session.on("change", (delta) => {
8     console.log(delta);
9     // Contoh Keluaran :
10    //
11    //   action: "insert"
12    //   end: {row: 3, column: 5}
13    //   id: 1
14    //   lines: ['a']
15    //   start: {row: 3, column: 4}
16    //
17 });
18 </script>

```

Kode 2.7 akan menggunakan *event listener change* dalam kelas `EditSession`, dengan mengakses kelas `EditSession` melalui `editor` yang dinamakan `session`. Pada kelas tersebut akan dijalankan fungsi `on` dengan parameter “`change`” dan sebuah fungsi anonimous sebagai fungsi *callback* yang akan memprint ke `console` isi perubahan pada editor kode.

2.6 Chart.js

Chart.js merupakan sebuah *library javascript open-source* untuk membuat visualisasi data bagan interaktif berbasis `canvas` dalam web page [8]. Chart.js memiliki fitur-fitur yang dapat digunakan untuk mendukung dan mempermudah visualisasi data dalam web page. Berikut merupakan beberapa fitur yang dimiliki oleh Chart.js:

- Chart.js menyediakan berbagai tipe bagan yang dapat digunakan dan juga memiliki opsi penyesuaian yang sering digunakan.
- Chart.js memiliki konfigurasi bawaan yang bagus dan mudah untuk diintegrasikan dalam sebuah web page.
- Chart.js menggunakan *canvas HTML5 rendering* yang membuat sangat cepat terutama untuk data yang besar.

Identik dengan *library Ace* untuk menintegrasikan *library Chart.js* dalam sebuah web page, *library Chart.js* dapat dibuild dan dimasukkan ke dalam folder projek dan menambahkan file `javascript` bernama `chart.js` dalam folder `dist` ke dalam web page menggunakan cara yang identik dengan cara memasukkan file `javascript` pada umumnya yaitu dengan cara sebagai berikut:

```
<script src="/chartjs/dist/chart.js" type="text/javascript"></script>
```

Setelah itu *library Chart.js* dapat digunakan dengan membuat sebuah kelas `javascript` baru bernama `Chart`. Untuk membuat kelas `Chart` dibutuhkan 2 argumen yaitu elemen `canvas` dalam HTML web page dan sebuah objek `javascript` yang dapat diisi dengan opsi-opsi yang diinginkan dengan menspesifikasi `key` dan `value` yang sesuai dengan opsi yang diinginkan. Berikut merupakan beberapa `key` dan `value` yang dapat digunakan ada dalam opsi *library Chart.js*:

- `type`
`type` hanya menerima sebuah kata yang menjadi tipe utama bagan yang dibuat oleh *library Chart.js*, tetapi tipe ini dapat berubah mengikuti data yang diberikan. Berikut merupakan beberapa tipe-tipe yang ada dalam *library Chart.js*:
 - `bar`
 Bagan `bar` menyediakan cara untuk menvisualisasikan data sebagai batang vertikal. Bagan ini biasanya digunakan untuk menunjukkan tren dan perbandingan beberapa set data secara berdampingan.
 - `line`
 Bagan `line` adalah cara menvisualisasikan data sebagai titik data yang disambungkan dengan garis. Identik dengan Bagan `bar`, Bagan `line` juga digunakan untuk menunjukkan tren dan perbandingan beberapa set data secara berdampingan.
- `data`
`data` sendiri menerima `value` objek `javascript` dengan isi `labels` dan `dataset`. Kedua `key`

menerima sebuah *array* dengan isi yang berbeda. **labels** hanya menerima sebuah *array* berisi teks untuk label data horizontal atau vertikal. **dataset** menerima *array* primitive type, *array* dengan isi *array*, dan *array objek*. *objek* dalam **dataset** menerima *key data* dan juga memiliki beberapa *key* opsi yaitu **label** untuk melabelkan data dalam horizontal maupun vertikal. *Key data* menerima *array* dengan isi *array objek* dengan data yang ingin divisualisasikan.

- **options**

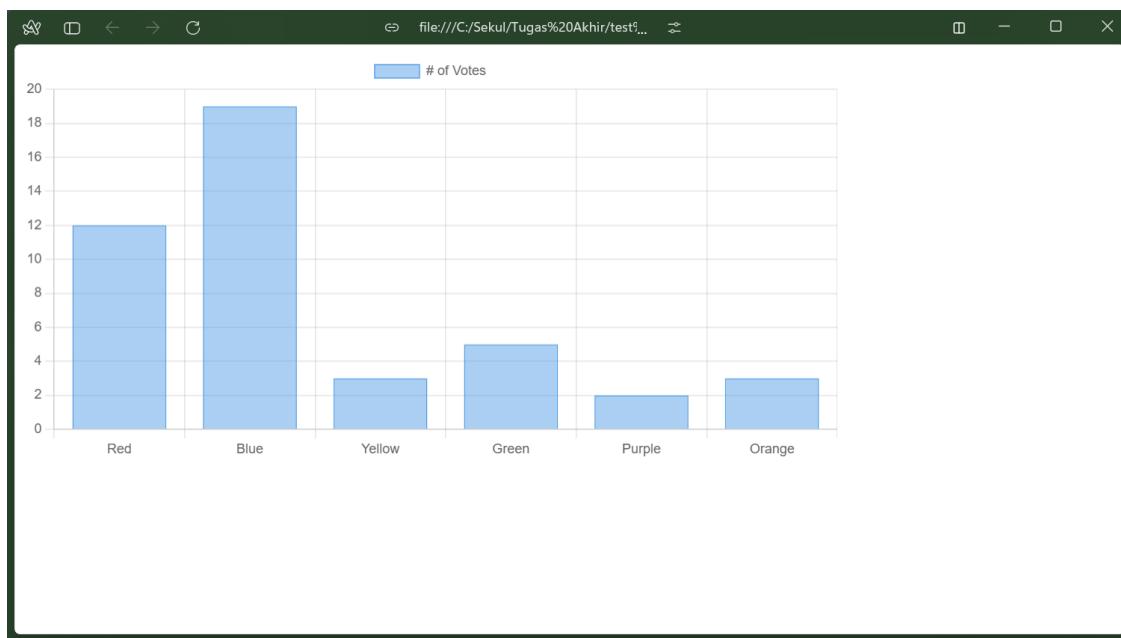
Key options merupakan fitur utama dari kelas **Chart** dan hanya menerima sebuah objek *javascript* yang memiliki banyak *key* untuk menyesuaikan bagan yang dibuat oleh *library Chart.js*. Salah satu *key* dalam **options** adalah **scales** yang digunakan untuk mengatur data yang ditampilkan untuk aksis X dan Y. Salah satu pengaturannya adalah untuk menumpuk data dengan data yang sama di aksis yang sama yaitu dengan menggunakan *key stacked* dalam aksis X atau Y.

Kode 2.8: Contoh kode penggunaan Chart.js

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <body>
4 <div>
5   <canvas id="myChart"></canvas>
6 </div>
7
8 <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
9
10<script>
11 const ctx = document.getElementById('myChart');
12
13 new Chart(ctx, {
14   type: 'bar',
15   data: {
16     labels: ['Red', 'Blue', 'Yellow', 'Green', 'Purple', 'Orange'],
17     datasets: [
18       {
19         label: '# of Votes',
20         data: [12, 19, 3, 5, 2, 3],
21         borderWidth: 1
22       }
23     ],
24   options: {
25     scales: {
26       y: {
27         beginAtZero: true
28       }
29     }
30   });
31 </script>
32 </body>
33 </html>
```

Kode 2.8 merupakan contoh penggunaan *library Chart.js* pada sebuah *canvas* dengan id **myChart**. *Key options* pada contoh ini menggunakan **beginAtZero** yang membuat data dimulai dari nol. Gambar 2.3 merupakan hasil web page yang dibuka dalam *browser* dengan Kode 2.8.



Gambar 2.3: Hasil Web Page *Library* Chart.js

BAB 3

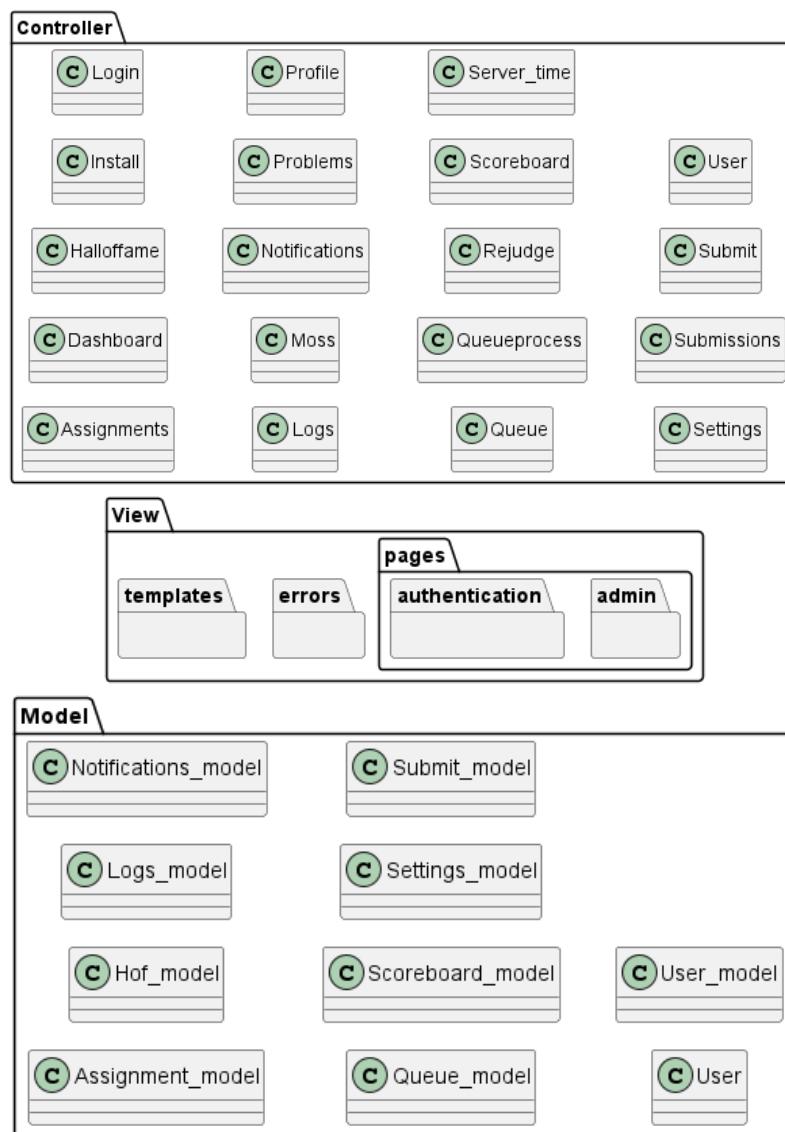
ANALISIS

3.1 Analisis Sistem Kini

Seperi yang sudah dibahas pada subbab 2.1, SharIF Judge merupakan sebuah website judge yang dimodifikasi sesuai dengan kebutuhan Teknik Informatika UNPAR. Analisis diawali dengan MVC aplikasi SharIF Judge. Berikut merupakan hasil eksplorasi SharIF Judge yang telah dilakukan:

3.1.1 Model, View, Controller

SharIF Judge menggunakan *framework* CodeIgniter 3 yang berbasis arsitektur Model-View-Controller seperti yang dijelaskan pada subbab 2.2.1. Gambar 3.1 merupakan kelas diagram struktur MVC pada SharIF Judge.



Gambar 3.1: Struktur MVC pada SharIF Judge

Berikut merupakan hasil eksplorasi dari struktur MVC pada SharIF Judge:

Model

Analisis MVC akan dimulai dengan *model* yang berada pada direktori `application/models`. Direktori *Model* berisi kelas-kelas yang digunakan untuk mengelola dan mengembalikan data dari *database*. Gambar 3.2 merupakan struktur kelas *model* dalam SharIF Judge.



Gambar 3.2: Struktur Kelas Model pada SharIF Judge

Berikut merupakan penjelasan dari kelas *model* dan fungsi-fungsinya yang terdapat pada SharIF Judge:

- **Assignment_model.php**

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

- **add_assignment(\$id, \$edit)**
Menambahkan atau memperbarui sebuah *assignment*.
- **delete_assignment(\$assignment_id)**
Menghapus sebuah *assignment*.
- **all_assignments()**
Mengembalikan daftar semua *assignment* dan informasinya.
- **new_assignment_id()**
Mendapatkan nomor terkecil dan dapat digunakan sebagai *id assignment* terbaru.
- **all_problems(\$assignment_id)**
Mengembalikan daftar semua *problems* dari sebuah *assignment*.
- **problem_info(\$assignment_id, \$problem_id)**
Mengembalikan semua informasi sebuah *problem*
- **assignment_info(\$assignment_id)**
Mengembalikan semua informasi sebuah *assignment*
- **is_participant(\$participants, \$username)**
Mengembalikan sebuah *boolean* yang menyatakan bahwa *\$username* terdapat dalam *\$participants*.
- **increase_total_submits(\$assignment_id)**

Menambahkan jumlah *total submits* sebanyak satu pada sebuah assignment.

- `set_moss_time($assignment_id)`
Memperbarui “*Moss Update Time*” pada sebuah *assignment*.
- `get_moss_time($assignment_id)`
Mengembalikan “*Moss Update Time*” pada sebuah assignment.
- `save_problem_description($assignment_id, $problem_id, $text, $type)`
Menambahkan atau memperbarui deskripsi pada sebuah *problem*.
- `_update_coefficients($a_id, $extra_time, $finish_time, $new_late_rule)`
Memperbarui koefisien dari sebuah *assignment*.

- **Hof_model.php**

Model ini digunakan untuk mengembalikan informasi yang digunakan dalam *hall of fame* dari tabel **submissions**. Fungsi yang dimiliki adalah sebagai berikut:

- `get_all_final_submission()`
Mengembalikan seluruh total nilai *final submission* untuk semua *user*.
- `get_all_user_assignments($username)`
Mengembalikan nilai *final submission* pada semua problem untuk *user* tertentu.

- **Logs_model.php**

Model ini berfungsi untuk mengelola tabel **logins** dan mengembalikan catatan *login*. Fungsi yang dimiliki adalah sebagai berikut:

- `insert_to_logs($username, $ip_address)`
Mencatat *login* sebuah *user* dan menghapus catatan jika melebihi 24 jam.
- `get_all_logs()`
Mengembalikan semua catatan *login*.

- **Notifications_model.php**

Model ini digunakan untuk mengelola tabel **notifications**. Fungsi yang dimiliki adalah sebagai berikut:

- `get_all_notifications()`
Mengembalikan semua *notifications*.
- `get_latest_notifications()`
Mengembalikan 10 *notifications* terbaru.
- `add_notification($title, $text)`
Menambahkan *notification* baru.
- `update_notification($id, $title, $text)`
Memperbarui sebuah *notification*.
- `delete_notification($id)`
Menghapus sebuah *notification*.
- `get_notification($notif_id)`
Mengembalikan sebuah *notification*.
- `have_new_notification($time)`
Mengembalikan sebuah *boolean* yang menyatakan bahwa terdapatnya *notification* baru.

- **Queue_model.php**

Model ini digunakan untuk mengelola tabel **queue** dan menampilkan data **queue**. Fungsi yang dimiliki adalah sebagai berikut:

- `in_queue($username, $assignment, $problem)`
Mengembalikan sebuah *boolean* yang menyatakan bahwa *username* masih memiliki *queue* dalam sebuah *problem*.
- `get_queue()`
Mengambil semua *submission queue*.
- `empty_queue()`
Menghapus semua *queue*.
- `add_to_queue($submit_info)`

- Menambahkan sebuah *submission* ke dalam *queue*.
 - `rejudge($assignment_id, $problem_id)`
Menambahkan seluruh *submissions* dalam sebuah *problem* ke dalam *queue* untuk dinilai ulang.
 - `rejudge_single($submission)`
Menambahkan sebuah *submission* ke dalam *queue* untuk dinilai ulang.
 - `get_first_item()`
Mengembalikan *item* pertama dalam tabel *queue*.
 - `remove_item($username, $assignment, $problem, $submit_id)`
Menghapus sebuah *item* tertentu dalam tabel *queue*.
 - `save_judge_result_in_db ($submission, $type)`
Menyimpan hasil penilaian *judge* ke dalam *database*.
 - `add_to_queue_exec($submit_info)`
Menambahkan sebuah *dummy submission* yang digunakan hanya untuk dijalankan ke dalam *queue*.
- **Scoreboard_model.php**
Model ini digunakan untuk mengelola tabel **scoreboard**. Fungsi yang dimiliki adalah sebagai berikut:
 - `_generate_scoreboard($assignment_id)`
Menghasilkan *scoreboard* untuk sebuah *assignment* dari nilai akhir semua *submission*.
 - `update_scoreboards()`
Memperbarui *scoreboard* untuk semua *assignment*.
 - `update_scoreboard($assignment_id)`
Memperbarui *scoreboard* untuk sebuah *assignment*.
 - `get_scoreboard($assignment_id)`
Mengembalikan *scoreboard* pada sebuah *assignment*.
- **Settings_model.php**
Model ini digunakan untuk mengelola tabel **settings**. Fungsi yang dimiliki adalah sebagai berikut:
 - `get_setting($key)`
Mengembalikan nilai dari sebuah `$key` pada tabel **settings**.
 - `set_setting($key, $value)`
Memperbarui nilai dari pada *setting* `$key`.
 - `get_all_settings()`
Mengembalikan seluruh **settings**.
 - `set_settings($settings)`
Memperbarui seluruh nilai perubahan **settings**.
- **Submit_model.php**
Model ini digunakan untuk mengelola tabel **submission**. Fungsi yang dimiliki adalah sebagai berikut:
 - `get_submission($username, $assignment, $problem, $submit_id)`
Mengembalikan sebuah baris data *submission* tertentu.
 - `get_final_submissions($a_id, $u_vl, $uname, $p_num, $fil_u, $fil_prob)`
Mengembalikan seluruh *final submission* pada sebuah *assignment*. *User* dengan role *student* hanya dapat melihat *final submission* dirinya sendiri.
 - `get_all_submissions($a_id, $u_vl, $uname, $p_num, $fil_u, $fil_prob)`
Mengembalikan seluruh *submission* pada sebuah *assignment*. *User* dengan role *student* hanya dapat melihat *submission* dirinya sendiri.
 - `count_final_submissions($a_id, $u_vl, $uname, $fil_u, $fil_prob)`
Mengembalikan jumlah *final submission* pada sebuah *assignment*.
 - `count_all_submissions($a_id, $u_vl, $uname, $fil_u, $fil_prob)`

Mengembalikan jumlah *submission* pada sebuah *assignment*.

- `set_final_submission($username, $assignment, $problem, $submit_id)`
Memperbarui sebuah *submission* menjadi *final submission*.
- `add_upload_only($submit_info)`
Menyimpan hasil *upload only problem* ke dalam tabel *database*.

- **User.php**

Model ini digunakan untuk menyimpan *settings* sebuah *user*. Fungsi yang dimiliki adalah sebagai berikut:

- `select_assignment($assignment_id)`
Menyimpan *assignment* yang dipilih oleh *user*.
- `save_widget_positions($positions)`
Menyimpan posisi *widget* sebuah *user*.
- `get_widget_positions()`
Mendapatkan posisi *widget* sebuah *user*.

- **User_model.php**

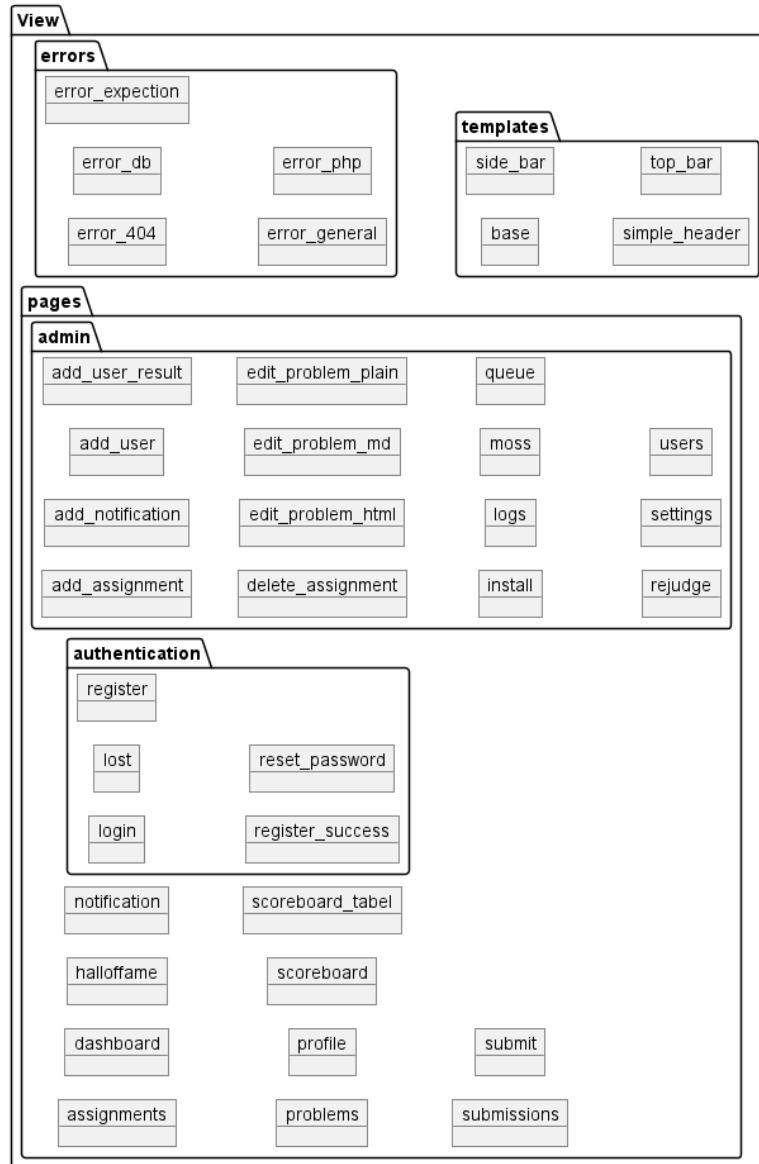
Model ini digunakan untuk mengelola tabel *users*. Fungsi yang dimiliki adalah sebagai berikut:

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

- Mengembalikan sebuah *user* yang memiliki id `$user_id`.
- `update_login_time($username)`
 - Memperbarui catatan *login* untuk sebuah *user*.

View

View merupakan tampilan yang menjadi perantara antara pengguna dan *sistem*. Pada SharIF Judge, *View* disimpan pada direktori `application/views` dan dibagi menjadi 3 direktori terpisah yaitu errors, pages, dan template. Gambar 3.3 merupakan struktur direktori *view* beserta *view* yang terdapat pada direktorinya dalam SharIF Judge.



Gambar 3.3: Struktur Direktori View pada SharIF Judge

Berikut merupakan penjelasan mengenai direktori penyimpanan untuk *view* pada SharIF Judge.

- **errors**

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

- `error_404`
- `error_db`

- error_expectation
- error_general
- error_php

- pages

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

- pages/admin

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

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

- pages/authentication

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

- * login.twig
- * lost.twig
- * register.twig
- * register_success.twig
- * reset_password.twig
- assignments.twig
- dashboard.twig
- halloffame.twig
- notification.twig
- problems.twig
- profile.twig
- scoreboard.twig
- scoreboard_tabel.twig
- submissions.twig
- submit.twig

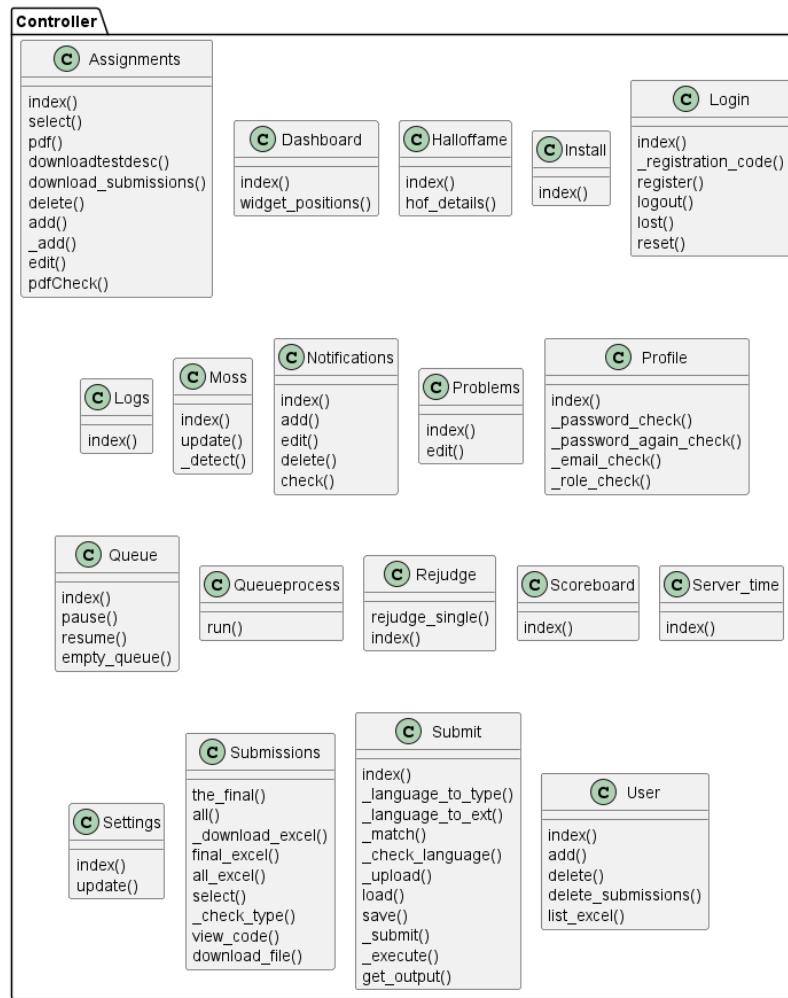
- templates

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

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

Controller

Pada bagian analisis MVC terakhir, terdapat *controller* yang berada pada direktori `application/controller`. Seperti yang dijelaskan pada subbab 2.2.1, *Controller* digunakan sebagai perantara antara *model*, *view*, dan *resources* lainnya yang dibutuhkan saat membuat sebuah web page. Direktori controller berisi kelas-kelas yang akan mengolah data yang didapat pada *model* dan menyatukan data tersebut ke dalam *views* yang akan ditampilkan kepada pengguna. Pada setiap kelas *controller*, terdapat fungsi `index()` yang menjadi fungsi utama saat kelas di akses oleh pengguna. Gambar 3.4 merupakan struktur kelas *controller* yang terdapat pada SharIF Judge.



Gambar 3.4: Struktur Kelas Controller pada SharIF Judge

Berikut merupakan file *controller* dan penjelasan fungsi-fungsinya yang terdapat pada SharIF Judge:

- **Assignments.php**

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

- **select()**

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

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

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

- **downloadtestsdesc(\$assignment_id)**

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

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

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

- `delete($assignment_id)`
Menghapus sebuah *assignment*.
- `add()`
Mendapatkan *input* dari pengguna untuk menambah atau memperbarui sebuah *assignment*.
- `_add()`
Menambahkan atau memperbarui sebuah *assignment*.
- `edit($assignment_id)`
Menandai *assignment* yang akan di *edit* dan memanggil fungsi `add`.
- `pdfCheck($assignment_id, $problem_id)`
Melakukan validasi ketersediaan pdf pada sebuah *assignment* atau pada sebuah *problem*.
- `index()`
Mengambil data dari `Assignment_model` dan menaruh data dan mengembalikan *views assignments.twig*. Gambar 3.5 menunjukkan hasil halaman Assignment.

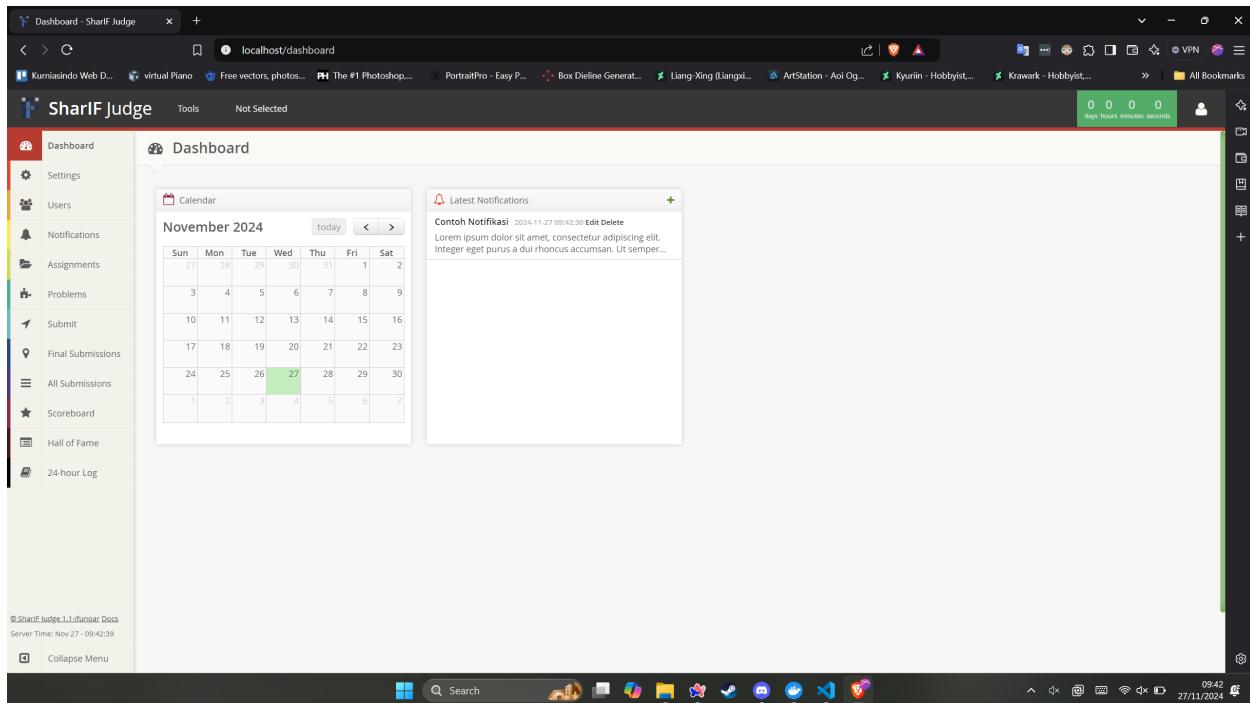
Select	Name	Problems	Submissions	Coefficient	Start Time	Finish Time	Status	PDF	Actions
<input checked="" type="checkbox"/>	Contoh Assignment	2.problems	0 submissions	100 %	2024-11-26 00:00:00	2024-11-30 00:00:00	Open		

Gambar 3.5: Halaman Assignments

- `Dashboard.php`

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

- `widget_positions()`
Menggunakan *ajax request* untuk menyimpan posisi *widget*.
- `index()`
Mendapatkan data dari beberapa model yaitu `Assignment_model`, `Settings_model`, `User`, dan `Notifications_model`. Data akan dimasukkan ke dalam `dashboard.twig` yang akan dikembalikan ke pengguna. Gambar 3.6 menunjukkan hasil halaman Dashboard yang dapat diakses oleh semua *role*.



Gambar 3.6: Halaman Dashboard

- **Halloffame.php**

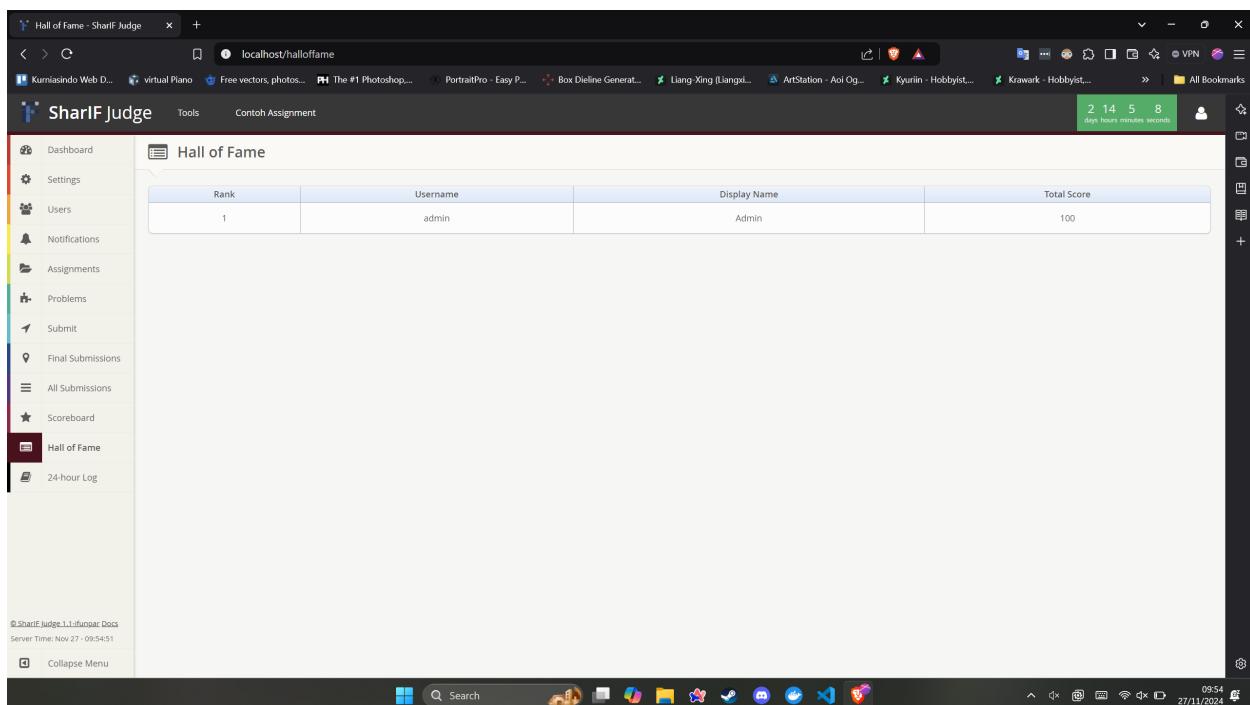
Berikut fungsi dengan penjelasannya pada *controller Halloffame.php*:

- **hof_details()**

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

- **index()**

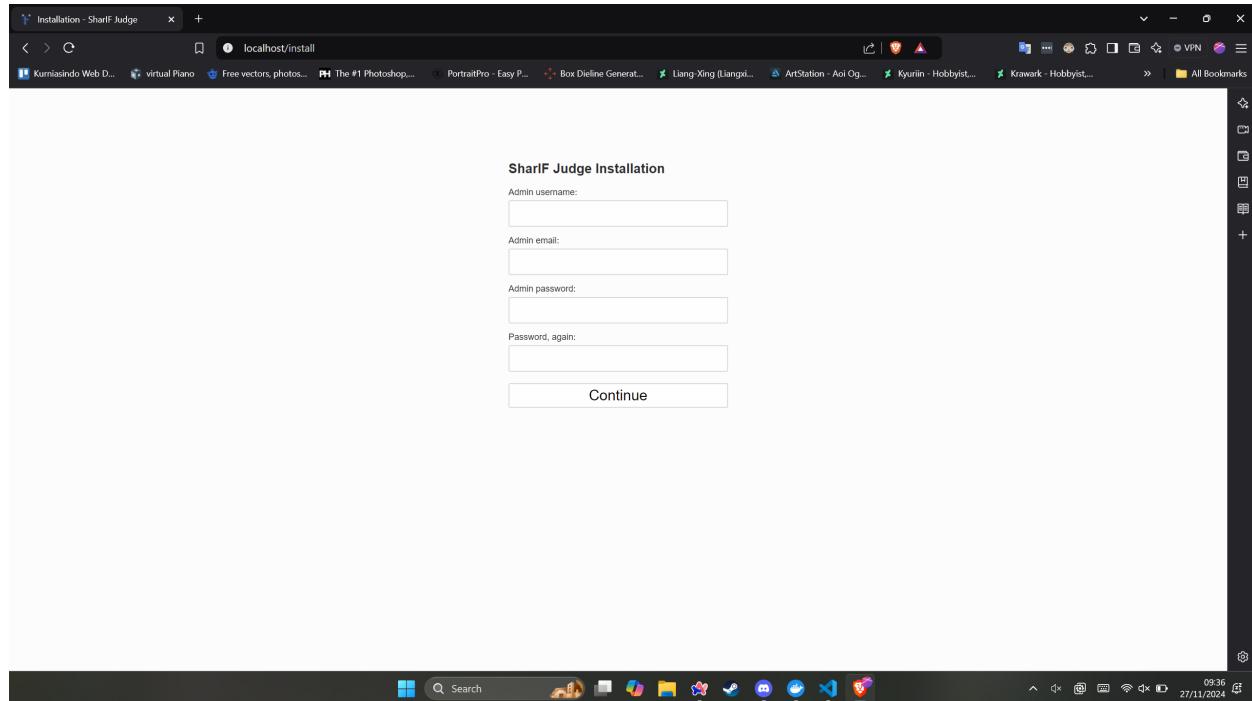
Mendapatkan data dari *Hof_model* dan mengembalikan *view halloffame.twig*. Gambar 3.7 menunjukkan hasil halaman Hall of Fame yang dapat diakses oleh semua *role*.



Gambar 3.7: Halaman Hall of Fame

- **Install.php**

Pada *controller* `Install.php` hanya ada satu fungsi yang menangani pembuatan seluruh tabel pada *database* yang dibutuhkan oleh SharIF Judge. Setelah membuat *database* akan mengembalikan *view* `install.twig` yang dapat diisi oleh pengguna tentang data *user* dengan role *admin* saat *form* di kirim. Gambar 3.8 menunjukkan hasil halaman Install.

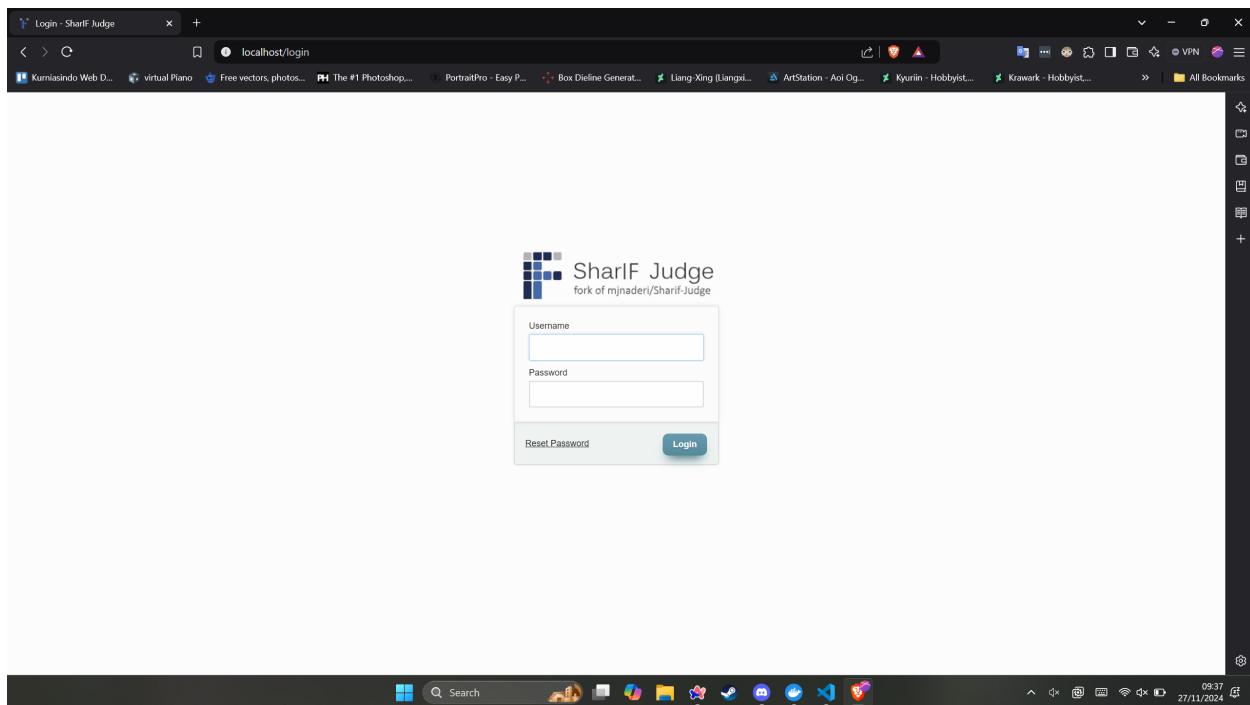


Gambar 3.8: Halaman Install

- **Login.php**

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

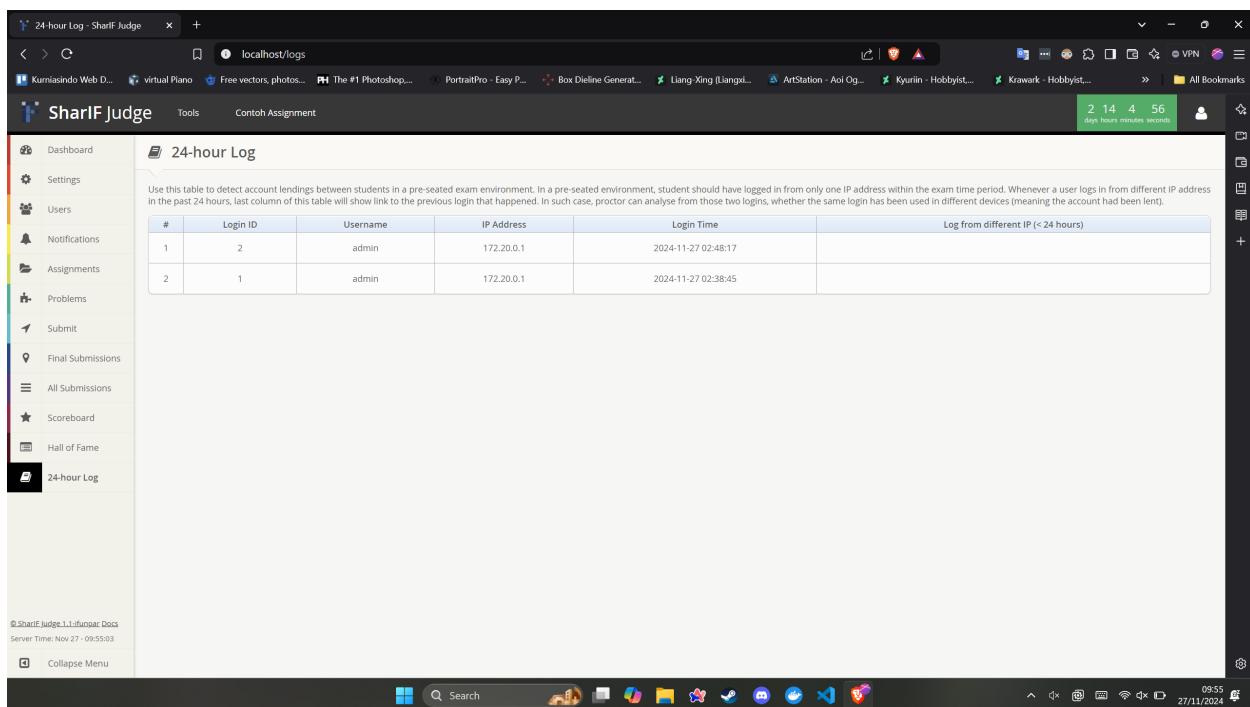
- `_registration_code($code)`
Melakukan validasi kode registrasi.
- `register()`
Menunjukkan halaman `register.twig` dan membuat *user* baru.
- `logout()`
Melakukan *Log out* dan mengalihkan ke halaman `login`.
- `lost()`
Mengirimkan email *reset password*.
- `reset($passchange_key)`
Melakukan *reset password* dengan halaman `reset_password.twig`.
- `index()`
Mengembalikan *view* `login.twig` dan memeriksa username dan password pada *form* saat di kirim. Gambar 3.9 menunjukkan hasil halaman Login.



Gambar 3.9: Halaman Login

- **Logs.php**

Pada *controller Logs.php* hanya memiliki satu fungsi yaitu `index()`, dimana fungsi tersebut akan mendapatkan data dari `Logs_model` dan memunculkan halaman `logs.twig`. Gambar 3.10 menunjukkan halaman Log yang dinamakan halaman 24-Hour Log.

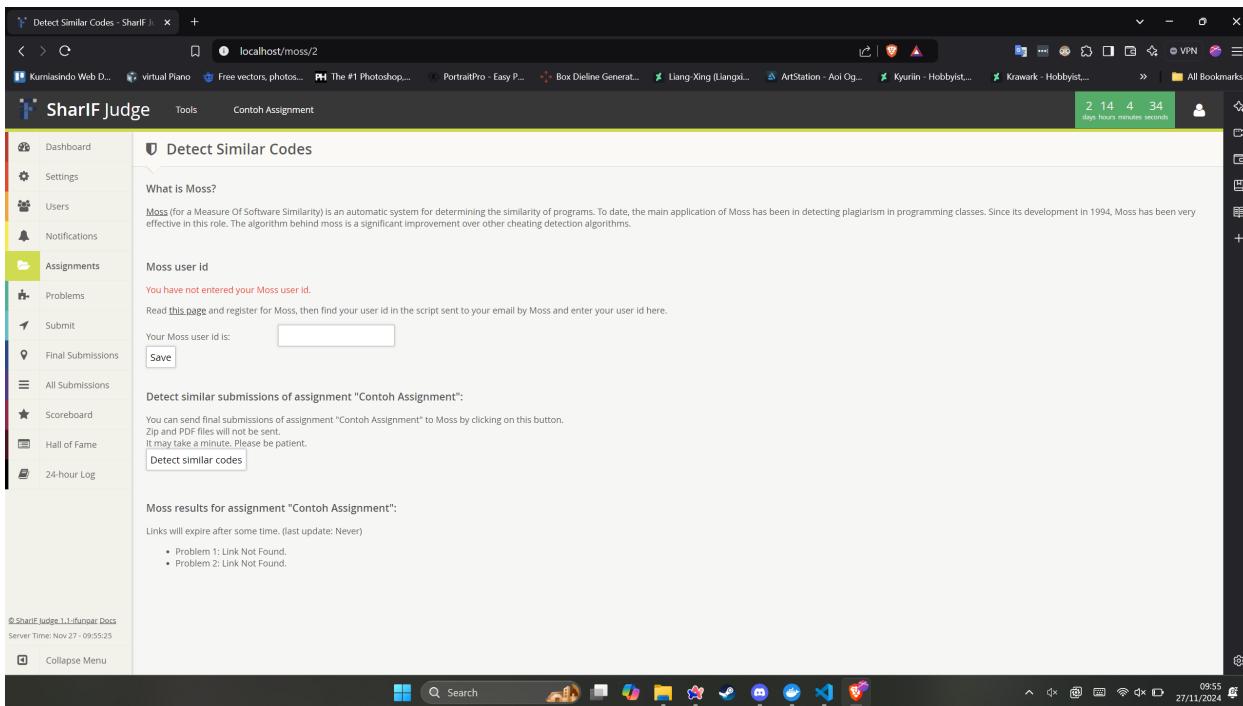


Gambar 3.10: Halaman 24-Hour Log

- **Moss.php**

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

- `update($assignment_id)`
Memperbarui *settings* dari masukkan `moss_userid` pengguna.
- `_detect($assignment_id)`
Melakukan pemeriksaan kesamaan kode dengan Moss.
- `index()`
Mengambil data dan memasukkannya ke dalam *view moss.twig*. Gambar 3.11 merupakan hasil halaman moss. Fungsi `_detect` juga akan dijalankan saat *form* terkirim.

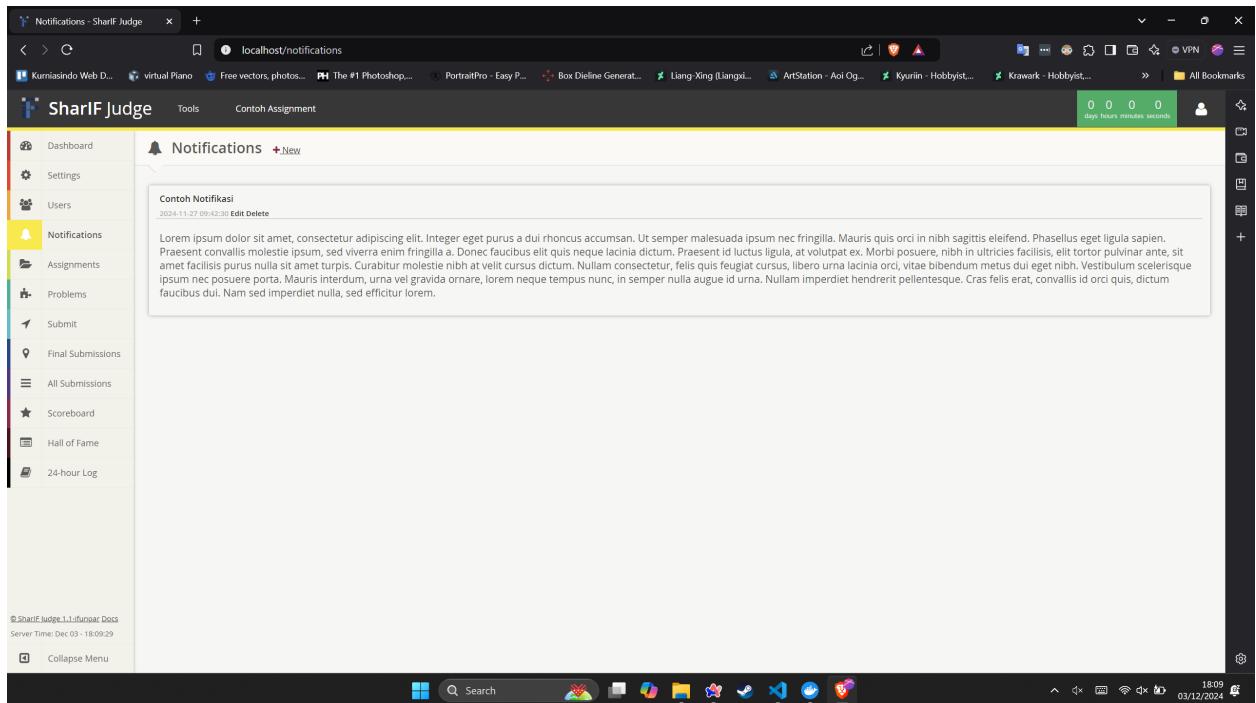


Gambar 3.11: Halaman Moss

- **Notifications.php**

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

- `add()`
Menambahkan atau memperbarui sebuah *notification*.
- `edit($notif_id)`
Menandai *notification* yang akan di *edit* dan memanggil fungsi `add`.
- `delete()`
Menghapus sebuah *notification*.
- `check()`
Menggunakan *ajax request* untuk mengetahui ketersediaan *notification* baru.
- `index()`
Mendapatkan data dari dua model yaitu `Assignment_model` dan `Notifications_model`. Data akan dimasukkan ke dalam *view notifications.twig* yang akan dikembalikan ke pengguna. Gambar 3.12 menunjukkan hasil halaman *Notifications*.



Gambar 3.12: Halaman Notifications

- **Problems.php**

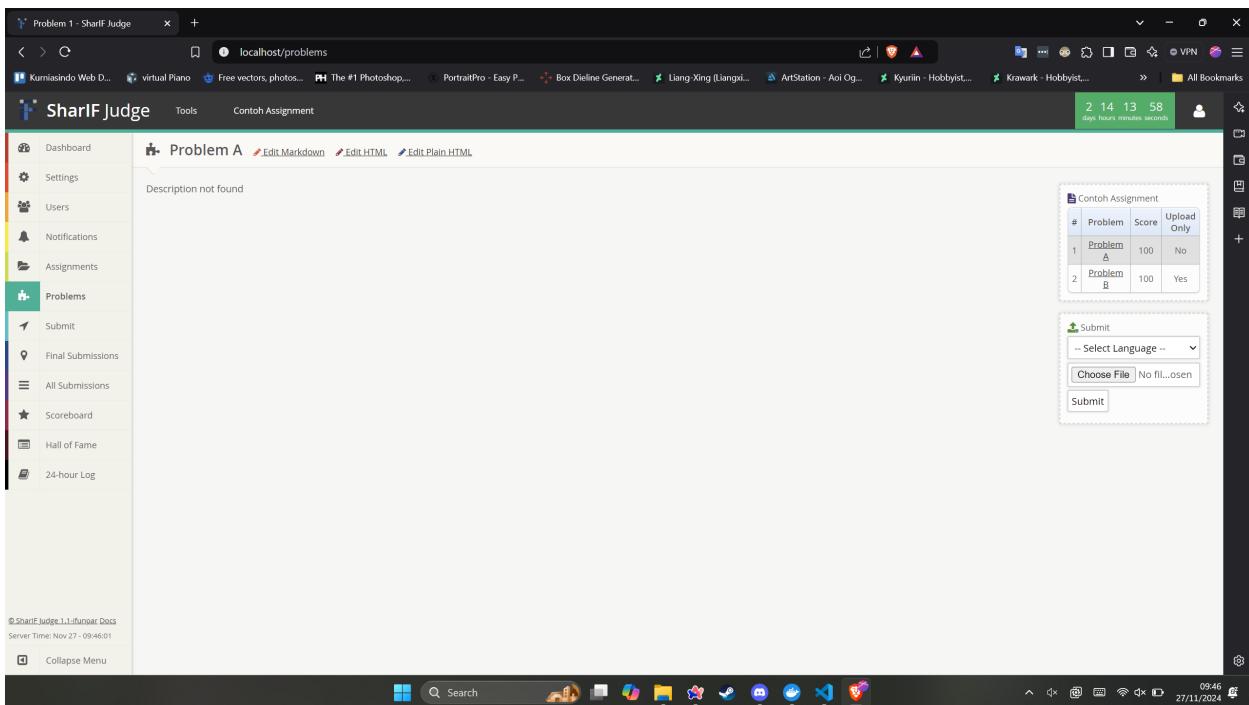
Berikut fungsi dengan penjelasannya pada *controller Notifications.php*:

- **edit()**

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

- **index()**

Mendapatkan data *problem* dari berbagai *model* sesuai dengan *assignment* yang dipilih dan menaruh data tersebut pada halaman *problems.twig* yang akan ditampilkan ke pengguna. Gambar 3.13 menunjukkan hasil halaman Problems.

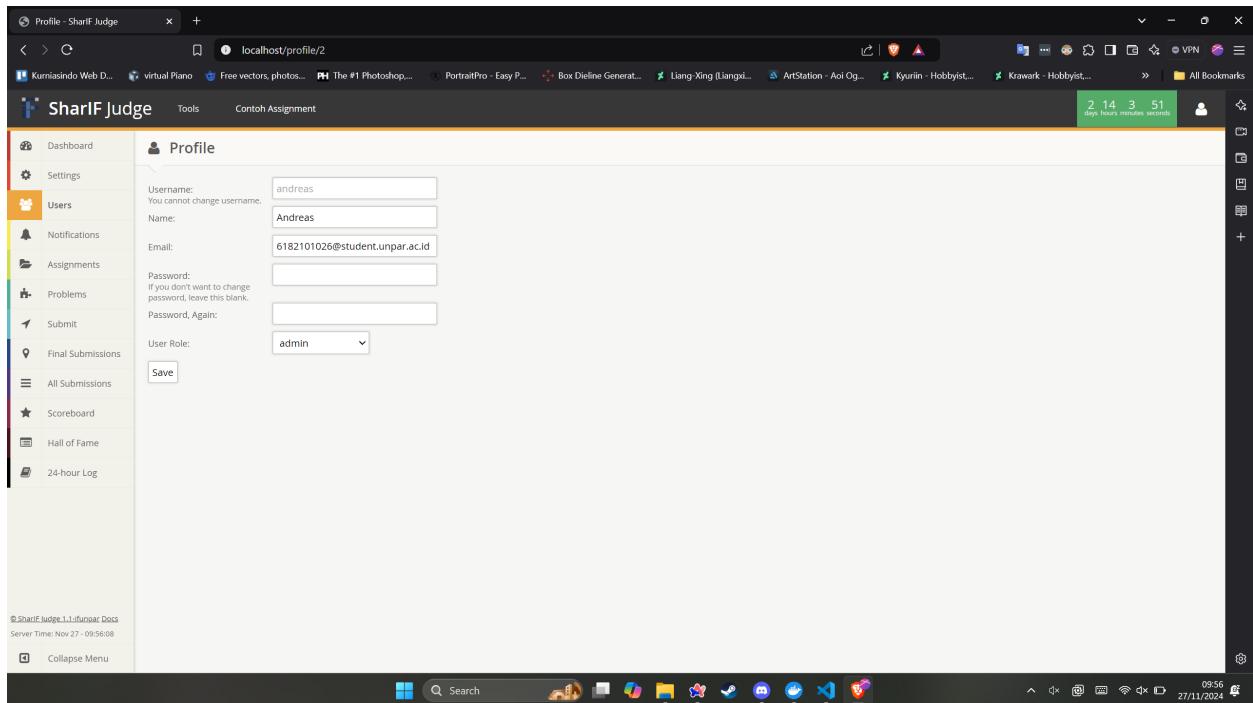


Gambar 3.13: Halaman Problems

- **Profile.php**

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

- `_password_check($str)`
Melakukan validasi *input password*.
- `_password_again_check($str)`
Melakukan validasi *input tulisan pengulangan password*.
- `_email_check($str)`
Melakukan validasi ketersediaan email pada *database*.
- `_role_check($str)`
Melakukan validasi *role* pengguna saat ingin mengubah *role user*.
- `index()`
Mendapatkan data dari berbagai *model* terutama dari *User* yang akan dimasukkan ke dalam *view profile.twig*. Fungsi ini juga menangani pengiriman *form* pembaharuan data *user* pengguna. Gambar 3.14 menunjukkan hasil halaman Profile.

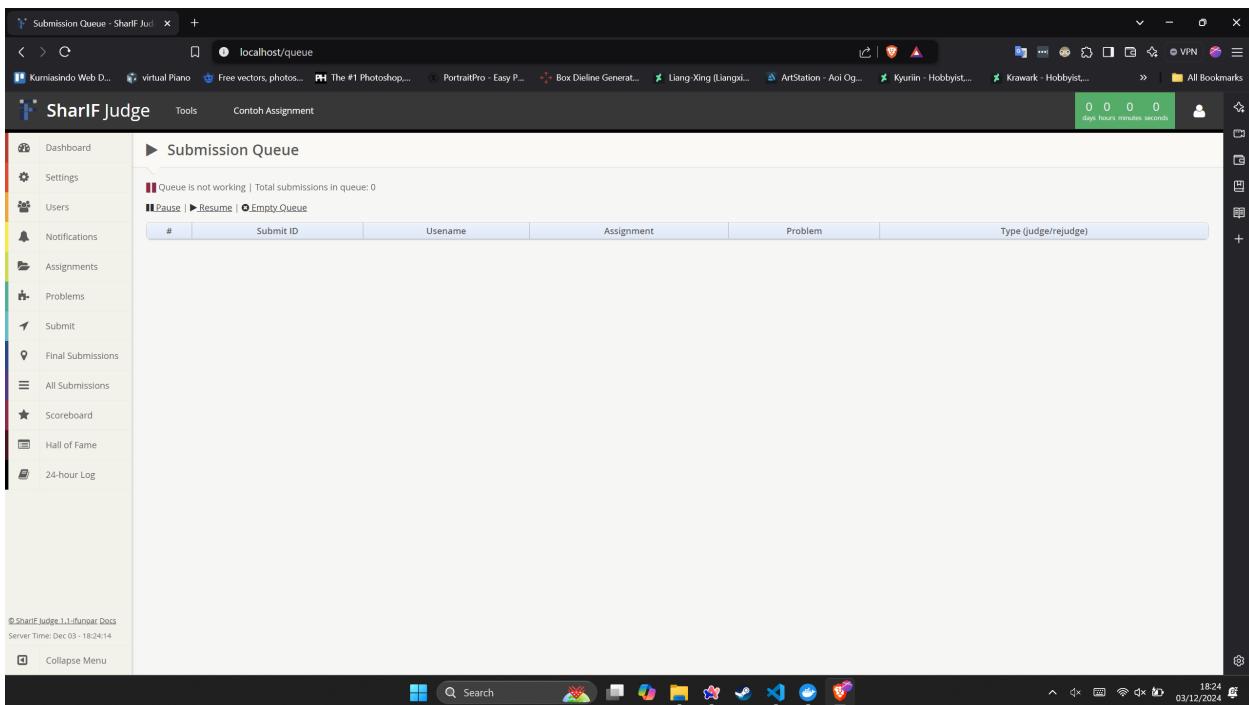


Gambar 3.14: Halaman Profile

- **Queue.php**

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

- **pause()**
Memberhentikan proses *queue*.
- **resume()**
Melanjutkan proses *queue*.
- **empty_queue()**
Menghapus semua *queue* yang ada.
- **index()**
Mendapatkan data dari *model Queue*, *Assignments_model*, dan *Settings_model* yang dipakai dalam *view queue.twig* dan ditampilkan kepada pengguna. Gambar 3.15 menunjukkan hasil halaman Queue.



Gambar 3.15: Halaman Queue

- **Queueprocess.php**

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

- **Rejudge.php**

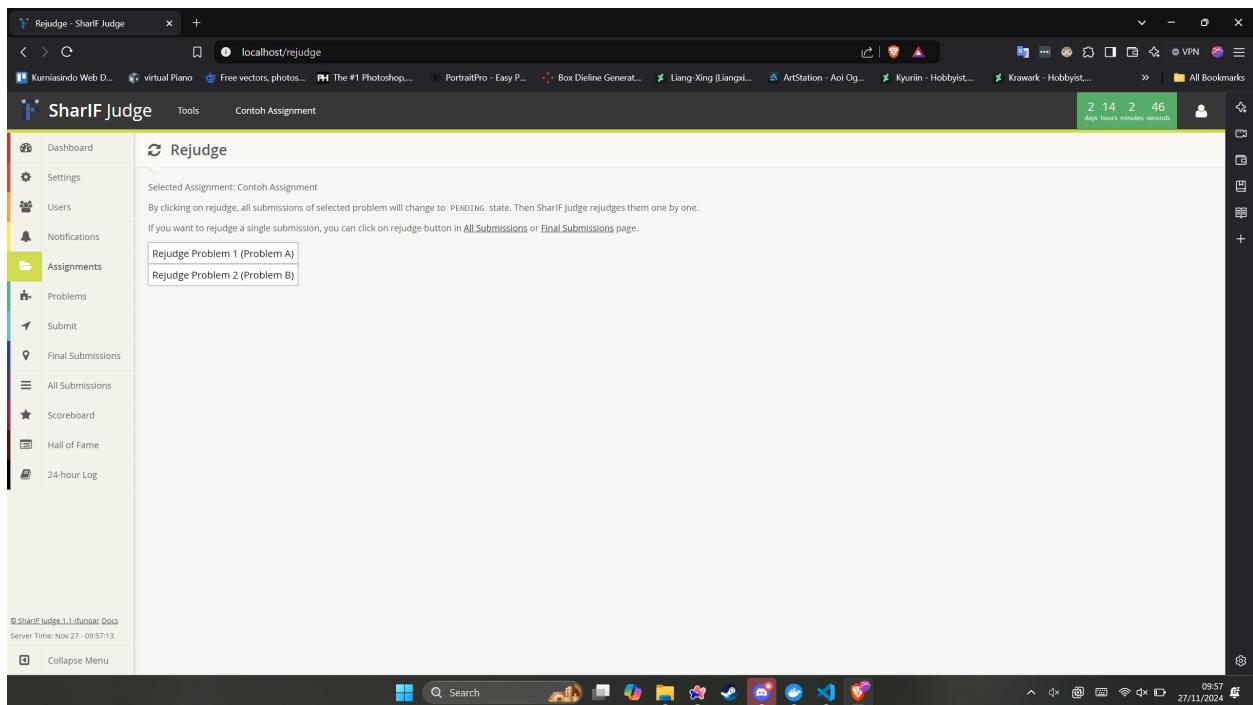
Berikut fungsi dengan penjelasannya pada *controller Profile.php*:

- `rejudge_single()`

Melakukan *rejudge* untuk satu buah *submission*.

- `index()`

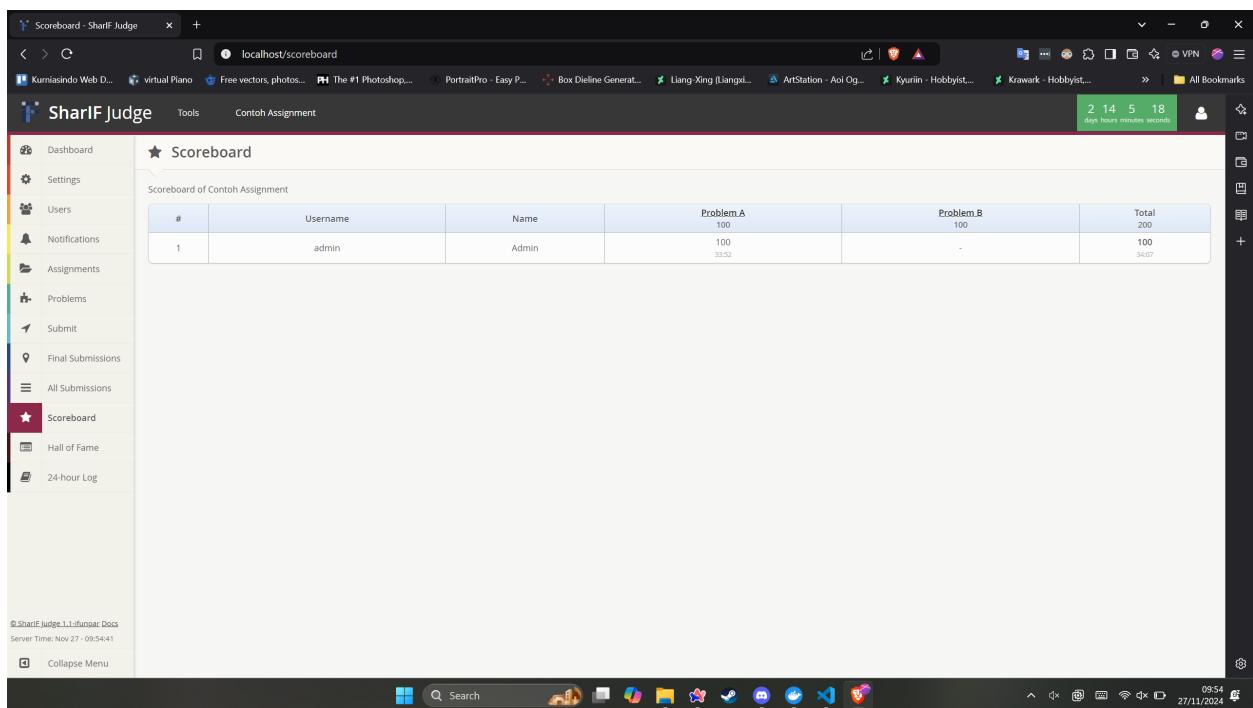
Mendapatkan data dan menampilkan *view rejudge.twig*. Fungsi ini juga dapat melakukan *rejudge* pada sebuah *problem* tertentu. Gambar 3.16 menunjukkan halaman Rejudge.



Gambar 3.16: Halaman Rejudge

- **Scoreboard.php**

Controller Queueprocess.php hanya memiliki satu fungsi yaitu `index()` yang akan menampilkan `view scoreboard.twig` dengan data dari `Scoreboard_model`. Gambar 3.17 menunjukkan hasil halaman Scoreboard.



Gambar 3.17: Halaman Scoreboard

- **Server_time.php**

Controller Queueprocess.php hanya memiliki satu fungsi yaitu `index()` yang akan mencetak

waktu pada *server*, waktu akan digunakan untuk sinkronisasi waktu.

- **Settings.php**

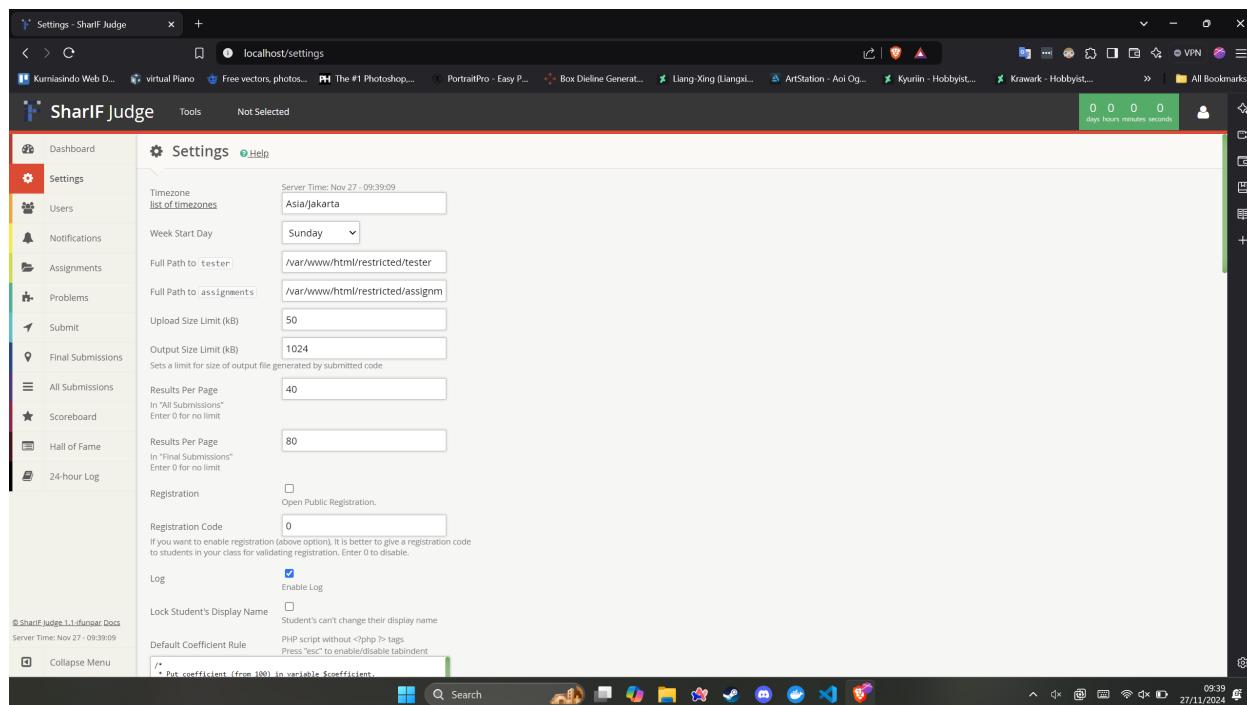
Berikut fungsi dengan penjelasannya pada *controller Settings.php*:

- `update()`

Memperbarui *settings* dari masukkan pengguna.

- `index()`

Mendapatkan data dari *Settings_model* dan menampilkan *view settings.twig*. Jika terdapat *error setting* pada sistem, akan ditampilkan juga pada *view* tersebut. Gambar 3.18 menunjukkan hasil halaman Users.



Gambar 3.18: Halaman Settings

- **Submissions.php**

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

- `_download_excel($view)`

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

- `final_excel()`

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

- `all_excel()`

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

- `select()`

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

- `_check_type($type)`

Melakukan validasi tipe *submission* yang dikumpulkan.

- `view_code()`

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

- `download_file()`

Mengunduh *file* kode sebuah *submission*.

- `the_final()`

Mendapatkan data dari `Submit_model` untuk mendapatkan *final submission* dan menampilkan halaman `submission.twig` berisi *final submission*. Gambar 3.19 menunjukkan halaman Final Submissions .

#	ID	Username	Name	Problem	Submit Time	Score			Language	Status	Code	Log	Actions
						Score	Delay %	Final Score					
1	3	admin	Admin	1	2024-11-27 09:52:35	100	No Delay 100%	100	Java	100	Code	Log	...
2	4	admin	Admin	2	2024-11-27 09:54:16	0	No Delay 100%	0	Java	Uploaded	Code	Log	...

Gambar 3.19: Halaman Final Submissions

- `all()`

Mendapatkan data dari `Submit_model` untuk mendapatkan seluruh *submission* dan menampilkan halaman `submission.twig` berisi semua *submission*. Gambar 3.20 menunjukkan halaman All Submissions .

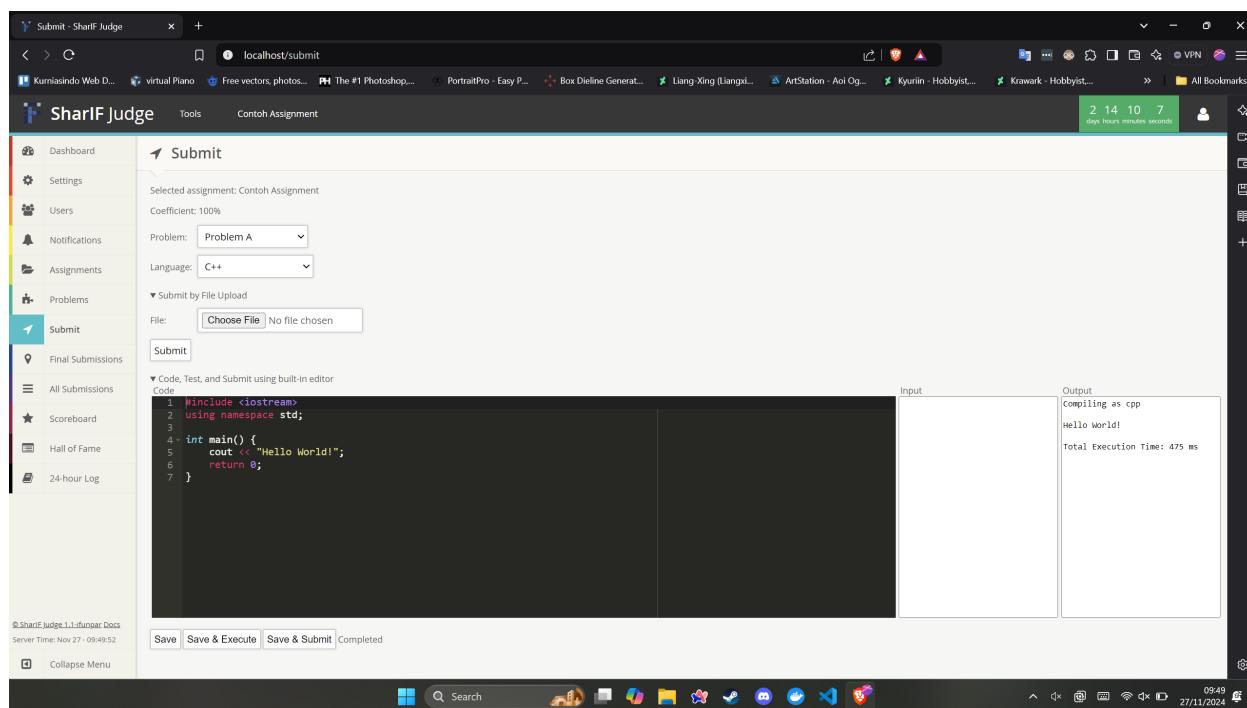
Final	ID	Username	Name	Problem	Submit Time	Score			Language	Status	Code	Log	Actions
						Score	Delay %	Final Score					
✓	4	admin	Admin	2	2024-11-27 09:54:16	0	No Delay 100%	0	Java	Uploaded	Code	Log	...
✓	3	admin	Admin	1	2024-11-27 09:52:35	100	No Delay 100%	100	Java	100	Code	Log	...
○	2	admin	Admin	1	2024-11-27 09:51:48	100	No Delay 100%	100	C++	100	Code	Log	...
○	1	admin	Admin	1	2024-11-27 09:51:33	0	No Delay 100%	0	C++	0	Code	Log	...

Gambar 3.20: Halaman All Submissions

- `Submit.php`

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

- `_language_to_type($language)`
Mengembalikan kode singkat dari `$language` dipilih.
- `_language_to_ext($language)`
Mengembalikan extensi file dari `$language` yang dipilih.
- `_match($type, $extension)`
Melakukan validasi untuk `$type` dan `$extension` agar sesuai.
- `_check_language($str)`
Melakukan validasi sudah dipilihannya bahasa.
- `_upload()`
Menyimpan jawaban pengguna yang dikirim dan menambahkannya ke dalam *queue* untuk dinilai jika bukan *upload only problem*.
- `load($problem_id)`
Mendapatkan isi file dan menaruh isi file ke editor kode.
- `save($type)`
Menyimpan isi editor kode ke dalam *server* dan menjalankan atau mengumpulkan jika diinginkan.
- `_submit($data, $problem_id, $language, $user_dir)`
Menambahkan kode ke dalam *submission* untuk dinilai.
- `_execute($data, $problem_id, $language, $user_dir)`
Menambahkan kode ke dalam *queue* untuk di jalankan saja.
- `get_output($problem_id)`
Mendapatkan keluaran dari kode yang telah dijalankan sebagai hasil eksekusi.
- `index()`
Mendapatkan data dari *model* `Assignments_model` untuk mendapatkan *problem* dan data lainnya. Semua data akan dimasukkan dalam *view* `submit.twig`. Gambar 3.21 menunjukkan hasil halaman Submit. Halaman ini terdapat editor kode yang sudah di implementasikan [7].

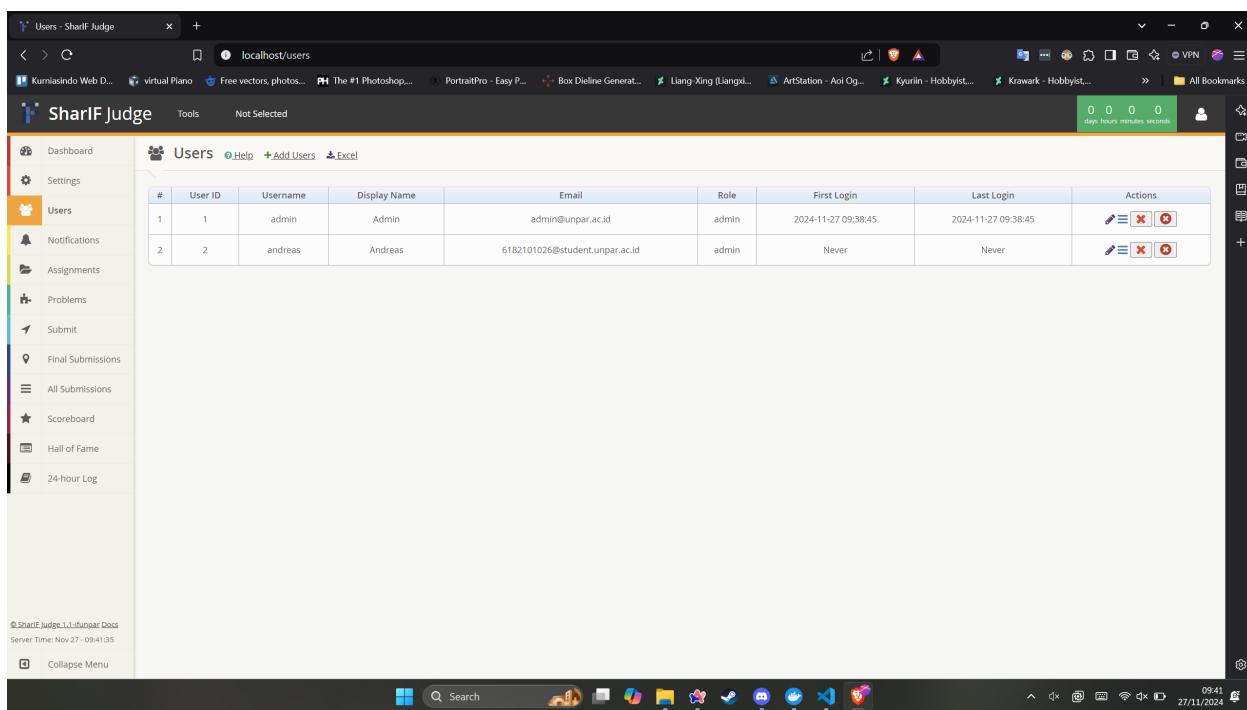


Gambar 3.21: Halaman Submit

- **User.php**

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

- **add()**
Menambahkan *user* baru ke dalam *sistem*.
- **delete()**
Menghapus sebuah *user*.
- **delete_submissions()**
Menghapus seluruh *submissions* dari sebuah *user*.
- **list_excel()**
Menggunakan *library PHPExcel* untuk membuat sebuah file excel dari seluruh daftar *user* yang akan diunduh pengguna.
- **index()**
Mendapatkan data dari *User_model* dan menunjukkan *view users.twig*. Gambar 3.22 menunjukkan hasil halaman Users. Pada halaman ini terdapat daftar seluruh *user* yang terdaftar pada SharIF Judge. Pengguna dapat membuat, memperbaharui, dan menghapus *user*.



Gambar 3.22: Halaman Users

3.1.2 Assets

Pada SharIF Judge terdapat direktori bernama *Assets* yang menjadi tempat untuk menyimpan seluruh kebutuhan dari sisi pengguna seperti gambar logo dan *library javascript*. Berikut merupakan isi dari direktori *assets* beserta dengan kegunaannya dalam aplikasi SharIF Judge:

- **Direktori ace**
Direktori ini berisikan hasil *build library Ace*, menghasilkan file *javascript* yang berfungsi untuk menambahkan editor kode pada aplikasi.
- **Direktori font**
Direktori ini berisikan font khusus dan juga memiliki *library javascript* bernama font-awesome yang digunakan untuk menaruh icon berformat svg dalam tampilan SharIF Judge.
- **Direktori fullcalendar**
Direktori ini berisikan hasil *build library FullCalendar*, menghasilkan file *javascript* dan *css*

yang berfungsi untuk memasukkan kalendar dalam tampilan SharIF Judge.

- Direktori **gridster**

Direktori ini berisikan hasil *build plugin* Gridster untuk *library* jQuery, menghasilkan file *javascript* dan *css* yang berfungsi untuk memasukkan *layout grid* yang dapat di ubah dengan menggunakan sifat mouse *drag and drop*.

- Direktori **images**

Direktori ini berisikan gambar kustom yang digunakan oleh SharIF Judge seperti logo dan banner.

- Direktori **js**

Direktori ini berisikan hasil *build library* jQuery, taboverride, moment, *plugins* jQuery berfungsi untuk membantu membangun *javascript* khusus yang dipakai dalam SharIF Judge. Direktori ini juga memiliki file *javascript* khusus yang dipakai dalam halaman SharIF Judge yaitu sebagai berikut:

- **shj_functions.js**

File *javascript shj_functions* digunakan untuk seluruh sistem umum dalam SharIF Judge seperti waktu server, sidebar toogle, loading dan berbagai macam fungsi yang digunakan dalam SharIF Judge.

- **shj_submissions.js**

File *javascript shj_submissions* digunakan untuk menangani berbagai fitur untuk halaman all submissions dan final submissions seperti menampilkan kode program dan memeriksa ulang hasil kode dalam judge.

- **shj_submit.js**

File *javascript shj_submit* digunakan untuk menangani berbagai fitur untuk halaman submit terutama fungsi aksi pada IDE yaitu aksi *save*, *execute*, *submit*, dan memuat kode lama ke dalam editor kode.

- Direktori **nano_scroller**

Direktori ini berisikan hasil *build plugin* nanoScrollerJS untuk *library* jQuery, menghasilkan file *javascript* dan *css* yang berfungsi untuk membangun *scrollbar* pada SharIF Judge.

- Direktori **noty**

Direktori ini berisikan hasil *build plugin* noty untuk *library* jQuery, menghasilkan file *javascript* dan *css* yang berfungsi untuk menampilkan pesan atau notifikasi kepada pengguna pada SharIF Judge.

- Direktori **pdfjs**

Direktori ini berisikan hasil *build library* pdfjs, menghasilkan file *javascript* yang berfungsi untuk menampilkan file pdf pada SharIF Judge.

- Direktori **reveal**

Direktori ini berisikan hasil *build plugin* Reveal yang sudah dimodifikasi untuk *library* jQuery, menghasilkan file *javascript* dan *css* yang berfungsi untuk menampilkan *modal* konfirmasi.

- Direktori **snippet**

Direktori ini berisikan hasil *build plugin* Snippet yang sudah dimodifikasi untuk *library* jQuery, berfungsi untuk sebagai *Syntax Highlighter* untuk teks kode dalam halaman Hall of Fame, Problems, dan Submissions.

- Direktori **styles**

Direktori ini berisikan css khusus yang digunakan untuk memperindah halaman dalam SharIF Judge.

- Direktori **tinymce**

Direktori ini berisikan hasil *build library* tinymce, berfungsi untuk memasukkan editor teks dalam SharIF Judge.

3.1.3 Penyimpanan Kode Submission

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

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

Penjelasan untuk format di atas adalah sebagai berikut:

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

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

```
assignment_5/p1/kenzhi/probA-6.java
```

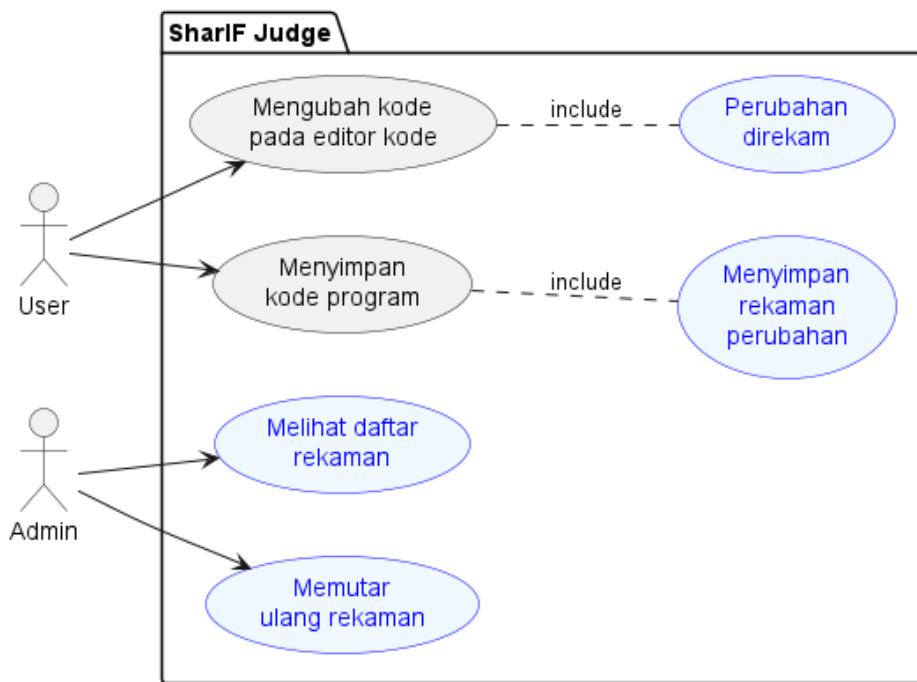
3.1.4 Antrean Penilaian Kode

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

1. *Controller Submit* akan menyimpan kode ke dalam file pada folder sesuai pada subbab 3.1.3.
2. *Controller Submit* akan memasukkan data *submission* ke dalam *model Queue_model*.
3. *Model Queue_model* akan menyimpan data *submission* pada *database submission* dan menambahkan data *queue*.
4. Selanjutnya *Controller Submit* akan memanggil fungsi *process_the_queue()* yang akan menjalankan fungsi *run()* pada *controller Queueprocess*.
5. *Controller Queueprocess* akan menjalankan **tester.sh** pada folder **tester** dengan data dari *queue*.
6. **tester.sh** akan menilai kode yang akan dibaca oleh *controller Queueprocess* yang akan menyimpan hasil penilaian.
7. Terakhir *Queueprocess* akan menyimpan hasil penilaian pada *database submission* dan menghapus data *queue* menggunakan *Queue_model*.

3.2 Analisis Sistem Usulan

Pembuatan sistem pemutaran ulang ketikan membutuhkan 4 fitur baru yaitu fitur perekaman perubahan, fitur penyimpanan rekaman perubahan, fitur melihat daftar rekaman yang ada, dan fitur untuk memutar ulang rekaman. Gambar 3.23 menunjukkan bagaimana fitur baru akan berinteraksi dengan sistem SharIF Judge dan *user*. Berikut merupakan penjelasan mengenai fitur-fitur yang akan ditambahkan pada SharIF Judge untuk membangun sistem perekaman ketikan.



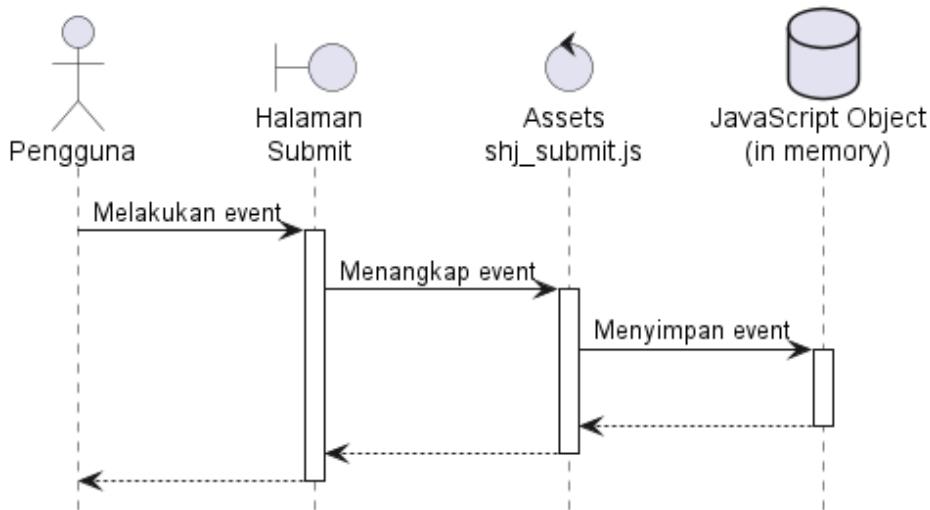
Gambar 3.23: Usecase analisis sistem usulan

3.2.1 Fitur perekaman perubahan atau event

Fitur perekaman perubahan pada editor kode bukan hanya perubahan text melainkan pada seluruh kejadian atau *event* yang terjadi pada editor kode seperti contohnya adalah perubahan posisi kursor maupun pilihan pada kode. Fitur perekaman perubahan akan otomatis oleh browser dengan bantuan *javascript* yang ada pada browser pengguna dan akan dijalankan saat sebuah *event* terjadi pada editor kode.

Sequence Diagram

Gambar 3.24 merupakan sebuah *sequence diagram* yang menunjukkan bagaimana fitur perekaman perubahan atau event akan berintegrasi dengan sistem IDE akan bekerja pada SharIF Judge.



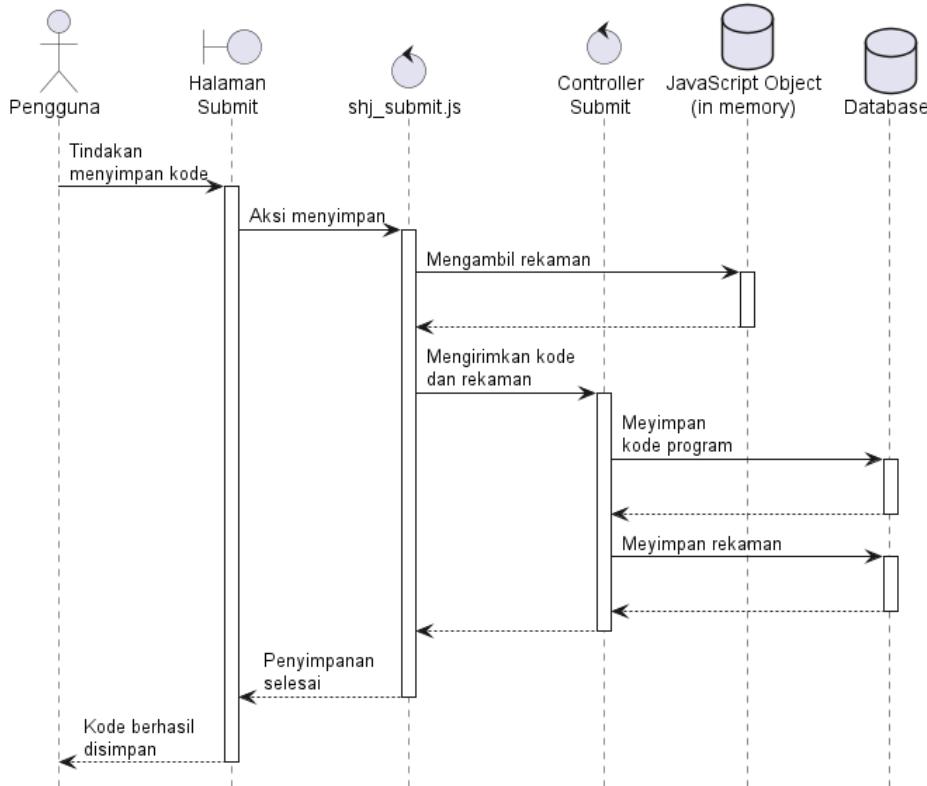
Gambar 3.24: Sequence Diagram Fitur Perekaman Perubahan

3.2.2 Fitur penyimpanan rekaman perubahan

Fitur penyimpanan rekaman perubahan akan dilakukan secara otomatis saat pengguna melakukan tindakan menyimpan kode program. Fitur penyimpanan rekaman akan berintegrasi dengan fitur penyimpanan kode program yang sudah ada. Pada fitur ini daftar rekaman akan diperbarui dengan adanya rekaman baru atau perubahan pada *file* rekaman.

Sequence Diagram

Gambar 3.25 merupakan sebuah *sequence diagram* yang menunjukkan bagaimana fitur penyimpanan rekaman perubahan akan berintegrasi dengan sistem IDE akan bekerja pada SharIF Judge.



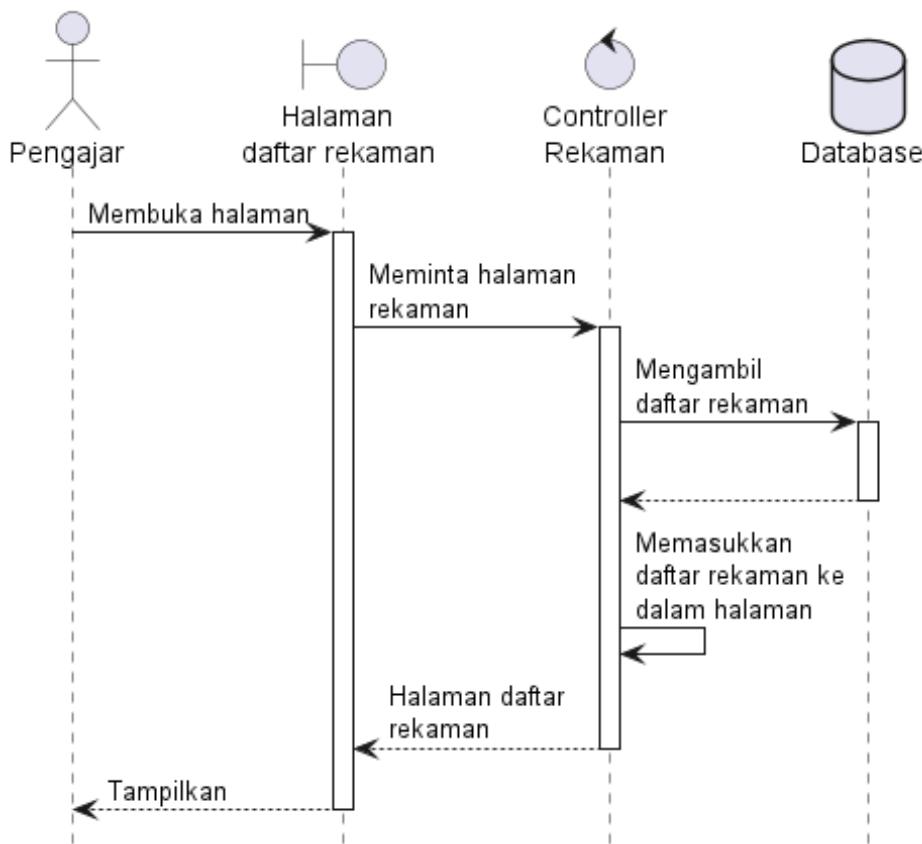
Gambar 3.25: Sequence Diagram Fitur Penyimpanan Rekaman

3.2.3 Fitur melihat daftar rekaman

Pada sistem pemutaran ulang ketikan dibutuhkannya sebuah halaman baru yang akan dinamakan halaman rekaman. Pada halaman ini akan dimunculkannya daftar rekaman untuk *assignment* yang dipilih pada halaman *Assignment*. Fitur ini akan dijalankan pada saat halaman rekaman dimuat ke dalam browser oleh SharIF Judge, dimana data yang dimasukkan ke dalam halaman rekaman adalah daftar rekaman tersebut dan beberapa data yang dibutuhkan.

Sequence Diagram

Gambar 3.26 merupakan *sequence diagram* yang menunjukkan bagaimana sistem akan bekerja saat user akan membuka halaman rekaman pada SharIF Judge.



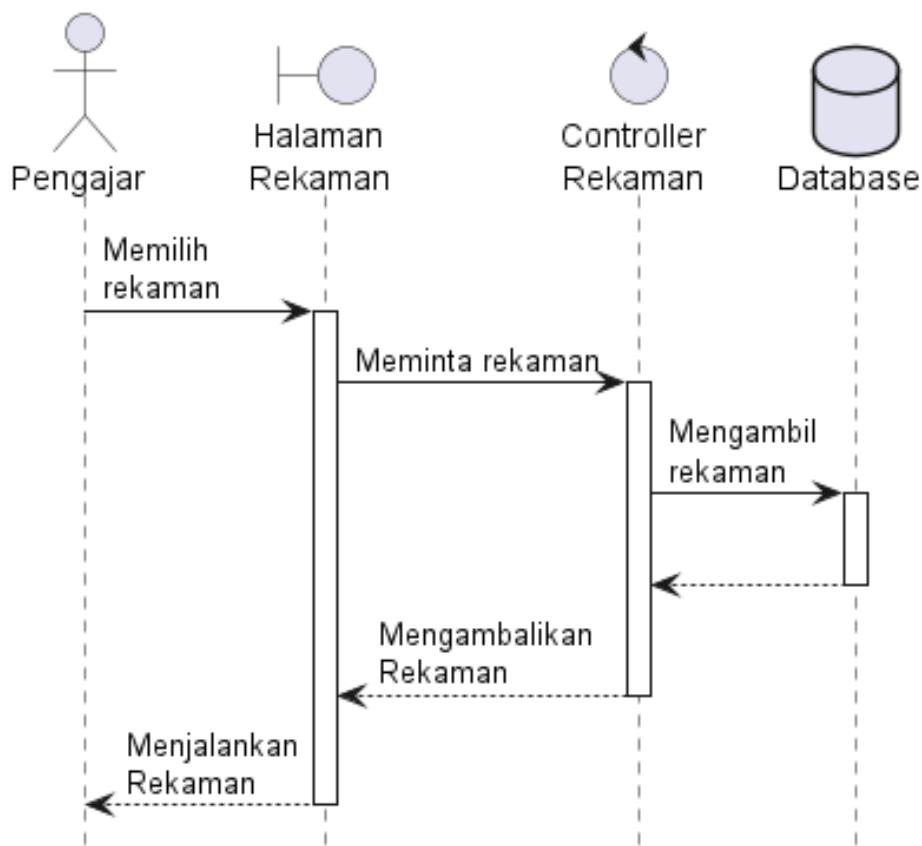
Gambar 3.26: Sequence Diagram Membuka Halaman Rekaman

3.2.4 Fitur pemutaran ulang rekaman

Fitur pemutaran ulang rekaman akan membuat satu buah rekaman dalam daftar rekaman dalam halaman rekaman dapat ditekan oleh pengguna untuk menandakan bahwa sebuah rekaman dipilih untuk dijalankan. Saat sebuah rekaman dipilih, browser akan meminta data rekaman kepada SharIF Judge dan dengan bantuan *javascript* akan melakukan pemutaran ulang rekaman pada sebuah editor kode yang tidak dapat diubah dalam halaman rekaman.

Sequence Diagram

Gambar 3.27 merupakan *sequence diagram* yang menunjukkan bagaimana sistem SharIF Judge bekerja dari pemilihan rekaman hingga pemutaran ulang rekaman akan terjadi.



Gambar 3.27: Sequence Diagram Membuka Halaman Rekaman

BAB 4

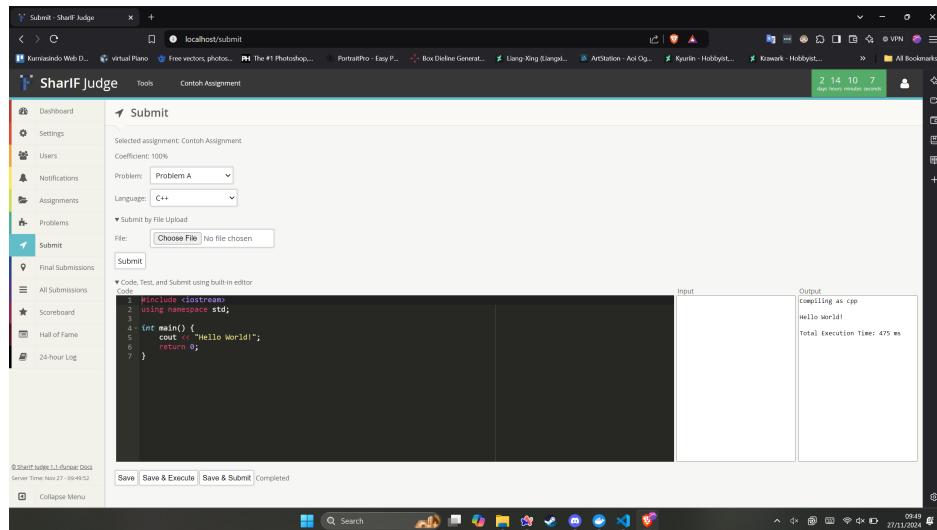
PERANCANGAN

Bab ini membahas tentang perancangan untuk implementasi sistem perekaman ulang dalam SharIF-Judge. Perancangan akan dilakukan

4.1 Rancangan Antarmuka

4.1.1 Sistem Rekaman

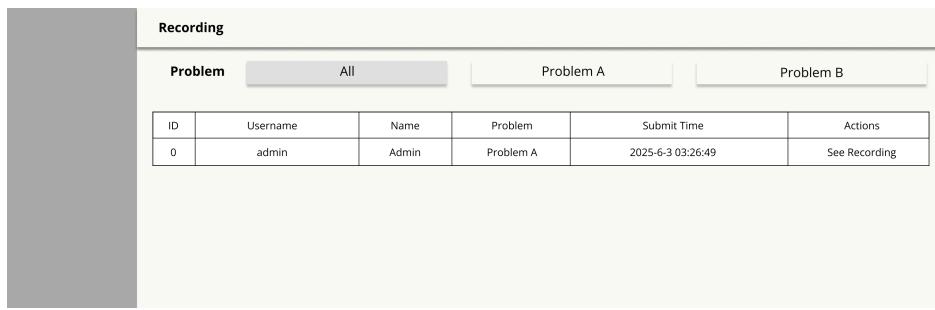
Seluruh sistem rekaman akan diimplementasikan dalam halaman Submit tidak memerlukan perubahan pada antarmuka. Gambar 4.1 menunjukkan halaman Submit dan IDE yang sudah diimplementasikan oleh Nicholas Aditya Halim [7].



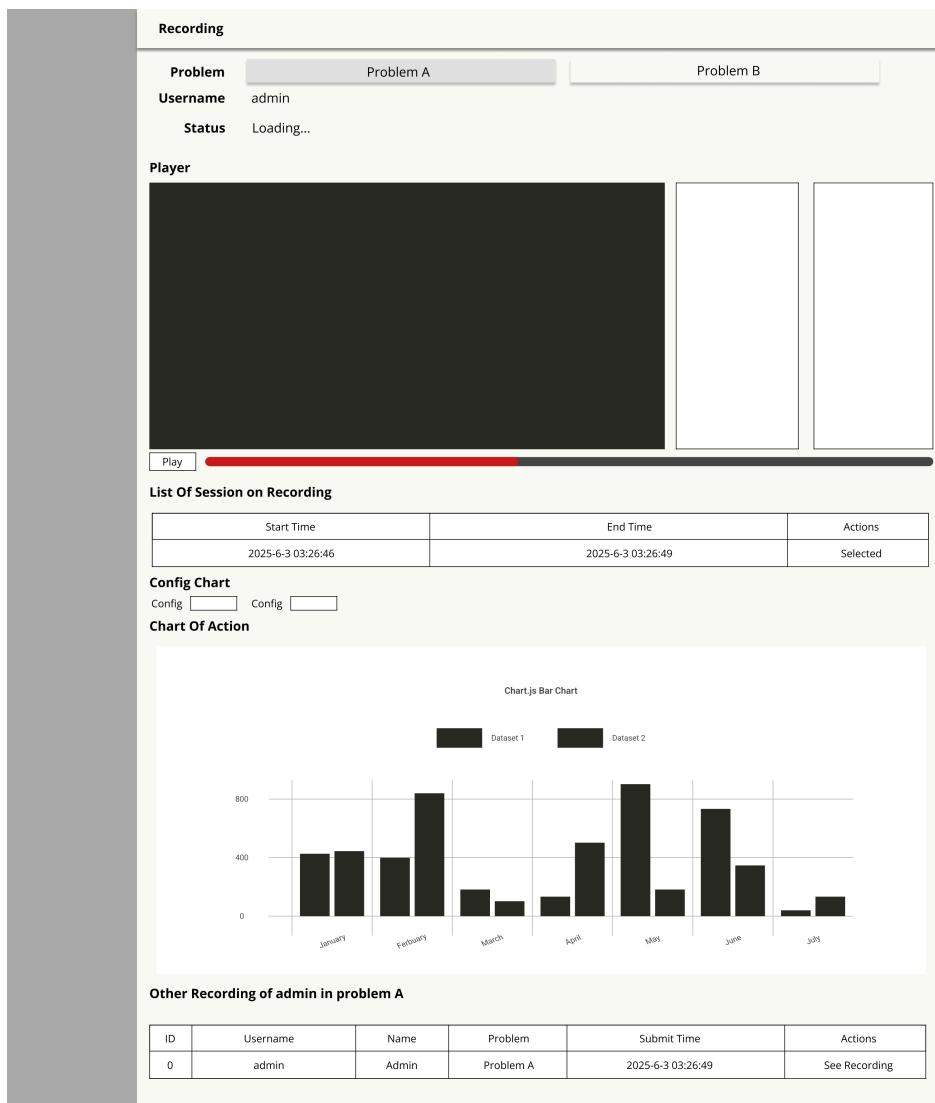
Gambar 4.1: Halaman

4.1.2 Sistem Pemutaran ulang

Pada sistem pemutaran ulang dibutuhkan dua halaman baru yaitu halaman untuk menunjukkan daftar rekaman dalam sistem dan halaman untuk sistem pemutaran ulang sebuah rekaman. Gambar 4.2 merupakan rancangan antarmuka untuk halaman daftar rekaman dan Gambar 4.3 merupakan rancangan antarmuka untuk halaman pemutaran ulang.



Gambar 4.2: Rancangan Antarmuka Halaman Daftar Rekaman



Gambar 4.3: Rancangan Antarmuka Halaman Pemutaran Ulang

4.2 Rancangan Penyimpanan Rekaman

Rekaman yang akan disimpan akan berupa sebuah daftar *event* yang terjadi. Dalam javascript, daftar tersebut akan menjadi sebuah *array* yang berisi *event-event* yang terjadi. Rekaman juga akan menyimpan waktu dimana rekaman dimulai, awal kode yang ada dalam editor kode dan juga awal posisi cursor dalam IDE. Maka dari itu, Rekaman akan menjadi sebuah *object* javascript yang

berisi sebagai berikut:

1. `timestart`: Waktu awal rekaman dimulai.
2. `start_value`: Isi awal dalam editor kode.
3. `start_cursor`: Posisi awal cursor dalam editor kode.
4. `events`: Daftar *event* yang terjadi.

Event yang akan direkam juga membutuhkan beberapa data yang harus disimpan yaitu: waktu *event* terjadi, *event* yang terjadi, dan muatan *event* yang terjadi. Maka *event* juga akan disimpan dalam bentuk *object javascript*. Berikut merupakan format sebuah *event*:

```
{time: <time>, event: <event>, payload: <payload>}
```

Berikut merupakan penjelasan tentang format penyimpanan perekaman.

- <time> akan menunjukkan pada milidetik berapa event terjadi setelah waktu awal perekaman dimulai. sedangkan <event> dan <payload> merupakan data event yang terjadi.
- <event> merupakan *event* yang terjadi pada waktu tersebut, pada contohnya adalah pengguna melakukan perubahan pada editor kode dengan menambahkan huruf ‘a’, maka *event* yang terjadi merupakan *insert*. Semua event yang ditangkap oleh sistem akan dijelaskan pada sub Bagian 4.3.1.
- <payload> merupakan muatan *event* yang terjadi. Muatan akan disesuaikan dengan *event* yang terjadi. Sebagai contoh untuk event *insert* di atas, maka isi dari *event* tersebut adalah huruf ‘a’, dan posisi cursor dalam editor kode dimana huruf tersebut dimasukkan.

Berikut contoh hasil untuk sebuah *event insert* pada penjelasan di atas:

```
{time: 1203, event: "insert", payload: {data: "a", start: [10, 9]}}
```

Penyimpanan rekaman juga akan disimpan pada folder yang sama dengan penyimpanan kode submission seperti yang dijelaskan pada Bagian 3.1.3 dengan nama file `record`.

4.3 Rancangan Perubahan Kode

Untuk mengimplementasikan fitur yang diusulkan pada Bagian 3.2, diperlukannya perubahan kode berikut ini pada SharIF-Judge.

4.3.1 Merekam perubahan atau event

Untuk menambahkan fitur ini, diperlukannya perubahan pada bagian javascript yaitu `assets/js/shj_submit.js` dalam halaman Submit. Dimana javascript tersebut akan menjalankan perekaman secara otomatis saat pengguna memilih *problem* yang ada dalam *assignment* yang dipilih. Berikut merupakan *event* yang akan ditangkap oleh *javascript* dalam halaman Submit:

- Perubahan isi kode pada editor kode.
- Perubahan posisi cursor pada editor kode.
- Perubahan fokus pada web page.
- Pergantian *tab* dalam browser.
- Perubahan fokus pada PDF Viewer.
- Perubahan isi pada editor *input* dalam IDE.
- Perubahan isi pada editor *output* dalam IDE.
- Aksi men-*Save*.
- Aksi men-*Save & Execute*.
- Aksi men-*Save & Submit*.

4.3.2 Menyimpan rekaman

Untuk setiap aksi menyimpan kode, menjalankan kode dengan tes kasus, dan mengumpulkan kode melalui IDE, rekaman juga akan disimpan ke dalam sistem.

Untuk menyimpan rekaman, perlu dilakukan perubahan sebagai berikut:

- *Controller* Submit:
 - Fungsi `save($type)`:
Fungsi ini akan diubah agar dapat menangani data rekaman yang dikirim oleh *user*. Data tersebut akan disimpan dalam folder yang sama dengan kode program.
 - Fungsi `_submit($data, $problem_id, $language, $user_dir)`:
Fungsi ini akan diubah agar dapat menangani data rekaman yang dikirim oleh fungsi `save($type)` dengan menambahkan parameter `$rec` berisi rekaman oleh *user*. Untuk setiap submit rekaman dari *save-save* sebelumnya akan diubah menjadi rekaman untuk *submit* tersebut.
- *Assets shj_submit.js*:
Menambahakan data rekaman yang dimuat ke dalam fungsi aksi menyimpan kode, menjalankan kode dengan tes kasus, dan mengumpulkan kode melalui IDE, rekaman juga akan disimpan ke dalam sistem.

4.3.3 Melihat daftar rekaman

Untuk melihat semua daftar rekaman yang terjadi, maka dibutuhkannya database untuk menyimpan daftar dan mendapatkan daftar rekaman yang sudah disimpan dalam sistem. Setelah itu dibutuhkannya juga halaman baru dalam SharIF-Judge, maka perubahan *Controller*, *Model*, dan *View* dalam SharIF-Judge.

Dikarenakan itu, perlu dilakukan perubahan kode sebagai berikut:

- *Controller* Submit:
 - Fungsi `save($type)`:
Fungsi ini akan menambahkan *metadata* rekaman *user* ke dalam *database*. *Metadata* yang dimaksud adalah *id problem*, *id assignment*, dan *user* rekaman ini direkam. Metadata dalam database digunakan untuk mendapatkan daftar rekaman yang belum disubmit pada sebuah *problem* dalam *assignment* beserta dengan nama *user* rekaman.
 - Fungsi `_submit($data, $problem_id, $language, $user_dir)`:
Fungsi ini akan menambahkan *metadata* ke dalam *database* sebagai daftar rekaman yang sudah di submit. *Metadata* yang dimaksud adalah *id problem*, *id assignment*, dan *user* rekaman ini direkam. Metadata dalam database digunakan untuk mendapatkan daftar rekaman yang sudah disubmit pada sebuah *problem* dalam *assignment* beserta dengan nama *user* rekaman.
- *Controller* Recording:
Sebuah *controller* baru yang menangani segala hal mengenai sistem pemutaran ulang dalam SharIF-Judge. Fungsi yang dibutuhkan agar fitur ini berjalan hanyalah fitur untuk menunjukkan daftar rekaman dalam sistem dengan mengambil data dari *model* Recording dan menaruhnya dalam *view* recording. Setelah itu, *controller* akan menunjukkan *view* tersebut kepada *user* yang sudah login dan memiliki akses *instructor* atau lebih tinggi.
- *Model* Recording:
Sebuah *model* baru yang menangani segala hal mengenai penyimpanan dan pengambilan data rekaman dalam *database*. Fungsi-fungsi yang direncanakan dalam *model* adalah sebagai berikut:
 - Fungsi `get_recording()`:
Fungsi ini digunakan untuk mendapatkan seluruh daftar rekaman dalam database, fungsi ini juga dapat menyaring daftar rekaman berdasarkan *assignment*, *problem*, dan *user*.
 - Fungsi `add_recording()`:

Fungsi ini digunakan untuk menaruh sebuah rekaman ke dalam database.

- *View Recording_list:*

Sebuah *view* baru yang menampilkan daftar rekaman yang ada dalam sistem. *View* ini akan digunakan oleh *Controller Recording*.

4.3.4 Pemutaran ulang rekaman

Untuk dapat memutar ulang rekaman diperlukan beberapa perubahan kode dalam SharIF-Judge. Berikut merupakan rencana perubahan kode dalam SharIF-Judge:

- *Controller Recording:*

Sebuah *controller* baru yang menangani segala hal mengenai sistem pemutaran ulang dalam SharIF-Judge. Untuk fitur pemutaran ulang rekaman, diperlukan dua fungsi pada *controller* yaitu sebagai berikut:

- Fungsi `index()`:

Fungsi ini digunakan untuk menunjukkan *view Recording* kepada *user*.

- Fungsi `download_record()`:

Fungsi ini digunakan untuk mendapatkan file rekaman dalam sistem. Fungsi ini akan dipanggil menggunakan AJAX dalam *assets Recording.js*.

- *Model Recording:*

Sebuah *model* baru yang menangani segala hal mengenai penyimpanan dan pengambilan data rekaman dalam *database*. Fungsi yang dibutuhkan oleh fitur pemutaran ulang rekaman adalah fungsi `get_recording()` untuk mendapatkan seluruh daftar rekaman sebuah *user* dalam database berdasarkan *problem* dan *assignment*.

- *View Recording:*

Sebuah *view* baru yang menampilkan rekaman sebuah *user* yang ada dalam sistem. *View* ini akan digunakan oleh *Controller Recording*.

- *Assets Recording.js:*

Sebuah *assets javascript* yang digunakan oleh *view Recording* sebagai *script* yang akan dijalankan oleh *browser user*.

BAB 5

IMPLEMENTASI DAN PENGUJIAN

Bab ini membahas mengenai implementasi dan pengujian sistem perekaman ulang dalam SharIF-Judge.

5.1 Implementasi

Bagian ini menjelaskan hasil implementasi sistem pemutaran ulang pada SharIF-Judge berdasarkan perancangan pada Bab 3. Pada saat implementasi juga dilakukan penyesuaian pada perancangan yang sudah dibuat untuk mengatasi kendala yang dialami pada saat implementasi.

5.1.1 Merekam Peristiwa pada IDE

Fitur merekam peristiwa pada editor kode diimplementasikan untuk menangkap seluruh interaksi pengguna terhadap IDE dan SharIF-Judge pada saat pengguna menyelesaikan sebuah masalah dalam *assignment*. Data yang direkam akan digunakan untuk memutar ulang penyelesaian yang dilakukan oleh pengguna. Fitur ini diimplementasikan dengan memanfaatkan *Library Ace*, *event hooks* pada javascript. Implementasi ini akan memerlukan penyesuaian pada bagian *javascript* yaitu file `assets/js/shj_submit.js` yang dimuat pada halaman *Submit*.

Berikut merupakan alur sistem perekaman peristiwa:

1. Inisialisasi Perekam

Pada alur ini, semua perekaman akan diinisialisasi dengan menjalankan sebuah fungsi dinamakan `recordStart`. Fungsi ini akan memanggil seluruh *event hooks* dan *event listener* dalam *Library ace* agar dinyalakan.

Dalam menjalankan sebuah fungsi *event listener* dibutuhkannya dua argumen yaitu event yang akan dipanggil dan sebuah *callback function* yang akan dipanggil pada saat terjadinya sebuah event tersebut.

Dalam implementasi akan dibuat 2 buah *object javascript* yang menjadi fungsi *event listener* yaitu *object* pemanggilan *event listener* dan *object* menyimpan *callback function* yang dipanggil oleh *event listener* masing-masing. Kode 5.1 merupakan *object callback function* yang diimplementasikan.

Kode 5.1: *object callback function*

```
1 const handlers = {
2   editor_change: (e) =>
3     recordEvent(e.action, {
4       data: e.lines,
5       start: e.start,
6       end: e.end,
7     }),
8 }
```

Kode 5.2 merupakan *object* pemanggilan *event listener* yang dipanggil pada saat inisialisasi dan mendapatkan fungsi *callback function* dari *object handlers* pada Kode 5.1.

Kode 5.2: *object event listener*

```
1 const addListener = {
2   editor_change: () => editor.session.on("change", handlers.editor_change),
```

3 }

Setelah itu *object addListener* akan dipanggil oleh fungsi **recordStart**. Kode 5.3 menunjukkan fungsi yang dipanggil saat inisialisasi.

Kode 5.3: Beberapa *event listener* yang dipanggil

```
1 addListener.editor_change();
```

Fungsi **recordStart** akan dilakukan pada saat pengguna mengubah *problem* yang dipilih dalam halaman Submit.

2. Penyimpanan sebuah rekaman

Untuk setiap perekaman yang dibutuhkan, dijalankan sebuah fungsi yang mendekripsi perubahan tersebut dan menjalankan sebuah *callback function* saat terjadi perubahan tersebut. Fungsi ini dipanggil pada saat inisialisasi. *Callback function* tersebut akan mendapatkan argumen sesuai dengan perubahan yang didekripsi. Pada contohnya untuk perubahan teks pada editor kode yaitu fungsi **onchange**, argumen yang diberikan merupakan teks yang dimasukan yaitu contohnya adalah ‘A’, dan juga posisi dimana teks tersebut dimasukkan dalam editor kode. Kode 5.4 merupakan contoh argumen yang diberikan.

Kode 5.4: Contoh argumen yang diberikan oleh fungsi **onchange**

```
1 {
2     data: ["A"],
3     start: {row: 0, column: 1},
4     end: {row: 0, column: 2}
5 }
```

Setelah itu data akan disimpan dalam sebuah *object javascript* seperti yang sudah dijelaskan pada Bab 4.2. data argumen akan disimpan menggunakan key ‘args’ atau ‘payload’.

3. Penyimpanan data rekaman

Selanjutnya sebuah *event* atau rekaman yang sudah dicatat dan menjadi sebuah *object javascript* bernama **recording**. seluruh event rekaman akan simpan dalam sebuah *array* dalam **recording** dengan key ‘events’ seperti yang sudah dijelaskan pada Bab 4.2. **recording** juga memiliki waktu dimulainya rekaman, isi awal editor kode, posisi awal cursor dalam editor kode. **recording** juga memiliki fungsi **init** untuk meninisialisasi seluruh *object recording*.

Kode 5.5: Contoh argumen yang diberikan oleh fungsi **onchange**

```
1 const recording = {
2     events: [],
3     startTime: -1,
4     startValue: "",
5     startSelection: [],
6
7     init: () => {
8         recording.events = [];
9         recording.startTime = Date.now();
10        recording.startValue = editor.getValue();
11        recording.startSelection = getSelection(editor);
12    },
13};
```

Kode 5.5 merupakan **recording** pada saat keadaan kosong. Pada *object recording*, **events** merupakan sebuah array dengan isi sebuah rekaman event, **startTime** merupakan waktu awal rekaman dimulai, **startValue** merupakan isi awal dalam editor kode, dan **startSelection** merupakan posisi awal cursor dalam editor kode.

Berikut merupakan peristiwa yang akan direkam oleh sistem perekaman:

- **editor.change**: Peristiwa ini akan menangkap perubahan isi teks dalam editor kode.
- **editor.changeCursor**: Peristiwa ini akan menangkap perubahan cursor dalam editor kode.
- **editor.changeSelection**: Peristiwa ini akan menangkap perubahan selection cursor dalam editor kode.
- **window.focus**: Peristiwa ini akan menangkap pengguna pada saat pengguna *focus* pada SharIF Judge web page.
- **window.blur**: Peristiwa ini akan menangkap pengguna pada saat pengguna tidak *focus* pada SharIF Judge web page atau *focus* pada aplikasi lain atau web page lain.

- `window.visibilitychange`: Peristiwa ini akan menangkap pengguna yang mengubah *tab* dari SharIF Judge ke *tab* lain dalam browser.
 - `pdf_viewer.focusin`: Peristiwa ini akan menangkap pengguna pada saat pengguna *focus* pada PDF Viewer dalam IDE.
 - `pdf_viewer.focusout`: Peristiwa ini akan menangkap pengguna pada saat pengguna tidak *focus* pada PDF Viewer dalam IDE.
 - `editor_input.input`: Peristiwa ini akan menangkap perubahan isi editor input dalam IDE.
 - `editor_output.change`: Peristiwa ini akan menangkap hasil output saat terjadinya aksi *Save & Execute* dalam IDE.
 - `save.click`: Peristiwa ini akan menangkap aksi *Save* yang dilakukan pengguna.
 - `execute.click`: Peristiwa ini akan menangkap aksi *Save & Execute* yang dilakukan pengguna.
 - `submit.click`: Peristiwa ini akan menangkap aksi *Save & Submit* yang dilakukan pengguna.
- Seluruh alur sistem perekaman peristiwa dalam SharIF-Judge akan ditambahkan ke dalam file `assets/js/shj_submit.js`. Kode perubahan terdapat pada Lampiran A.

Perbaikan Implementasi

Pada saat mengujian fungsi penyimpanan rekaman, dalam tahap alur penyimpanan data rekaman dan juga bagaimana inisialisasi perekaman akan diubah dari perancangan Bab 4.3.1. Hal ini dikarenakan saat pengguna memilih ulang masalah atau memuat ulang halaman Submit, maka semua events yang sudah direkam akan hilang. Maka dari itu berikut merupakan perubahan pada alur fitur sistem perekaman ketikan:

1. Inisialisasi Perekam

Alur inisialisasi akan diubah agar dapat memuat events yang sebelumnya sudah disimpan dalam *object javascript* bernama `befRecording`. Oleh karena itu, dibutuhkannya penambahan fungsi pada `Controller Submit.php` yaitu fungsi untuk mengambil data rekaman sebelumnya bernama `load_rec` yang mengambil argumen pengenal masalah yang dipilih oleh pengguna. Kode penambahan terdapat di Lampiran A

2. Penyimpanan data rekaman

Pada `object recording`, `startValue` dan `startSelection` tidak dibutuhkan karena isi awal dari editor kode dan juga posisi awal cursor dalam editor kode akan selalu berisi dengan nilai kosong yaitu teks kosong dan posisi di baris dan kolom pertama.

5.1.2 Menyimpan Rekaman pada Sistem

Fitur penyimpanan rekaman pada sistem bertujuan untuk menyimpan data secara permanen ke dalam database dan *server* agar dapat diputar kembali di lain waktu. Berikut merupakan alur fitur penyimpanan rekaman pada sistem:

1. Mengirimkan Data ke *Server*

Dalam halaman Submit, pengguna memiliki 3 aksi penting dalam IDE SharIF-Judge yaitu: `save`, `execute`, dan `submit`. Alur ini akan mengirimkan data rekaman pada saat pengguna melakukan aksi tersebut. Data yang dikirim merupakan *object recording* yang sudah jadikan sebagai teks JSON dengan menggunakan fungsi `JSON.stringify`.

2. Menyimpan Data Dalam File Sistem

File akan disimpan dalam *folder* yang sama dengan penyimpanan kode *submission* yang dijelaskan pada Bab 3.1.3. File akan diisi secara langsung oleh data rekaman dan tidak diubah oleh *server*. Penamaan file dapat dibagi berdasarkan aksi yang membuat pengguna mengirimkan data rekaman. Untuk aksi `save` dan `execute`, file dengan data rekaman akan disimpan dengan nama `recording`. Untuk aksi `submit`, file akan disimpan dengan nama `recording` dilanjutkan dengan sebuah '-' dan `submit id` yang dibuat. File tersebut akan memiliki tipe data yang sama yaitu JSON dikarenakan itu extensi file yang digunakan adalah `.json`.

3. Menyimpan Data Dalam Database

Saat penyimpanan data ke dalam file sistem berhasil, maka penyimpanan kedalam database juga akan dilakukan. Data yang akan disimpan kedalam database akan digunakan untuk mendaftar rekaman yang ada dalam sistem, maka data yang akan disimpan bukan data rekaman melainkan data statistik. Berikut merupakan data yang akan disimpan ke dalam Database:

- **rec_id**: pengenal rekaman yang sama dengan *submit id*.
- **username**: nama pengguna yang mengirimkan data rekaman.
- **problem_id**: pengenal masalah yang pengguna kerjakan.
- **assignment_id**: pengenal tugas yang pengguna kerjakan.
- **upload_at**: waktu sistem menyimpan data rekaman.

Untuk membuat databasenya sendiri, dibutuhkan penambahan tabel bernama tabel **recording** yang memiliki lima atribut diatas. Kode 5.6 menunjukkan pembuatan tabel baru menggunakan *CodeIgniter* dalam SharIF-Judge.

Kode 5.6: Kode membuat database pada SharIF-Judge

```

1 // create table 'recording'
2 $fields = array(
3     'rec_id'      => array('type' => 'INT', 'constraint' => 11, 'unsigned' => TRUE),
4     'upload_at'   => array('type' => DATETIME),
5     'assignment'  => array('type' => 'SMALLINT', 'constraint' => 4, 'unsigned' => TRUE),
6     'problem'     => array('type' => 'SMALLINT', 'constraint' => 4, 'unsigned' => TRUE),
7     'username'    => array('type' => 'VARCHAR', 'constraint' => 20),
8 );
9 $this->dbforge->add_field($fields);
10 if (! $this->dbforge->create_table('recording', TRUE))
11     show_error("Error creating database table " . $this->db->dbprefix('recording'));
12 // ADD Unique constraint
13 $this->db->query(
14     "ALTER TABLE {$this->db->dbprefix('recording')}
15         ADD CONSTRAINT {$this->db->dbprefix('srup_unique')} UNIQUE (rec_id, username, assignment, problem);"
16 );

```

Mengikuti arsitektural *CodeIgniter*, untuk menambahkan sebuah data ke dalam database perlu menggunakan sebuah *Model*. Oleh karena itu, dibutuhkannya *model* baru bernama **Recording_model.php** yang ditambahkan fungsi **add_recording()** yang memiliki argumen yaitu seluruh data yang ingin disimpan dalam database.

Untuk aksi *save* dan *execute* dimana tidak adanya *submit id* maka akan dibuat menjadi angka nol ('0') pada pengenal rekamannya atau **rec_id** jika aksinya merupakan *submit*.

Untuk alur pengiriman data ke *server*, dibutuhkannya penambahan kode ke dalam **assets/js/shj_submit.js**. Agar dapat menyimpan dibutuhkannya perubahan dalam kode *Controller Submit* pada fungsi **save(\$type)** dan fungsi **_submit()**. Kode pembahannya terdapat pada Lampiran A.

Perbaikan Implementasi

Pada saat pengujian yang sama dengan Bab 5.1.1, dibutuhkannya perubahan pada alur pengiriman data ke server. Dikarenakan *events* yang sudah di save dapat terhapus karena pengguna memilih ulang masalah dan memuat ulang halaman Submit, yang membuat rekaman inisialisasi dan menghapus rekaman lama. Maka dari itu pada saat mengirimkan data **recording**, akan disertakan juga data **befRecording** yang sudah diambil pada saat inisialisasi. Format pengiriman data juga akan berubah dikarenakan adanya rekaman yang lama menjadi sebuah *key* dan *value* karena hanya dua *value* yang harus disimpan yaitu *events* sebagai *value* dan *startTime* sebagai *key*. Maka format ini menjadi format keseluruhan events yang terjadi dan dapat disatukan dengan format yang sama menggunakan *spread operator* agar seluruh rekaman lama digabungkan. Kode 5.7 merupakan kode untuk mengirimkan teks JSON dengan mengabungkan kedua rekaman menggunakan *spread operator*.

Kode 5.7: *object callback function*

```

1 JSON.stringify({
2     ...befRecording,
3     [recording.startTime]: recording.events
4 }),
```

5.1.3 Melihat Daftar Rekaman

Fitur ini digunakan untuk melihat daftar rekaman mahasiswa yang tersimpan dalam sistem, pengguna juga dapat melihat isi rekaman yang terdapat dalam daftar rekaman tersebut. Fitur ini dibutuhkan dua tahap untuk diimplementasikan yaitu sebagai berikut:

- Pengambilan Data Rekaman

Fitur pengambilan data rekaman digunakan agar bagian depan SharIF-Judge dapat meminta daftar rekaman yang ada pada bagian belakang SharIF-Judge. Oleh karena ini merupakan sebuah halaman baru dalam SharIF-Judge, maka dibutuhkannya sebuah *Controller* baru bernama *Recording.php* yang menampilkan sebuah halaman baru yang dapat diakses melalui rute */recording/all/* yang menggunakan fungsi baru dalam *Recording_model.php* yaitu fungsi untuk mendapatkan daftar rekaman dinamakan *all_user_recordings*. Kode pertambahan pada *Model Recording.php* berada pada Lampiran A.

Berikut fungsi yang akan diimplementasikan dalam *Controller Recording.php*:

1. *__construct()*

Fungsi ini akan memuat seluruh kebutuhan *Model* dan *Helper* ke dalam *Recording.php*, fungsi *construct* juga akan membatasi akses oleh pengguna dibawah *instructor*. Fungsi ini juga mendapatkan *params url* yang dikirim oleh pengguna.

2. *all()*

Fungsi ini akan mengambil beberapa data yang dibutuhkan oleh antarmuka SharIF-Judge yaitu *assignment* yang dipilih oleh pengguna menggunakan *assignment_model*, *problem* yang ada dalam *assignment* tersebut menggunakan *assignment_model*, dan daftar *recording* yang tersimpan dalam sistem menggunakan *recording_model*. Setelah itu, server akan menempatkan seluruh data tersebut ke dalam *view* baru bernama *recording_list.twig*.

Seluruh alur untuk mengimplementasikan fitur pengambilan data rekaman dalam *Controller Recording.php* terdapat dalam Lampiran A.

- Antarmuka dan Tampilan Data

Antarmuka yang akan dibuat serupa dengan perancangan pada Bab 4.1.2 yang diimplementasikan ke dalam SharIF-Judge. Data yang dikirim oleh *Controller Recording.php* juga dapat ditampilkan menggunakan *Library twig* tanpa menbutuhkan *javascript* maupun *php* dalam antarmukanya. Gambar B.1 menunjukkan implementasi antarmuka beserta data yang terdapat dalam sistem. Untuk kode keseluruhan antarmuka menggunakan *Library twig* dapat dilihat pada Lampiran A.

Perbaikan Implementasi

Untuk fitur ini, tidak dilakukan perubahan pada implementasi yang sudah ada karena dianggap sudah memenuhi kebutuhan pengguna.

5.1.4 Pemutaran Ulang Rekaman

Fungsi pemutaran ulang rekaman menggunakan data rekaman yang sudah disimpan dalam sistem untuk menvisualisasikan proses penyelesaian masalah pengguna secara kronologis. Fitur pemutaran ulang ini membutuhkan yaitu sebagai berikut:

- Implementasi Antarmuka

Untuk menambahkan sebuah halaman baru dalam SharIF-Judge dibutuhkannya juga fungsi baru pada *Controller Recording.php*. Fungsi baru akan dinamakan *index* untuk menampilkan sebuah *view* baru bernama *recording.twig*. Fungsi *index* akan menampilkan halaman baru itu melalui rute */recording/*. Penamaan *index* itu agar rute tidak memerlukan */index* pada akhir rute karena jika rute *function* (Bab 2.2.2) akan otomatis mengarah pada fungsi *index* dalam kelas tersebut.

Fungsi `index` akan mengirim data daftar rekaman pengguna lainnya dalam masalah yang dipilih. Data tersebut akan dipakai oleh `recording.twig` untuk menambahkan daftar rekaman pengguna lainnya pada `assignment` dan `problem` yang sama.

Gambar B.2 merupakan antarmuka yang diimplementasikan serupa dengan perancangan pada Bab 4.1.2.

- Implementasi Memuat Data Rekaman

Data rekaman yang akan diambil sudah disimpan (Bab 5.1.2) dalam sistem menggunakan *Controller*. Tetapi data rekaman tidak akan dikirim oleh *Controller* pada saat halaman *recording* dimuat, melainkan menggunakan AJAX pada halaman *recording*. Maka dari itu, butuh fungsi baru pada *Controller Recording.php* dan sebuah assets *javascript* baru bernama `shj_function.js`.

Fungsi baru dalam *Controller Recording.php* akan dinamakan `download_record` yang memiliki argumen `assignment_id`, `problem_id`, dan `rec_id`. Fungsi tersebut akan mengambil file rekaman dalam file sistem dengan menggunakan argumen untuk mendapatkan lokasi dan nama file rekaman (Bab 4.2) SharIF-Judge dan mengirimkan file tersebut secara langsung. File juga akan dirikim dengan header `Content-Type: application/json` dan `Content-Disposition: attachment; filename= "rec.json"`.

Fungsi `download_record` akan dipanggil pada saat halaman *recording* dimuat oleh *javascript shj_function.js* menggunakan fungsi baru yaitu `getRecording`. Fungsi `getRecording` akan meninisialisasi editor kode dalam antarmuka dan menformat data rekaman agar lebih mudah untuk di putar ulang. Berikut merupakan format data tambahan yang akan diubah oleh fungsi `getRecording`:

- `events`: Data rekaman
- `eventsIndex`: sebuah map dengan *key* waktu saat sebuah `events` terjadi dan *value* `index` waktu saat sebuah `events` terjadi.
- `indexEvents`: sebuah map dengan *key* `index` waktu saat sebuah `events` terjadi dan *value* waktu saat sebuah `events` terjadi.
- `presumIndexDuration`: menkalkulasikan panjang rekaman sebelum rekaman selesai.
- `length`: panjang data rekaman
- `duration`: durasi dari seluruh rekaman yang ada pada data rekaman

Data tersebut akan disimpan dalam sebuah *object javascript* yang dinamakan `recording` dan akan dipakai pada saat menjalankan rekaman dan untuk menampilkan histogram `events` yang terjadi.

- Implementasi Menjalankan Rekaman

Fitur menjalankan rekaman akan menggunakan data bedasarkan data yang didapatkan oleh AJAX yang dijelaskan pada bagian 5.1.4. Fitur menjalankan rekaman akan membutuhkan penambahan kode pada *javascript shj_function.js* yang akan menjalankan fungsi `play` atau `stop` untuk menjalankan atau memberhentikan rekaman oleh pengguna.

Fungsi menjalankan atau mematikan rekaman dibagi menjadi dua yaitu fungsi rekaman dalam IDE dengan data rekaman dinamakan `Recording` dan fungsi timer yang digunakan untuk memberitahu kepada pengguna progress waktu pemutaran rekaman dinamakan `Timer`. Fungsi `Recording` menggunakan fungsi dalam *Library Ace* dan fungsi *javascript* `javascipt` untuk memperbaiki IDE antarmuka berdasarkan `event` yang dipanggil, fungsi `setTimeout` dalam *javascript* akan digunakan untuk menjalankan `event` selanjutnya bedasarkan perbedaan waktu antara event sekarang dan event selanjutnya dengan memanggil fungsi `playRecording` dengan `event` selanjutnya. Sedangkan fungsi `Timer` menggunakan fungsi `setInterval` yang akan dijalankan berulang untuk setiap detiknya dan memperbaiki progress waktu dalam antarmuka berdasarkan waktu yang sudah lewat.

- Menampilkan Histogram Events yang Terjadi

Pada fungsi `getRecording` setelah memuat data rekaman dan menformat data rekaman tersebut, fungsi `setUpChart` akan dipanggil dan membuat data grafik histogram. Data histogram akan dimuat menggunakan *Library Chart.js*.

Perbaikan Implementasi

Untuk fitur ini, tidak dilakukan perubahan pada implementasi yang sudah ada karena dianggap sudah memenuhi kebutuhan pengguna.

5.2 Pengujian Fungsional

Pengujian fungsional dilakukan secara lokal. Berikut merupakan pengujian terhadap fitur-fitur sistem pemutaran ulang dalam SharIF Judge:

Tabel 5.1: Tabel Pengujian Fungsional

No.	Aksi Pengguna	Reaksi yang diharapkan	Reaksi
1	Memilih masalah dalam Halaman Submit	Sistem memulai rekaman	sesuai
2	Melakukan Aksi <i>Save</i>	Menyimpan rekaman sesi dalam sistem	sesuai
3	Melakukan Aksi <i>Save & Execute</i>	Menyimpan rekaman sesi dalam sistem	sesuai
4	Melakukan Aksi <i>Save & Submit</i>	Menyimpan rekaman sesi dalam sistem	sesuai
5	Membuka Halaman Daftar Rekaman	Daftar Rekaman ditampilkan	sesuai
6	Memilih Rekaman dalam Halaman Rekaman	Memuka halaman Rekaman	sesuai
7	Menyaring Daftar Rekaman dengan User	Daftar rekaman menampilkan hanya daftar rekaman user	sesuai
8	Menyaring Daftar Rekaman dengan Problem	Daftar rekaman menampilkan hanya daftar rekaman untuk problem saringan	sesuai
9	Menyaring Daftar Rekaman dengan User dan Problem	Daftar rekaman menampilkan hanya daftar rekaman user pada problem saringan	sesuai
10	Membuka Halaman Rekaman	Rekaman ditampilkan	sesuai
11	Menjalankan Rekaman	Rekaman diputar ulang	sesuai
12	Memberhentikan Rekaman yang berjalan	Putaran ulang rekaman berhenti	sesuai

5.3 Pengujian Eksperimental

Pada Bagian ini dilakukannya pengujian terhadap sistem Perekaman Ulang pada SharIF Judge.

5.3.1 Lingkungan pengujian

Pengujian sistem perekaman ulang akan dilakukan pada sebuah server VPS atau *Virtual Private Server*. Pada server akan dijalankannya *docker* agar sistem aplikasi akan identik dengan pada saat sistem sedang diimplementasikan. Kode C.1 dan Kode C.2 merupakan file *docker-compose* dan *Dockerfile* yang digunakan membangun sistem SharIF Judge dalam server VPS.

Tabel 5.2 menunjukkan spesifikasi perangkat keras yang digunakan saat pengujian.

Tabel 5.2: Perangkat Keras Lingkungan Pembangunan

Parameter	Nilai
<i>Processor</i>	<i>AMD EPYC 9354P 2 vCPU</i>
<i>Random Access Memory (RAM)</i>	8 GB
<i>Storage</i>	100 GB SSD

Tabel 5.3 menunjukkan spesifikasi perangkat lunak yang digunakan saat pengujian.

Tabel 5.3: Perangkat Lunak Lingkungan Pembangunan

Parameter	Nilai
Sistem Operasi	<i>Debian Version 12</i>
Bahasa Pemrograman	<i>PHP, JavaScript, CSS, dan HTML</i>
<i>Framework</i>	<i>CodeIgniter 3.1.13</i>
<i>Code Editor</i>	<i>Visual Studio Code 1.99.3</i>
Perangkat Lunak Pendukung	<i>Docker Version 20.10.24+dfsg1</i> <i>Debian 11-slim</i> <i>MySQL 5.7</i> <i>phpMyAdmin 5.2.1</i> <i>PHP 7.3.33</i>

5.3.2 Eksperimen

Pengujian dilakukan untuk mengetahui permasalahan yang terjadi jika IDE dalam SharIF Judge dimasukkan sistem perekaman. Hasil Pengujian juga akan dianalisis secara sederhana agar dapat dibuatnya sebuah sistem untuk mendeteksi tindakan kecurangan.

Pengujian pada sistem pemutaran ulang dilakukan dengan cara mengajak beberapa peserta yang masih menempuh kuliah maupun yang sudah lulus dengan mengerjakan tiga permasalahan dengan tingkat kesusahan mudah, sedang, dan sulit dalam waktu tempuh satu hari. Para peserta dianjurkan mengerjakan 2 soal yang sudah disediakan dan dapat melihat *syntax* bahasa pemrograman. Pada pengujian peserta juga dapat melakukan kecurangan pada satu buah nomor dengan cara apapun. Kecurangan akan dianggap terjadi pada saat peserta melihat cara pengerjaan permasalahan dengan cara apapun.

Berikut merupakan masalah yang ditemukan pada saat pengujian sistem pemutaran ulang:

- Fitur menampilkan soal pada IDE

Fitur ini tidak dapat dilakukan karena permasalahan dengan *networking* dalam server VPS yang menjadikan tidak bisa mengakses PDF permasalahan dalam SharIF Judge dan menghasilkan *status 404*. Fitur ini diabaikan karena fitur ini tidak mengganggu pengujian yang sedang berjalan, tetapi untuk peserta melihat soal dengan cara mendownload soal dan melihatnya pada aplikasi lain.

- Fitur Menyimpan Rekaman pada Sistem

Fitur ini tidak mengubah database pada saat pengguna melakukan aksi *submit*. Pada saat aksi *submit* dilakukan file rekaman dibuat menjadi file rekaman yang sudah di-*submit* tetapi pada daftar rekaman dalam database itu sendiri belum dihapus dan pada saat rekaman diambil, tidak ditemukannya rekaman tersebut dan mengembalikan error pada pengguna. Untuk menyelesaikan masalah ini, diperlukannya sebuah fungsi baru dalam *Model Recording_model*. Kode 5.8 merupakan fungsi yang ditambahkan pada *Model Recording_model*.

Kode 5.8: Fungsi tambahan pada *Recording model*

```
1 public function remove_saveonly_recording($assignment_id, $problem_id, $username) {
```

```

2     $this->db->delete('recording', array(
3         'assignment' => $assignment_id,
4         'problem' =>$problem_id,
5         'username'=>$username,
6         'rec_id'=>0)
7     );
8 }

```

Penambahan fungsi ini akan digunakan pada *Controller Recording* pada saat fungsi `_submit()` dipanggil.

Setelah pengujian berakhir, peserta diminta untuk mengisi beberapa pertanyaan mengenai permasalahan yang dikerjakan dan juga beberapa pertanyaan mengenai pengalaman mengerjakan permasalahan pemrograman.

Analisa Hasil Pengujian

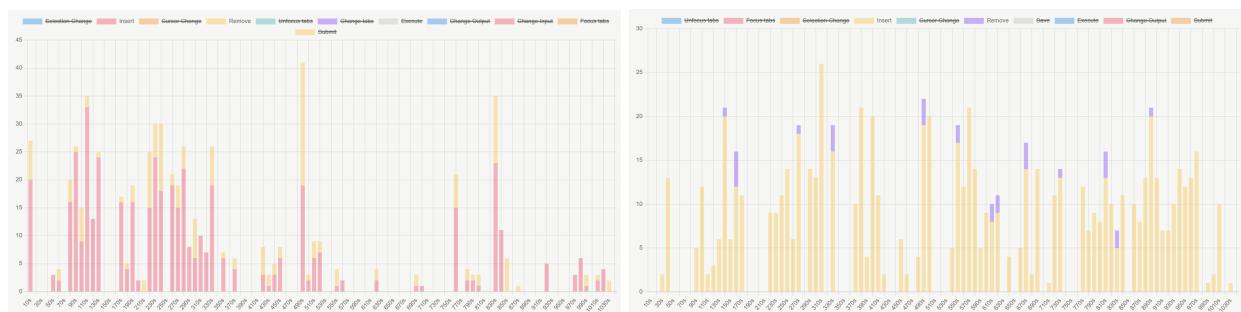
Analisa ini dilakukan dengan melihat data yang rekaman yang sudah di proses dan menjadi bagan histogram dalam sistem dan bagan lainnya yang dibutuhkan pada proses analisa. Ada beberapa pola yang dapat dilihat pada saat melihat histogram hasil rekaman peserta pengujian. Berikut merupakan pola yang dapat dilihat pada:

- Pola Pembuatan Kode

Pola ini dapat dilihat pada `events editor.change` dan hasil kode yang dibuat. Pola ini didasari dengan adanya *Code Churn* atau bisa disebut dengan penggerjaan ulang kode. *Code Churn* ini terjadi pada saat peserta atau mahasiswa manghapus atau menulis ulang kode program [9]. Pada *Code Churn* ada yang dinamakan *Code Churn Rate* adalah sebuah tingkat pergantian kode selama sedang mengerjakan kode program tersebut. *Code Churn Rate* dapat menilai seberapa banyak kode berubah dalam suatu tim, module, file maupun fungsi. Pada pola ini digunkannya *Code Churn Rate* dalam tingkat file sebagai sebuah penilaian yang dapat dilakukan dengan menggunakan rumus sebagai berikut:

$$(\text{inserted_kode} + \text{removed_kode}) / \text{total_kode}$$

`inserted_kode` dihitung dengan menjumlahkan total karakter yang di masukkan dalam editor. Begitu juga dengan `removed_kode` dihitung dengan menjumlahkan total karakter yang dihapus dalam editor. `total_kode` dihitung dengan menjumlahkan jumlah total karakter pada hasil kode program akhir. Berikut merupakan contoh perhitungan untuk *Code Churn Rate*:



Gambar 5.1: Bagan Histogram Perubahan Kode Program

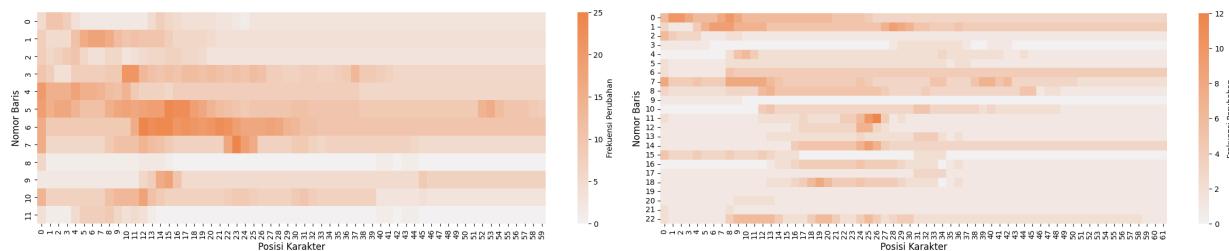
Pada Gambar 5.1 sebelah kiri peserta mengerjakan dengan tidak melakukan kecurangan. Terjadi 1436 karakter yang dimasukkan dan 2240 karakter yang dihapus pada editor dengan total kode akhir memiliki panjang 362 karakter. Maka menggunakan rumus di atas, *Code Churn Rate* pada peserta ini menjadi 10.15. Dengan nilai *Code Churn Rate* yang tinggi maka peserta ini banyak mengubah kode program.

Sebaliknya pada Gambar 5.1 sebelah kanan peserta menggunakan AI untuk mengerjakan dengan men-copy hasil kode AI ke dalam editor. Terjadi 1723 karakter yang dimasukkan dan

28 karakter yang dihapus pada editor dengan total kode akhir memiliki panjang 1771 karakter. Maka menggunakan rumus di atas, *Code Churn Rate* pada peserta ini menjadi 0.99. Dengan nilai *Code Churn Rate* yang rendah maka peserta ini tidak banyak mengubah kode program. Maka Pola Pembuatan Kode ini dinilai dengan *Code Churn Rate*, semakin tinggi nilai tersebut maka semakin tinggi juga perubahan kode yang terjadi. Jika banyak perubahan kode maka dimungkinkan bahwa peserta tidak melakukan kecurangan pada saat pengerjaan.

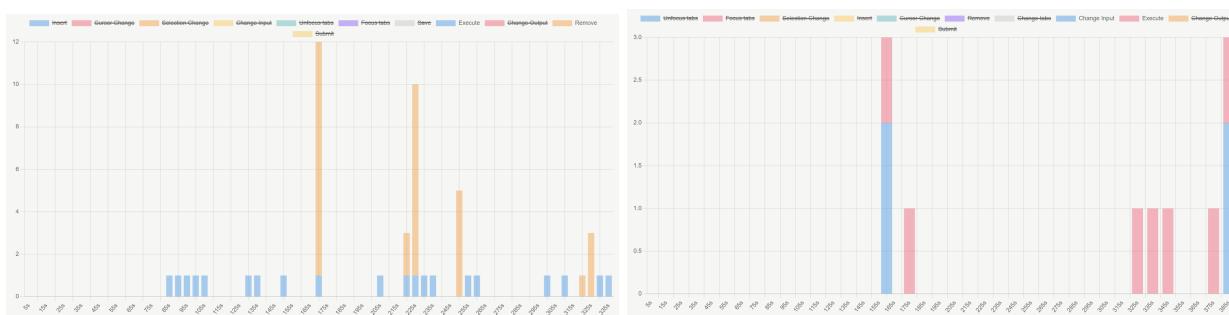
- Pola *Debugging*

Pola ini dapat dilihat pada `events editor.change`, `editor.changeCursor`, `editor.changeSelection`, `editor_input.input`, dan `editor_output.change`. Pengguna yang *debugging* akan mengubah kode pada lokasi yang sama. Gambar 5.2 sebelah kiri menunjukkan bahwa frekuensi lebih tinggi akan berwarna lebih merah dan menandakan pola debugging pada posisi tertentu yang dapat dilihat karena terjadinya banyak perubahan pada posisi yang sama dibandingkan dengan Gambar 5.2 sebelah kanan yang memiliki frekuensi maksimum 12 pada warna termerahnya.



Gambar 5.2: Bagan Heatmap Perubahan Lokasi Kode Program

Peserta yang *debugging* juga akan mencoba untuk mengubah input dan melakukan aksi *Execute* lebih banyak dibandingkan dengan peserta yang tidak melakukan *debugging*. Gambar 5.3 sebelah kiri menunjukkan warna kuning sebagai perubahan input dan warna biru sebagai aksi *Execute* yang dilakukan sedangkan Gambar 5.3 sebelah kanan menunjukkan warna biru sebagai perubahan input dan warna merah sebagai aksi *Execute* yang dilakukan. Perbedaan kedua bagan adalah frekuensi dimana aksi *Execute* dilakukan dan juga perubahan input. Maka pola *debugging* juga dapat dilihat melalui frekuensi aksi *Execute* dan juga frekuensi perubahan input.



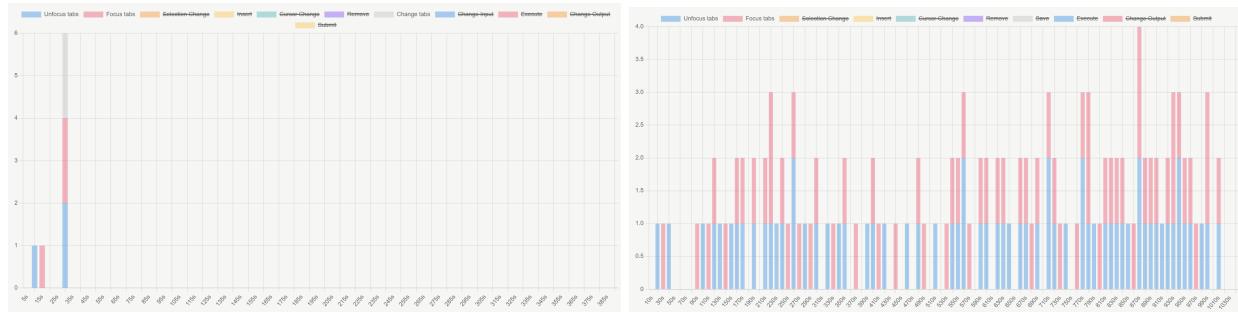
Gambar 5.3: Bagan Histogram Perubahan Input dan Aksi *Execute*

Pola *Debugging* ini akan mendukung Pola Pembuatan Kode untuk mengetahui apakah peserta melakukan kecurangan. Tetapi pola ini tidak dapat mengetahui pasti peserta yang melakukan kecurangan.

- Pola Perubahan Navigasi

Pola ini dapat dilihat pada `events window.focus`, `window.blur`, dan `window.visibilitychange`. Pola ini dilihat dengan membandingkan seberapa sering pengguna mengubah navigasi dalam sesi mengerjakan kode program. Gambar 5.4 menunjukkan perbedaan bagan histogram

event window.focus, window.blur, dan window.visibilitychange pada peserta yang men-copy kode program dari jawaban *online* dan peserta yang mengerjakannya hanya pada website SharIF Judge.

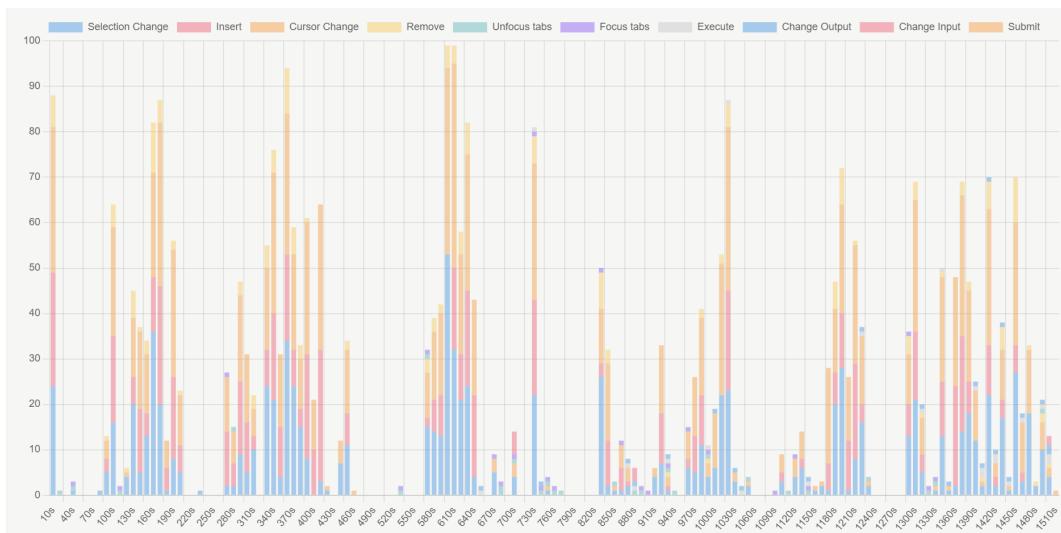


Gambar 5.4: Bagan Histogram Perubahan Navigasi

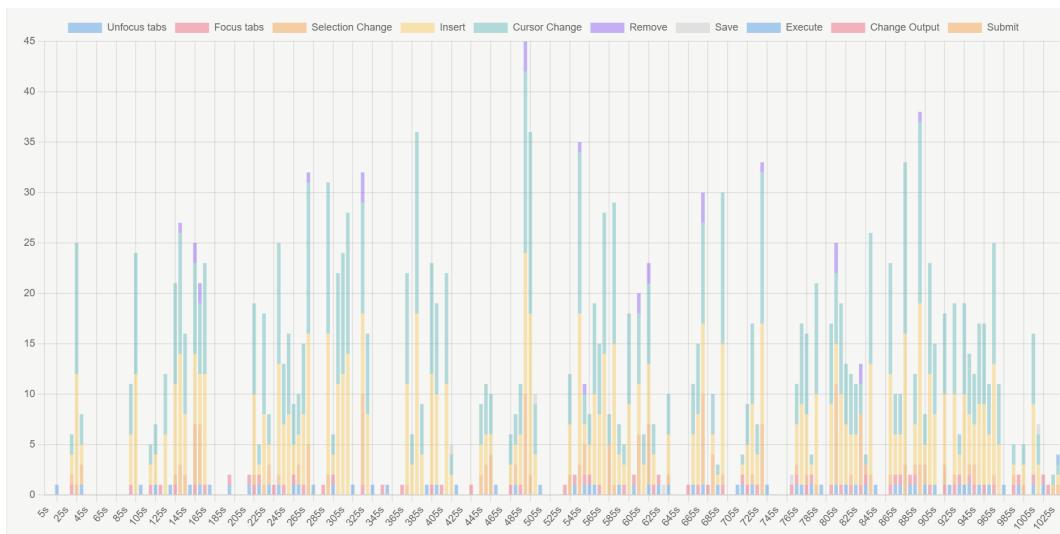
Pola Perubahan Navigasi ini dapat diukur dengan membatasi perubahan *tab* atau fokus web page untuk waktu tertentu. Jika nilai melebihi batas, maka peserta dapat diduga melakukan kecurangan.

- Pola Berpikir

Pola ini dapat dilihat pada keseluruhan *events* yang terjadi. Pada Pola ini dapat dilihat dari saat pemberhentian peserta sebelum melakukan perubahan yang kompleks terhadap kode program dan juga keseringannya terjadinya pemberhentian.



Gambar 5.5: Bagan Histogram Perubahan



Gambar 5.6: Bagan Histogram Perubahan

Gambar 5.5 memiliki histogram yang dipisah setiap sepuluh detik dan memiliki waktu yang cukup lama sebelum melakukan perubahan yang cukup besar. Sebaliknya Gambar 5.6 menunjukkan peserta yang men-*copy* kode dari sumber jawaban *online*. Bagan ini menunjukkan lebih seringnya berhenti untuk melihat jawaban terlebih dahulu dan durasi pemberhentianya lebih kecil dibandingkan bagan histogram pertama.

Pola ini dinilai dengan memberikan batas frekuensi berpikir dan lamanya berpikir dalam suatu waktu. Jika nilai melebihi batas, maka peserta dapat diduga melakukan kecurangan.

- Pola *Copy-Paste*

Pola ini merupakan pola sederhana dimana jika terjadinya penambahan kode yang panjang tetapi tidak melakukan penghapusan yang panjang terlebih dahulu, maka akan diduga melakukan kecurangan.

Tetapi pola-pola tersebut dapat juga mendapatkan *false positive* karena banyaknya *variable* yang harus diperhatikan dalam menentukan tidakan kecurangan itu sendiri. *Variable* yang dimaksudkan antara lain adalah kesulitan masalah dan pengalaman programing. Jika kesulitan masalah mudah maka peserta dapat mengerjakan dengan mudah dan tidak memerlukannya *debugging*, begitu juga jika pengalaman programing yang tinggi.

BAB 6

KESIMPULAN DAN SARAN

Bab ini akan membahas kesimpulan dari hasil analisis, perancangan, implementasi, pengujian, dan saran-saran untuk pengembangan yang dapat dipertimbangkan.

6.1 Kesimpulan

Berdasarkan proses dan hasil analisis, perancangan, implementasi, dan pengujian perangkat lunak yang telah dibuat, maka diperoleh kesimpulan sebagai berikut:

1. Implementasi fitur merekam dan memutar ulang ketikan pada SharIF judge berhasil dikembangkan dengan memanfaatkan fitur *Library ace*, dan fitur pada javascript.
2. Fitur pemutaran ulang ini dapat membantu dosen dalam menganalisa proses berpikir mahasiswa saat menyelesaikan sebuah masalah pemrograman dan menjadi bukti indikasi terjadinya kecurangan.

6.2 Saran

Berdasarkan proses dan hasil analisis, perancangan, implementasi, dan pengujian perangkat lunak yang telah dibuat, maka diperoleh saran-saran sebagai berikut:

1. Menambahkan analisis otomatis untuk mendeteksi pola kecurangan dari data ketikan.
2. Optimasi penyimpanan data rekaman dengan kompresi data.
3. Pada saat pengujian, banyak peserta yang menyampaikan keluhan mengenai perilaku IDE terutama untuk mengetahui kesalahan dalam kode program berbahasa *python*. Maka IDE dapat dioptimasi agar kesalahan dalam kode program dapat dimunculkan.
4. Menambahkan sebuah pilihan pada saat pembuatan tugas atau *assignment* untuk menyalakan *auto-complete* kode dalam SharIF Judge.

DAFTAR REFERENSI

- [1] Prihatini, F. N. dan Indudewi, D. (2016) Kesadaran dan Perilaku Plagiarisme dikalangan Mahasiswa(Studi pada Mahasiswa Fakultas Ekonomi Jurusan Akuntansi Universitas Semarang). *Dinamika Sosial Budaya*, **18**, 68–75.
- [2] Önder Demir, Aykut Soysal, Ahmet Arslan, Burcu Yürekli, dan Özgür Yilmazel (2010) Automatic grading system for programming homework. *Proceedings of the Annual International Conference on Computer Science Education: Innovation & Technology CSEIT 2010 & Proceedings of the Annual International Conference on Software Engineering SE 2010*, **18**, 68–75.
- [3] Kurnia, A., Lim, A., dan Cheang, B. (2001) Online judge. *Computers & Education*, **18**, 299–315.
- [4] IDCloudHost (2020) Algoritma pemrograman beserta contohnya. <https://idcloudhost.com/blog/algoritma-pemrograman-pengertian-fungsi-cara-kerja-dan-contohnya/>. 6 Desember 2024.
- [5] Vallian, S. (2018) Kustomisasi Sharif Judge Untuk Kebutuhan Program Studi Teknik Informatika. Skripsi. Universitas Katolik Parahyangan, Indonesia.
- [6] Version 1.4 (2014) *Sharif Judge Documentation*. Mohammad Javad Naderi. Tehran, Iran.
- [7] Halim, N. A. (2021) Implementasi Editor Kode pada SharIF Judge. Skripsi. Universitas Katolik Parahyangan, Indonesia.
- [8] Version (2025) *Chart.js Documentation*. Chart.js Contributors. Open Source Project.
- [9] Ghufran, H. (2022) What is code churn? <https://www.hatico.io/blog/code-churn-rate/>. 2 Juni 2025.

LAMPIRAN A

KODE PROGRAM

Kode A.1: Install.php

```

1 diff --git a/application/controllers/Install.php b/application/controllers/Install.php
2 index 2d87d1f..389e195 100644
3 --- a/application/controllers/Install.php
4 +++ b/application/controllers/Install.php
5 @@ -45,7 +45,7 @@ class Install extends CI_Controller
6
7     // Use InnoDB engine for MySQL database
8     if ($this->db->dbdriver === 'mysql' || $this->db->dbdriver === 'mysqli')
9     -
10        $this->db->query('SET storage_engine=InnoDB;');
11    +
12        $this->db->query('SET default_storage_engine=InnoDB;');
13
14     // Creating Tables:
15     // sessions, submissions, assignments, notifications, problems, queue, scoreboard, settings, users
16     @@ -86,9 +86,30 @@ class Install extends CI_Controller
17         $this->dbforge->add_key(array('assignment', 'submit_id'));
18         if ( ! $this->dbforge->create_table('submissions', TRUE))
19             show_error("Error creating database table ".$this->db->dbprefix('submissions'));
20
21     +
22     +
23     // create table 'recording'
24     $fields = array(
25         'rec_id'      => array('type' => 'INT', 'constraint' => 11, 'unsigned' => TRUE),
26         'upload_at'   => array('type' => $DATETIME),
27         'assignment'  => array('type' => 'SMALLINT', 'constraint' => 4, 'unsigned' => TRUE),
28         'problem'    => array('type' => 'SMALLINT', 'constraint' => 4, 'unsigned' => TRUE),
29         'username'   => array('type' => 'VARCHAR', 'constraint' => 20),
30     );
31     $this->dbforge->add_field($fields);
32     if ( ! $this->dbforge->create_table('recording', TRUE))
33         show_error("Error creating database table " . $this->db->dbprefix('recording'));
34     // ADD Unique constraint
35     $this->db->query(
36         "ALTER TABLE {$this->db->dbprefix('recording')}"
37         "ADD CONSTRAINT {$this->db->dbprefix('srup_unique')} UNIQUE (rec_id, username, assignment, problem);"
38     );
39
40     // create table 'assignments'
41     $fields = array(
42         'id'          => array('type' => 'INT', 'constraint' => 11, 'unsigned' => TRUE, 'auto_increment' => TRUE),

```

Kode A.2: main.css

```

1 diff --git a/assets/styles/main.css b/assets/styles/main.css
2 index 57a6f61..1b3d981 100644
3 --- a/assets/styles/main.css
4 +++ b/assets/styles/main.css
5 @@ -432,6 +432,7 @@ div#extra_time i {
6     .color-notifications {border-color: #FAE80B!important;}
7     .color-assignments {border-color: #CCDB29!important;}
8     .color-problems {border-color: #0DB297!important;}
9     +.color-recording {border-color: #0d99b2!important;}
10    .color-submit {border-color: #35C0CB!important;}
11    .color-final_submissions {border-color: #22478A!important;}
12    .color-all_submissions {border-color: #543680!important;}
13 @@ -445,6 +446,7 @@ div#extra_time i {
14     .color-notifications.selected i {background-color: #FAE80B!important; border-color: #FAE80B!important;}
15     .color-assignments.selected i {background-color: #CCDB29!important; border-color: #CCDB29!important;}
16     .color-problems.selected i {background-color: #0DB297!important; border-color: #0DB297!important;}
17     +.color-recording.selected i {background-color: #0d99b2!important; border-color: #0d99b2!important;}
18     .color-submit.selected i {background-color: #35C0CB!important; border-color: #35C0CB!important;}
19     .color-final_submissions.selected i {background-color: #22478A!important; border-color: #22478A!important;}
20     .color-all_submissions.selected i {background-color: #543680!important; border-color: #543680!important;}

```

Kode A.3: recording.css

```

1 /* Overall */
2
3 .title_section {
4     margin: 0;

```

```
5     display: flex;
6     align-items: center;
7     font-size: 1.5em;
8 }
9
10 .title_normal {
11     font-weight: normal;
12     justify-content: flex-start;
13 }
14
15 /* Loader */
16
17 .loader {
18     border: 4px solid rgba(255, 255, 255, 0.3);
19     border-top: 4px solid #3498db;
20     border-radius: 50%;
21     width: 40px;
22     height: 40px;
23     animation: spin_2s_linear_infinite;
24     -webkit-animation: spin_2s_linear_infinite; /* Safari */
25 }
26
27 /* Safari */
28 @-webkit-keyframes spin {
29     0% {
30         -webkit-transform: rotate(0deg);
31     }
32     100% {
33         -webkit-transform: rotate(360deg);
34     }
35 }
36
37 @keyframes spin {
38     0% {
39         transform: rotate(0deg);
40     }
41     100% {
42         transform: rotate(360deg);
43     }
44 }
45
46 /* Recording Header */
47
48 div#wrap_selector {
49     display: grid;
50     grid-template-columns: minmax(100px, auto) 1fr;
51     column-gap: 20px;
52     row-gap: 8px;
53     margin-bottom: 8px;
54 }
55
56 h2.title_selector {
57     justify-content: flex-end;
58 }
59
60 table#wrap_sel_problem {
61     table-layout: fixed;
62     width: 100%;
63     height: 32px;
64 }
65
66 td.sel_problem {
67     text-align: center;
68     border-bottom: 2px solid #dddad0;
69     border-left: 2px solid #dddad0;
70     background: #faf9f5;
71 }
72
73 td.selected {
74     background: #e0e0e0;
75 }
76
77 td.sel_problem > a {
78     display: flex;
79     justify-content: center;
80     align-items: center;
81     width: 100%;
82     height: 100%;
83     text-decoration: none;
84     /* text-align: center; */
85     /* vertical-align: middle; */
86 }
87
88 div#recording_wrap {
89     display: grid;
90     /* width: ; */
91     min-height: 50vh;
92     gap: 8px;
93 }
94
95 /* IDE In Recording */
96
97 div#ide_wrap {
98     display: flex;
99     /* width: calc(100vw - 200px); */
100    height: 50vh;
101    padding: 0;
102    border: 0;
103    position: relative;
```

```
104 }
105 fieldset#editor_wrap {
106   width: 70%;
107   height: 100%;
108   padding: 0;
109   border: 0;
110 }
111
112 div#code_editor {
113   width: 100%;
114   height: 100%;
115   padding: 0;
116   border: 0;
117 }
118
119 fieldset#in_wrap {
120   width: 15%;
121   height: 100%;
122   padding: 0;
123   border: 0;
124 }
125
126 fieldset#out_wrap {
127   width: 15%;
128   height: 100%;
129   padding: 0;
130   border: 0;
131 }
132
133 textarea.in_out {
134   resize: none;
135   width: 100%;
136   height: 100%;
137 }
138
139 /* Status */
140
141 #status_wrapper {
142   position: absolute;
143   display: flex;
144   /* display: none; */
145   align-items: center;
146   justify-content: center;
147   top: 0;
148   bottom: 0;
149   left: 0;
150   right: 0;
151   background-color: rgba(0, 0, 0, 0.3);
152   z-index: 10;
153   border-radius: 8px;
154   font-size: 5em;
155   font-weight: 900;
156   color: #fff;
157 }
158
159 /* Input Player */
160
161 div.wrap_input_player {
162   display: flex;
163   flex-direction: row;
164   align-items: center;
165   gap: 8px;
166 }
167
168 div.wrap_input_player > input {
169   flex: 1;
170 }
171
172 #wrap_input_range {
173   width: 100%;
174   display: flex;
175   flex-direction: row;
176 }
177
178 .wrap_range {
179   width: 100%;
180 }
181
182 .range {
183   appearance: none;
184   background: #444;
185   cursor: pointer;
186   height: 8px;
187   margin: 0;
188   transition: 0.1s ease-in;
189   vertical-align: bottom;
190   width: 100%;
191   border-radius: 8px;
192   margin-bottom: 4px;
193 }
194 .wrap_range:hover .range,
195 .wrap_range.hover .range {
196   width: 16px;
197   margin-bottom: 0px;
198 }
199
200 .range::-webkit-slider-thumb {
201   appearance: none;
```

```

203     background:#aeaea;
204     border-radius:8px;
205     box-shadow:inset 0 0 0 5px #aeaea;
206     height:0;
207     transition:0.1s_ease-in;
208     width:0;
209 }
210 .wrap_range:hover.range::-webkit-slider-thumb,
211 .wrap_range.hover.range::-webkit-slider-thumb{
212   width:16px;
213   height:16px;
214 }
215 /* List Recording */
216 .sel_recording {
217   cursor:pointer;
218 }
219 .sel_selected {
220   font-weight:bold;
221   text-decoration:underline;
222 }
223 /* Chart */
224 #wrap_config_chart {
225   display:grid;
226   gap:16px;
227   grid-template-columns:100px_1fr;
228 }
```

Kode A.4: Recording.php

```

1 <?php
2 /**
3  * SharIF Judge online judge
4  * @file Recording.php
5  * @author Andreas Ronaldi <andreasronaldi25@gmail.com>
6  */
7 */
8 defined('BASEPATH') or exit('No direct script access allowed');
9
10 class Recording extends CI_Controller
11 {
12   public function __construct()
13   {
14     parent::__construct();
15     if (! $this->session->userdata('logged_in')) // if not logged in
16       redirect('login');
17     if ($this->user->level <= 1) // permission denied
18       show_404();
19
20     $this->load->model('recording_model');
21
22     $input = $this->uri->uri_to_assoc();
23
24     // var_dump($input);
25
26     $this->filter_user = $this->filter_problem = NULL;
27     if (array_key_exists('user', $input) && $input['user'])
28       $this->filter_user = $this->form_validation->alpha_numeric($input['user'])? $input['user']:NULL;
29     if (array_key_exists('problem', $input) && $input['problem'])
30       $this->filter_problem = is_numeric($input['problem'])? $input['problem']:NULL;
31   }
32
33   // For seeing all recording
34   public function all()
35   {
36     $assignment_id = $this->user->selected_assignment['id'];
37
38     if ($assignment_id == 0)
39       show_error('No_assignment_selected.');
40
41     $problem = $this->assignment_model->all_problems($assignment_id);
42     $assignment = $this->assignment_model->assignment_info($assignment_id);
43     $recordings = $this->recording_model->all_user_recordings($assignment_id, $this->filter_problem, $this->filter_user);
44     $names = $this->user_model->get_names();
45
46     foreach ($recordings as &$item) {
47       $item['name'] = $names[$item['username']];
48     }
49
50     $data = array(
51       'all_problems' => $problem,
52       'assignment' => $assignment,
53       'recordings' => $recordings,
54       'filter_problem' => $this->filter_problem,
55       'filter_user' => $this->filter_user,
56     );
57
58     $this->twig->display('pages/recording_list.twig', $data);
59   }
60
61   public function index($assignment_id = NULL, $problem_id = 1, $username = NULL, $rec_id = NULL)
62   {
63     // If no assignment is given
64     if ($assignment_id === NULL)
65       redirect('recording/all');
```

```

66|     if ($assignment_id == 0)
67|         show_error('No_assignment_selected.');
68|
69|     $assignment = $this->assignment_model->assignment_info($assignment_id);
70|
71|     $data = array(
72|         'all_problems' => $this->assignment_model->all_problems($assignment_id),
73|         'assignment' => $assignment,
74|         'filter_user' => $username,
75|         'filter_problem' => $problem_id,
76|     );
77|
78|     if ($username !== NULL) {
79|         $list_rec = $this->recording_model->get_user_recordings($assignment_id, $problem_id, $username);
80|         $data['list_recording'] = $list_rec;
81|
82|         if ($rec_id === NULL && $list_rec) {
83|             $rec_id = $list_rec[0]['rec_id'];
84|         }
85|
86|         $data['rec_id'] = $rec_id;
87|     }
88|
89|     $this->twig->display('pages/recording.twig', $data);
90}
91|
92|
93| public function download_record($assignment_id, $problem_id, $username, $rec_id)
94| {
95|     $assignment_root = rtrim($this->settings_model->get_setting('assignments_root'), '/');
96|     $file_path = $assignment_root.'/assignment'.$assignment_id.'/'.$problem_id.'/'.$username;
97|     $rec_path = $file_path.'/'.$RECORD_FILE_NAME.'.'.$RECORD_FILE_EXT;
98|
99|     if ($rec_id !== "0") {
100|         $rec_path = $file_path.'/'.$RECORD_FILE_NAME.'-'.$rec_id.'.'.$RECORD_FILE_EXT;
101|     }
102|
103|     $this->load->helper('file');
104|     $this->load->helper('url');
105|
106|     if (!file_exists($rec_path)) {
107|         throw new Exception("File $rec_path does not exist");
108|     }
109|     if (!is_readable($rec_path)) {
110|         throw new Exception("File $rec_path is not readable");
111|     }
112|
113|     $file = glob($rec_path);
114|     $content = file_get_contents($file[0]);
115|     header('Content-Type: application/json');
116|     header('Content-Disposition: attachment; filename="rec.json"');
117|     die($content);
118}
119}

```

Kode A.5: recording.twig

```

1  {#
2  # SharIF Judge
3  # file: recording.twig
4  # author: Andreas Ronaldi <andreasronaldi25@gmail.com>
5  #}
6  {% set selected = 'recording' %}
7  {% extends 'templates/base.twig' %}
8  {% block icon %}fa-video-camera
9  {% endblock %}
10 | {% block title %}Recording
11 | {% endblock %}
12 | {% block head_title %}Recording
13 | {% endblock %}
14 |
15 |
16 | {% block other_assets %}
17 |     <link rel="stylesheet" type='text/css' href="{{ _base_url('assets/styles/recording.css') }}"/>
18 |
19 |     <script src="{{ base_url('assets/ace/ace.js') }}></script>
20 |     <script src="{{ base_url('assets/ace/ext-language_tools.js') }}></script>
21 |
22 |     <script>
23 |         var rec_path='{{ assignment.id }}/{{ filter_problem }}/{{ filter_user }}/{{ rec_id }}';
24 |     </script>
25 |
26 |     <script src="{{ _base_url("assets/chartjs/dist/chart.umd.js") }}></script>
27 |
28 |     <script type='text/javascript' src="{{ _base_url("assets/js/shj_recording.js") }}></script>
29 | {% endblock %}
30 |
31 | {% block main_content %}
32 | <div id="main_rec">
33 |     {% if all_problems|length == 0 %}
34|         <p style="text-align:center;">Nothing to show...</p>
35 |     {% else %}
36 |         <div id="wrap_selector">
37 |             <h2 class="title_section_title_selector">Problem</h2>
38 |             <table id="wrap_sel_problem">
39 |                 <tr>
40 |                     {% for one_problem in all_problems %}
41 |                         <td class="sel_problem{{ _filter_problem == one_problem.id ? '_selected' }}>

```

```

42          <a dir="auto" href="{{_site_url("recording/#{{assignment.id}}/{{one_problem.id}}/{{filter_user}}")}}>{{
43              one_problem.name }}</a>
44      </td>
45      {% endfor %}
46  </tr>
47 </table>
48
49 <h2 class="title_section_title_selector">Username</h2>
50 <div id="wrap_sel_user">
51     <h2 class="title_section_title_normal">{{ filter_user }}</h2>
52 </div>
53 <h2 class="title_section_title_selector">Status</h2>
54 <h2 id="status_rec" class="title_section_title_normal"></h2>
55 </div>
56
57 {% if list_recording is not empty %}
58 <div id="recording_wrap">
59
60     {%# Chart #}
61     <h2 class="title_section">Player</h2>
62     <div id="ide_wrap">
63         <fieldset id="editor_wrap">
64             <legend>Code</legend>
65             <div id="code_editor"></div>
66         </fieldset>
67         <fieldset id="in_wrap">
68             <legend>Input</legend>
69             <textarea id="editor_input" class="in_out" readonly></textarea>
70         </fieldset>
71         <fieldset id="out_wrap">
72             <legend>Output</legend>
73             <textarea id="editor_output" class="in_out" readonly></textarea>
74         </fieldset>
75         <div id="status_wrapper">
76             Status
77         </div>
78     </div>
79
80     {%# Player #}
81     <div class="wrap_input_player">
82         {%# For Playing Currently %}
83         <button type="button" class="sharif_input" id="rec_btn">Play</button>
84         <span id="timer_player">00:00</span>
85
86         <div class="wrap_range">
87             <input type="range" min="0" max="0" value="0" class="range" id="range_player">
88         </div>
89     </div>
90
91     {%# List of saved %}
92     <h2 class="title_section">List Of Work on Recording</h2>
93     <table class="sharif_table">
94         <thead>
95             <tr>
96                 <th>Start Time</th>
97                 <th>End Time</th>
98                 <th>Actions</th>
99             </tr>
100        </thead>
101        <tbody id="tbody_saved">
102        </tbody>
103    </table>
104
105    {%# Chart #}
106    <h2 class="title_section">Config Chart</h2>
107    <div class="wrap_config_chart">
108        <label for="config_chart_type">Chart Type</label>
109        <select id="config_chart_type" name="config_chart_type">
110            <option value="bar">Bar</option>
111            <option value="line">Line</option>
112        </select>
113        <label for="config_chart_stack">Stack</label>
114        <input type="checkbox" value="stack" id="config_chart_stack" name="config_chart_stack" checked></input>
115        <label for="config_chart_divider">Divider<label>
116        <input min="0" value="1" id="config_chart_divider" name="config_chart_divider" type="number"></input>
117        <label for="config_chart_time">Time<label>
118        <select id="config_chart_time" name="config_chart_time"></select>
119    </div>
120
121    <h2 class="title_section">Chart Of Action</h2>
122    <canvas id="recording_chart"></canvas>
123
124    <h2 class="title_section">Other Recording of {{ filter_user }} in problem {{ filter_problem }}</h2>
125    <table class="sharif_table">
126        <thead>
127            <tr>
128                <th>ID</th>
129                <th>Username</th>
130                <th>Submit Time</th>
131                <th>Actions</th>
132            </tr>
133        </thead>
134        <tbody>
135            {% for recording in list_recording %}
136            <tr>
137                <td>{{ recording.rec_id }}</td>
138                <td>{{ recording.username }}</td>
139                <td>{{ recording.upload_at }}</td>

```

```

140
141          <td>
142              <a dir="auto" href="{{site_url("recording/#{assignment.id}/#{filter_problem}/#{recording.username})}}>See Recording</a>
143          </td>
144      {% endfor %}
145  </tbody>
146 </table>
147 </div>
148 {% else %}
149 Nothing Here...
150 {% endif %}
151 {% endif %}
152 </div>
153 {% endblock %}
154 {# main_content #}

```

Kode A.6: recording_list.twig

```

1  {#
2  # SharIF Judge
3  # file: recording_list.twig
4  # author: Andreas Ronaldi <andreasronaldi25@gmail.com>
5  #}
6  {% set selected = 'recording' %}
7  {% extends 'templates/base.twig' %}
8  {% block icon %}fa-video-camera
9  {% endblock %}
10 {% block title %}Recording
11 {% endblock %}
12 {% block head_title %}Recording
13 {% endblock %}
14
15
16 {% block other_assets %}
17     <link rel="stylesheet" type='text/css' href="{{base_url('assets/styles/recording.css')}}"/>
18
19     <script>
20         var test = '{{ all_problems | json_encode(constant('JSON_PRETTY_PRINT')) }}';
21         console.log(test)
22     </script>
23 {% endblock %}
24
25
26 {% block main_content %}
27     {% if all_problems|length == 0 %}
28         <p style="text-align: center;>Nothing to show...</p>
29     {% else %}
30         <div id="wrap_selector">
31             <h2 class="title_section_title_selector">Problem</h2>
32             <table id="wrap_sel_problem">
33                 <tr>
34                     <td class="sel_problem{{_.filter_problem_is_null_?'selected'}}">
35                         <a dir="auto" href="{{site_url('recording/all/~(filter_user?/user/~filter_user))}}>All</a>
36                     </td>
37                     {% for one_problem in all_problems %}
38                         <td class="sel_problem{{_.filter_problem_==one_problem.id_?'selected'}}">
39                             <a dir="auto" href="{{site_url('recording/all/~(filter_user?/user/~filter_user)~/problem/~one_problem.id)}}>{{ one_problem.name }}</a>
40                         </td>
41                     {% endfor %}
42                 </tr>
43             </table>
44         </div>
45
46         <table class="sharif_table">
47             <thead>
48                 <tr>
49                     <th>ID</th>
50                     <th>Username</th>
51                     <th>Name</th>
52                     <th>Problem</th>
53                     <th>Submit Time</th>
54                     <th>Actions</th>
55                 </tr>
56             </thead>
57             {% for recording in recordings %}
58                 <tr>
59                     <td> {{ recording.rec_id }} </td>
60                     <td> <a dir="auto" href="{{site_url('recording/all/user/~recording.username~(filter_problem?/problem/~filter_problem))}}>{{ recording.username }}</a> </td>
61                     <td> {{ recording.name }} </td>
62                     <td> <a dir="auto" href="{{site_url('recording/all/~(filter_user?/user/~filter_user)~/problem/~recording.problem)}}>{{ recording.problem }}</a> </td>
63                     <td> {{ recording.upload_at }} </td>
64                     {% Action #}
65                     <td>
66                         <a dir="auto" href="{{site_url("recording/#{assignment.id}/#{recording.problem}/#{recording.username})}}>See Recording</a>
67                     </td>
68                 </tr>
69             {% endfor %}
70         </table>
71
72
73     {% if username != NULL %}
74     Please used /see_recording/...
75     {% endif %}

```

```

76    {% endif %}
77  {% endblock %}
78  {% main_content %}
```

Kode A.7: Recording_model.php

```

1 <?php
2
3 /**
4  * SharIF Judge online judge
5  * @file Submit_model.php
6  * @author Andreas Ronaldi <andreasronaldi25@gmail.com>
7  */
8 defined('BASEPATH') or exit('No direct script access allowed');
9
10 class Recording_model extends CI_Model
11 {
12     public function __construct()
13     {
14         parent::__construct();
15     }
16
17     public function all_user_recordings($assignment_id, $filter_problem = NULL, $filter_user = NULL) {
18         $arr['assignment'] = $assignment_id;
19         if ($filter_problem !== NULL)
20             $arr['problem'] = $filter_problem;
21         if ($filter_user !== NULL)
22             $arr['username'] = $filter_user;
23
24         return $this->db
25             ->order_by('upload_at_asc')
26             ->get_where('recording', $arr)
27             ->result_array();
28     }
29
30     public function get_user_recordings($assignment_id, $problem_id, $username) {
31         $arr['assignment'] = $assignment_id;
32         $arr['problem'] = $problem_id;
33         $arr['username'] = $username;
34
35         return $this->db
36             ->order_by('upload_at_asc')
37             ->get_where('recording', $arr)
38             ->result_array();
39     }
40
41     public function add_recording($rec_info) {
42         $this->db->replace('recording', $rec_info);
43     }
44
45     public function remove_saveonly_recording($assignment_id, $problem_id, $username) {
46         $this->db->delete('recording', array(
47             'assignment' => $assignment_id,
48             'problem' => $problem_id,
49             'username' => $username,
50             'rec_id' => 0
51         ));
52     }
53 }
```

Kode A.8: routes.php

```

1 diff --git a/application/config/routes.php b/application/config/routes.php
2 index 060bcbb8..a6c2d54 100644
3 --- a/application/config/routes.php
4 +++ b/application/config/routes.php
5 @@ -1,5 +1,5 @@
6 <?php
7 -defined('BASEPATH') OR exit('No direct script access allowed');
8 +defined('BASEPATH') or exit('No direct script access allowed');
9
10 /*
11 | -----
12 @@ -52,15 +52,19 @@ defined('BASEPATH') OR exit('No direct script access allowed');
13 $route['default_controller'] = 'dashboard';
14 $route['register'] = "login/register";
15 $route['logout'] = "login/logout";
16 $route['submissions/final']="submissions/the_final";
17 $route['submissions/final(.*)']="submissions/the_final/$1";
18 $route['profile/(:num)'] = "profile/index/$1";
19 $route['moss/(:num)'] = "moss/index/$1";
20 $route['problems/(:num)'] = "problems/index/$1";
21 $route['problems/(:num)/(:num)'] = "problems/index/$1/$2";
22 $route['rejudge/(:num)'] = "rejudge/index/$1";
23 +$route['recording/(:num)/(:num)/(:any)'] = "recording/index/$1/$2/$3";
24 +$route['recording/(:num)/(:num)/(:any)/(:num)'] = "recording/index/$1/$2/$3/$4";
25 $route['404_override'] = '';
26 $route['translate_uri_dashes'] = FALSE;
```

Kode A.9: shj_recording.js

```

1 /**
2  * SharIF Judge
```

```

3  * @file shj_recording.js
4  * author: Andreas Ronaldi <andreasronaldi25@gmail.com>
5  *
6  *     Javascript codes for "Recording" page
7  */
8
9 $(document).ready(() => {
10    // ##### Variable #####
11    // ##### Variable #####
12    // ##### Variable #####
13    const editor = ace.edit("code_editor");
14    const Range = ace.Range;
15
16    let funcTimeoutRecording = null;
17    let funcIntervalTimer = null;
18
19    let curChart = null;
20    let isRecPlaying = false;
21
22    let confTimer = {
23        delay: 1,
24    };
25
26    let confChart = {
27        type: "bar",
28        autoDivider: 20, // How many section does the auto divider will giveout (default: 10)
29        divider: 1,
30        time: "s",
31        stack: true,
32    };
33
34    let confPlayer = {
35        durationNext: 500, // in ms
36        mergeDuration: 0, // if time difference smaller than value, then merge the events
37    };
38
39    editor.setOptions({
40        theme: "ace/theme/monokai",
41        fontSize: "11pt",
42        readOnly: true,
43        enableLiveAutocompletion: true,
44        enableBasicAutocompletion: true,
45        enableSnippets: true,
46    });
47
48    // Object for playing record
49    const recording = {
50        events: {}, // Map -> time events to list of events
51        eventsIndex: {}, // Map -> index to time events
52        indexEvents: {}, // Map -> time events to index
53        presumIndexDuration: {}, // Map -> index to duration before timestamp event
54        length: -1, // Length of list of saved events
55        curEvents: -1, // Currently selected index
56        duration: 0, // Duration of events
57        save_state: [], // Saved state to use in playback
58
59        reset: () => {
60            recording.events = {};
61            recording.eventsIndex = {};
62            recording.indexEvents = {};
63            recording.presumIndexDuration = {};
64            recording.length = -1;
65            recording.curEvents = -1;
66            recording.duration = 0;
67            recording.save_state = [];
68        },
69    };
70
71    const handlersIncludeInGoState = {
72        insert: true,
73        remove: true,
74        cursor_selection: true,
75        sel_selection: true,
76        focus: false,
77        blur: false,
78        visibility: false,
79        pdf_focus: false,
80        pdf_blur: false,
81        input_change: true,
82        output_change: true,
83        save: false,
84        submit: false,
85        execute: false,
86    };
87
88    const handlers = {
89        insert: (args) => {
90            editor.session.replace(
91                new ace.Range(
92                    args.start.row,
93                    args.start.column,
94                    args.start.row,
95                    args.start.column
96                ),
97                args.data.join("\n")
98            );
99        },
100        remove: (args) => {
101            editor.session.remove({ start: args.start, end: args.end });
101

```

```

102 },
103 cursor_selection: (args) => {
104     setSelection(editor, args);
105 },
106 sel_selection: (args) => {
107     setSelection(editor, args);
108 },
109 focus: (args) => {
110     setStatus("User_is_focus_now");
111     setTitle("User_is_focus_now");
112 },
113 blur: (args) => {
114     setStatus("User_is_not_focus_on_website_now");
115     setTitle("User_is_not_focus_on_website_now");
116 },
117 visibility: (args) => {
118     if (args) {
119         setStatus("Open_ShariF-Judge");
120         setTitle("User_is_on_ShariF-Judge_now");
121     } else {
122         setStatus("Switching_tabs");
123         setTitle("User_is_on_other_tabs_now");
124     }
125 },
126 pdf_focus: (args) => {
127     setStatus("User_is_focus_on_pdf_viewer_now");
128     setTitle("User_is_focus_on_pdf_viewer_now");
129 },
130 pdf_blur: (args) => {
131     setStatus("User_is_not_focus_on_pdf_viewer_now");
132     setTitle("User_is_not_focus_on_pdf_viewer_now");
133 },
134 input_change: (args) => $("#editor_input").val(args),
135 output_change: (args) => $("#editor_output").val(args),
136 save: (args) => {
137     setStatus("User_just_saved");
138     setTitle("User_just_saved");
139 },
140 submit: (args) => {
141     setStatus("User_just_submit!");
142     setTitle("User_just_submit!");
143 },
144 execute: (args) => {
145     setStatus("User_is_running_the_program.");
146     setTitle("User_is_running_the_program.");
147 },
148 };
149
150 const mult = {
151     s: 1000,
152     m: 1000 * 60,
153     h: 1000 * 60 * 60,
154     d: 1000 * 60 * 60 * 24,
155     w: 1000 * 60 * 60 * 24 * 7,
156     y: 1000 * 60 * 60 * 24 * 7 * 365.25,
157 };
158
159 const nameInChart = {
160     insert: "Insert",
161     remove: "Remove",
162     cursor_selection: "Cursor_Change",
163     sel_selection: "Selection_Change",
164     focus: "Focus_tabs",
165     blur: "Unfocus_tabs",
166     visibility: "Change_tabs",
167     pdf_focus: "Focus_PDF_Viewer",
168     pdf_blur: "Unfocus_PDF_Viewer",
169     input_change: "Change_Input",
170     output_change: "Change_Output",
171     save: "Save",
172     submit: "Submit",
173     execute: "Execute",
174 };
175
176 // ##### Main Method #####
177 // #####
178 // #####
179
180 const getRecording = () => {
181     setTitle("Loading...");
182     setLoading(true);
183     disabledInput(true);
184     recording.reset();
185     emptyEditor();
186
187     console.log("GETTING_recording/download_record/" + rec_path);
188
189     $.ajax({
190         type: "GET",
191         url: shj.site_url + "recording/download_record/" + rec_path,
192         cache: false,
193         success: (data) => {
194             console.log(data);
195             recording.events = data;
196
197             $("select#rec_selection").empty();
198
199             recording.length = 0;
200

```

```

201     Object.keys(data).forEach((c, i, a) => {
202         // Push to table of saved
203         $('<tr>
204             <td>${convTimeToEpoch(c)}</td>
205             <td>${convTimeToEpoch(parseInt(c) + data[c][data[c].length - 1].time)}</td>
206             <td id="sel_${c}" class="sel_recording">
207                 Select
208             </td>
209         </tr>').appendTo("tbody#tbody_saved");
210
211         $('#sel_${c}').click(() => {
212             playOrStop(true);
213             setValueTimer(recording.presumIndexDuration[i]);
214             setSelectedSaveTime(c);
215         });
216
217         recording.eventsIndex[c] = i;
218         recording.indexEvents[i] = c;
219         recording.presumIndexDuration[i] = recording.duration;
220         recording.length++;
221
222         recording.duration += data[c][data[c].length - 1].time;
223
224         if (recording.curEvents === -1) {
225             recording.curEvents = c;
226         }
227
228         recording.duration += confPlayer.durationNext;
229     });
230
231     $('#range_player').attr("max", recording.duration);
232     $('#sel_${recording.curEvents)').text("Selected");
233     $('#sel_${recording.curEvents}').addClass("sel_selected");
234     disabledInput(false);
235     setTitle("Ready!");
236     setLoading(false);
237     setStatus("Ready!");
238     setSelectedSaveTime(recording.curEvents);
239 },
240     error: function (error) {
241         console.error(error);
242     }
243 );
244 };
245
246 const playOrStop = (stopRec = isRecPlaying) => {
247     if (stopRec) {
248         // if recording is playing -> stop
249         stop();
250         $('#rec_btn').text("Play");
251     } else {
252         // if recording is not played -> played in the time in input.
253         play($('#range_player').val());
254         $('#rec_btn').text("Stop");
255     }
256
257     isRecPlaying = !stopRec;
258 };
259
260 const play = (time = 0) => {
261     emptyEditor();
262
263     let left = 0;
264     let right = recording.length - 1;
265     let eventIndex = 0;
266
267     // lower bound
268     while (left <= right) {
269         let middle = left + Math.floor((right - left) / 2);
270
271         if (recording.presumIndexDuration[middle] > time) {
272             right = middle - 1;
273         } else {
274             eventIndex = middle;
275             left = middle + 1;
276         }
277     }
278
279     eventIndex = recording.indexEvents[eventIndex];
280     setSelectedSaveTime(eventIndex);
281
282     let timeEvent =
283         time - recording.presumIndexDuration[recording.eventsIndex[eventIndex]];
284     let events = recording.events[eventIndex];
285     left = 0;
286     right = events.length - 1;
287     let eventsIndex = events.length - 1;
288
289     // upper bound
290     while (left <= right) {
291         let middle = left + Math.floor((right - left) / 2);
292
293         if (events[middle].time < timeEvent) {
294             left = middle + 1;
295         } else {
296             eventsIndex = middle;
297             right = middle - 1;
298         }
299     }

```

```

300|
301|     while (
302|       eventsIndex > 0 &&
303|         events[eventsIndex].time - events[eventsIndex - 1].time <=
304|           confPlayer.mergeDuration
305|     ) {
306|       eventsIndex--;
307|     }
308|
309|     handlerForLast = {};
310|
311|     // from index 0 of selected save time to eventsIndex, run all insert and remove
312|     for (let index = 0; index < eventsIndex; index++) {
313|       const event = events[index];
314|       if (handlersIncludeInGoState[event.event]) {
315|         handlers[event.event](event.args);
316|       } else if (events[eventsIndex] - event.time <= confPlayer.mergeDuration) {
317|         handlerForLast[event.event] = () => handlers[event.event](event.args);
318|       }
319|     }
320|
321|     // Run all handler that set to false the last handler that the event is
322|     Object.values(handlerForLast).forEach((f) => {
323|       f();
324|     });
325|
326|     startTimer();
327|
328|     console.log(timeEvent, eventsIndex, events[eventsIndex]);
329|
330|     // Start after time to eventsIndex
331|     if (events[eventsIndex].time < timeEvent)
332|       funcTimeoutRecording = setTimeout(() => {
333|         startRecording(eventsIndex);
334|       }, timeEvent - events[eventsIndex].time);
335|     else
336|       funcTimeoutRecording = setTimeout(() => {
337|         startRecording(eventsIndex);
338|       }, events[eventsIndex].time - timeEvent);
339|   };
340|
341|   // Methods for plays
342|   const startRecording = (index) => {
343|     let events = recording.events[recording.curEvents];
344|     if (index >= events.length) {
345|       if (playNextRecording())
346|         setStatus("Playing_Next_Session");
347|         setTitle("Playing_Next_Session");
348|         return;
349|     } else {
350|       funcTimeoutRecording = setTimeout(() => {
351|         setStatus("Finish...");
352|         setTitle("Finish...");
353|         playOrStop(true);
354|       }, confPlayer.durationNext);
355|       return;
356|     }
357|   }
358|
359|   let event = events[index];
360|   let timeDiff =
361|     (index + 1 < events.length
362|      ? events[index + 1].time
363|      : events[index].time) - event.time;
364|
365|   handlers[event.event](event.args);
366|
367|   while (index + 1 < events.length && timeDiff <= confPlayer.mergeDuration) {
368|     index++;
369|     event = events[index];
370|     timeDiff = events[index + 1].time - event.time;
371|     handlers[event.event](event.args);
372|   }
373|
374|   funcTimeoutRecording = setTimeout(() => {
375|     startRecording(index + 1);
376|   }, timeDiff);
377| };
378|
379| const startTimer = () => {
380|   let offset = 0;
381|
382|   function delta() {
383|     var now = Date.now(),
384|       d = now - offset;
385|
386|     offset = now;
387|     return d;
388|   }
389|
390|   const update = () => {
391|     let val = parseInt($('#range_player').val());
392|     val += delta();
393|     $('#range_player').val(val);
394|     updateTimerRange();
395|   };
396|
397|   if (!funcIntervalTimer) {
398|     offset = Date.now();

```

```

399         funcIntervalTimer = setInterval(update, confTimer.delay);
400     }
401   };
402
403 const playNextRecording = () => {
404   if (recording.eventsIndex[recording.curEvents] < recording.length - 1) {
405     setSelectedSaveTime(
406       recording.indexEvents[recording.eventsIndex[recording.curEvents] + 1],
407       false
408     );
409   }
410   funcTimeoutRecording = setTimeout(() => {
411     emptyEditor();
412
413     funcTimeoutRecording = setTimeout(() => {
414       startRecording(0);
415       }, recording.events[recording.curEvents][0].time);
416     }, confPlayer.durationNext);
417
418     return true;
419   }
420   funcTimeoutRecording = setTimeout(() => {}, confPlayer.durationNext);
421   return false;
422 };
423
424 const stop = () => {
425   stopTimer();
426   stopRecording();
427 };
428
429 // Methods for stop
430 const stopTimer = () => {
431   clearTimeout(funcIntervalTimer);
432   funcIntervalTimer = null;
433 };
434
435 const stopRecording = () => {
436   clearTimeout(funcTimeoutRecording);
437   funcTimeoutRecording = null;
438 };
439
440 // ##### Chart Method #####
441 // #####
442 // #####
443
444 const setUpChart = (
445   index = recording.curEvents,
446   type = confChart.type ? confChart.type : "bar",
447   divider = confChart.divider ? confChart.divider : 1,
448   time = confChart.time ? confChart.time : "S",
449   stack = confChart.stack != undefined ? confChart.stack : true,
450   step = confChart.step != undefined ? confChart.step : false,
451   fill = confChart.fill != undefined ? confChart.fill : false
452 ) => {
453   if (curChart != null) curChart.destroy();
454
455   let times = calcTimeForChart(recording.events[index], divider, time);
456   let data = formatToChartData(recording.events[index], times);
457
458   const ctx = document.getElementById("recording_chart");
459
460   const dataChart = {
461     labels: times.map((c) => c.time),
462     datasets: Object.entries(data).map(([k, v]) => {
463       return {
464         label: nameInChart[k],
465         data: v,
466         fill: fill,
467         stepped: step,
468       };
469     }),
470   };
471
472   const configChart = {
473     type: type,
474     options: {
475       scales: {
476         x: {
477           stacked: stack,
478         },
479         y: {
480           stacked: stack,
481         },
482       },
483     },
484   };
485
486   const chart = new Chart(ctx, {
487     ...configChart,
488     data: dataChart,
489   });
490
491   curChart = chart;
492   return chart;
493 };
494
495 const formatToChartData = (arrEvent, arrTimes) => {
496   let res = {};
497

```

```

498     let idxTimes = 0;
499
500     arrEvent.forEach((e) => {
501         while (e.time > arrTimes[idxTimes].ms) {
502             idxTimes++;
503         }
504
505         if (!res[e.event]) {
506             res[e.event] = {};
507         }
508
509         if (!res[e.event][arrTimes[idxTimes].time]) {
510             res[e.event][arrTimes[idxTimes].time] = 0;
511         }
512
513         res[e.event][arrTimes[idxTimes].time]++;
514     });
515
516     return res;
517 };
518
519 const calcTimeForChart = (arrEvent, divider, time) => {
520     let res = [];
521
522     let dev = calcTimeToMilliSec(divider, time);
523     let max = arrEvent[arrEvent.length - 1].time;
524     let i = 1;
525
526     for (; dev * i < max; i++) {
527         res.push({
528             time: Math.floor(divider * i * 100) / 100 + time,
529             ms: dev * i,
530         });
531     }
532
533     res.push({
534         time: Math.floor(divider * i * 100) / 100 + time,
535         ms: dev * i,
536     });
537
538     return res;
539 };
540
541 const calcTimeToMilliSec = (divider, time) => {
542     if (mult[time]) {
543         return divider * mult[time];
544     }
545
546     return divider;
547 };
548
549 // ##### Set Up Config Chart #####
550 // ###### Listener Config Chart #####
551 // #####
552
553 const setUpTimeDividerSelector = (selectID) => {
554     Object.keys(mult).forEach((c) => {
555         $(`

```



```

696  };
697
698 const setTitle = (title) => {
699   $("#status_rec").text(title);
700 };
701
702 const setSelectedSaveTime = (time, stop = true) => {
703   if (stop) playOrStop(true);
704   $('#sel_${recording.curEvents}').text("Select");
705   $('#sel_${recording.curEvents}').removeClass("sel_selected");
706
707   recording.curEvents = time;
708   $('#sel_${time}').text("Selected");
709   $('#sel_${time}').addClass("sel_selected");
710
711   let duration =
712     recording.events[time][recording.events[time].length - 1].time;
713
714   let divider = (duration / confChart.autoDivider / 1000).toFixed(2);
715   let times = "s";
716
717   $('#config_chart_divider').val(divider);
718   $('#config_chart_time').val(times);
719
720   setUpChart(time, confChart.type, divider, times);
721 };
722
723 const setStatus = (text = "...") => {
724   const status = $("#status_wrapper");
725
726   status.show();
727   status.text(text);
728
729   setTimeout(() => {
730     status.hide();
731   }, 1000);
732 };
733
734 const setLoading = (bool) => {
735   const mainEle = $("#recording_wrap");
736
737   if (bool) {
738     mainEle.hide();
739   } else {
740     mainEle.show();
741   }
742 };
743
744 const setValueTimer = (value) => {
745   $('#range_player').val(value);
746   updateTimerRange();
747 };
748
749 const updateTimerRange = (
750   idRange = "#range_player",
751   idSpan = "#timer_player"
752 ) => {
753   const val = $(idRange).val() / recording.duration * 100;
754   const ms = $(idRange).val();
755
756   $(idSpan).text(convTimeToHHMMSS(ms));
757
758   $(idRange).css(
759     "background",
760     "linear-gradient(to_right, #cc181e 0%, #cc181e " +
761       val +
762       "%, #444 " +
763       val +
764       "%, #444 100%)"
765   );
766 };
767
768 // #####
769 // ##### Runner #####
770 // #####
771
772 getRecording();
773
774 // Runner Config
775 setUpTimeDividerSelector("config_chart_time");
776 });

```

Kode A.10: shj_submit.js

```

1 diff --git a/assets/js/shj_submit.js b/assets/js/shj_submit.js
2 index 7a70670..babaf43 100644
3 --- a/assets/js/shj_submit.js
4 +++ b/assets/js/shj_submit.js
5 @@ -5,181 +5,747 @@
6   * Javascript codes for "Submit" page
7   */
8
9 -$document.ready(function(){
10 -   var editor = ace.edit("code_editor");
11 -   editor.setOptions({
12 -     theme: "ace/theme/monokai",
13 -     fontSize: "11pt"
14 -   });

```

```

15| -     function disableEditor(bool) {
16| -         $("#editor_save").prop("disabled", bool);
17| -         $("#editor_execute").prop("disabled", bool);
18| -         $("#editor_submit").prop("disabled", bool);
19| -         $("#editor_input").prop("disabled", bool);
20| -         editor.setReadOnly(bool);
21| -     }
22| -
23| -
24| -     function loadCode(problem_id){
25| -         $("#editor_input").val("");
26| -         $("#editor_output").val("");
27| -
28| -         if(problem_id == 0){
29| -             disableEditor(true);
30| -             editor.setValue("");
31| -             $("#ajax_status").html("Select problem and language");
32| -         } else{
33| -             disableEditor(true);
34| -             $.ajax({
35| -                 url: shj.site_url + 'submit/load/' + problem_id,
36| -                 cache: false,
37| -                 success: function (data){
38| -                     data = JSON.parse(data);
39| -                     editor.setValue(data.content);
40| -                     $("#ajax_status").html(data.message);
41| -                 },
42| -                 error: function (error){
43| -                     console.error(error);
44| -                 },
45| -             });
46| -         }
47| -     }
48| -
49| -
50| -     $("select#problems").change(function(){
51| -         var v = $(this).val();
52| -         loadCode(v);
53| -         $('select#languages').empty();
54| -         $(<option value="0" selected="selected">-- Select Language --</option>').appendTo('select#languages');
55| -         for (var i=0;i<shj.p[v].length;i++)
56| -             $(<option value="'+shj.p[v][i]+'>'+'shj.p[v][i]'+</option>').appendTo('select#languages');
57| -     });
58| -
59| -     $("select#languages").change(function(){
60| -         if(this.value.toLowerCase().includes("java")){
61| -             editor.session.setMode("ace/mode/java");
62| -             disableEditor(false);
63| -         }
64| -         else if(this.value.toLowerCase().includes("python")){
65| -             editor.session.setMode("ace/mode/python");
66| -             disableEditor(false);
67| -         }
68| -         else if(this.value.toLowerCase().includes("c")){
69| -             editor.session.setMode("ace/mode/c_cpp");
70| -             disableEditor(false);
71| -         }
72| -         else if(this.value.toLowerCase().includes("txt")){
73| -             editor.session.setMode("ace/mode/plain_text");
74| -             disableEditor(false);
75| -             $("#editor_execute").prop("disabled", true);
76| -             $("#editor_input").prop("disabled", true);
77| -         }
78| -         else{
79| -             editor.session.setMode("ace/mode/plain_text");
80| -             disableEditor(true);
81| -         }
82| -     });
83| -
84| -     $("#editor_save").click(function(){
85| -         disableEditor(true);
86| -         $.ajax({
87| -             type: "POST",
88| -             url: shj.site_url + 'submit/save',
89| -             data: {
90| -                 shj_csrf_token: shj.csrf_token,
91| -                 code_editor: editor.getValue(),
92| -                 problem_id: $("select#problems").val(),
93| -                 language: $("select#languages").val(),
94| -             },
95| -             cache: false,
96| -             success: function(data){
97| -                 data = JSON.parse(data);
98| -                 $("#ajax_status").html(data.message);
99| -                 disableEditor(false);
100| -             },
101| -             error: function (error){
102| -                 console.error(error);
103| -                 disableEditor(false);
104| -             },
105| -         });
106| -     });
107| -
108| -     $("#editor_submit").click(function(){
109| -         disableEditor(true);
110| -         $.ajax({
111| -             type: "POST",
112| -             url: shj.site_url + 'submit/save/submit',
113| -             data: {

```

```

114| -         shj.csrf_token: shj.csrf_token,
115| -         code_editor: editor.getValue(),
116| -         problem_id: $("#"select#problems").val(),
117| -         language: $("select#languages").val(),
118| },
119| -         cache: false,
120| -         success: function(data){
121| -             data = JSON.parse(data);
122| -             $("#ajax_status").html(data.message);
123| -             disableEditor(false);
124| -             if(data.status){
125| -                 window.location.href = shj.site_url + 'submissions/all';
126| -             }
127| -         },
128| -         error: function (error){
129| -             console.error(error);
130| -             disableEditor(true);
131| -         },
132| });
133| });
134|
135| $("#editor_execute").click(function(){
136|     disableEditor(true);
137|     $.ajax({
138|         type: "POST",
139|         url: shj.site_url + 'submit/save/execute',
140|         data: {
141|             shj.csrf_token: shj.csrf_token,
142|             code_editor: editor.getValue(),
143|             editor_input: $('#textareaeeditor_input').val(),
144|             problem_id: $("#"select#problems").val(),
145|             language: $("select#languages").val(),
146|         },
147|         cache: false,
148|         success: function(data){
149|             data = JSON.parse(data);
150|             $("#ajax_status").html(data.message);
151|             if(data.status){
152|                 (function update() {
153|                     $.ajax({
154|                         url: shj.site_url + 'submit/get_output/' + $("#"select#problems").val(),
155|                         cache: false,
156|                         success: function (data){
157|                             data = JSON.parse(data);
158|                             $('#textareaeeditor_output').val(data.content);
159|                             if(!data.status){
160|                                 setTimeout(update, 1000);
161|                             }
162|                             else{
163|                                 $("#ajax_status").html("Completed");
164|                                 disableEditor(false);
165|                             }
166|                         },
167|                         error: function (error){
168|                             console.error(error);
169|                             disableEditor(true);
170|                         },
171|                     })
172|                 })();
173|             }
174|             else{
175|                 disableEditor(false);
176|             }
177|         },
178|         error: function (error){
179|             console.error(error);
180|             disableEditor(true);
181|         },
182|     });
183| });
184|
185| loadCode($("#select#problems").val());
186| });
187| \ No newline at end of file
188| +$(document).ready(function () {
189| +    var editor = ace.edit("code_editor");
190| +
191| +    editor.setOptions({
192| +        theme: "ace/theme/monokai",
193| +        fontSize: "11pt",
194| +        enableLiveAutocompletion: true,
195| +        enableBasicAutocompletion: true,
196| +        enableSnippets: true,
197| +    });
198| +
199| +    function disableEditor(bool) {
200| +        $("#editor_save").prop("disabled", bool);
201| +        $("#editor_execute").prop("disabled", bool);
202| +        $("#editor_submit").prop("disabled", bool);
203| +        $("#editor_input").prop("disabled", bool);
204| +        editor.setReadOnly(bool);
205| +    }
206| +
207| +    function loadCode(problem_id) {
208| +        $("#editor_input").val("");
209| +        $("#editor_output").val("");
210| +
211| +        if (problem_id == 0) {
212| +            disableEditor(true);

```



```

312+    // let rec = { ...recording };
313+    // rec.reset = "";
314+
315+    console.log(getCurrentTime());
316+
317+    // let fd = new FormData();
318+    // fd.append("shj_csrf_token", shj.csrf_token);
319+    // fd.append("code_editor", editor.getValue());
320+    // fd.append("problem_id", $("select#problems").val());
321+    // fd.append("language", $("select#languages").val());
322+    // fd.append("rec", JSON.stringify(rec));
323+    // fd.append('buffer', blob, 'rec.bin');
324+
325+    $.ajax({
326+        type: "POST",
327+        url: shj.site_url + "submit/save",
328+        data: {
329+            shj.csrf_token: shj.csrf_token,
330+            code_editor: editor.getValue(),
331+            problem_id: $("select#problems").val(),
332+            language: $("select#languages").val(),
333+            rec_data: JSON.stringify({
334+                ...befRecording,
335+                [recording.startTime]: recording.events
336+            }),
337+            // rec_start: convTimeToEpoch(recording.startTime),
338+            // rec_end: convTimeToEpoch(recording.startTime + recording.events[recording.events.length - 1].time),
339+        },
340+        // processData: false,
341+        // contentType: false,
342+        cache: false,
343+        success: function (data) {
344+            data = JSON.parse(data);
345+
346+            // console.log(JSON.parse(data["test"]));
347+            // console.log(getCurrentTime());
348+            // console.log(data['test']);
349+
350+            $("#ajax_status").html(data.message);
351+            disableEditor(false);
352+        },
353+        error: function (error) {
354+            console.error(error);
355+            disableEditor(false);
356+        },
357+    });
358+});
359+
360+$("#editor_submit").click(function () {
361+    disableEditor(true);
362+    handlers.submit();
363+    // let rec = { ...recording };
364+
365+    $.ajax({
366+        type: "POST",
367+        url: shj.site_url + "submit/save/submit",
368+        data: {
369+            shj.csrf_token: shj.csrf_token,
370+            code_editor: editor.getValue(),
371+            problem_id: $("select#problems").val(),
372+            language: $("select#languages").val(),
373+            rec_data: JSON.stringify({
374+                ...befRecording,
375+                [recording.startTime]: recording.events
376+            }),
377+            // rec_data: JSON.stringify(rec),
378+            // rec_start: convTimeToEpoch(rec.startTime),
379+            // rec_end: convTimeToEpoch(rec.startTime + rec.events[rec.events.length - 1].time),
380+        },
381+        cache: false,
382+        success: function (data) {
383+            data = JSON.parse(data);
384+            $("#ajax_status").html(data.message);
385+            disableEditor(false);
386+            if (data.status) {
387+                window.location.href = shj.site_url + "submissions/all";
388+            }
389+        },
390+        error: function (error) {
391+            console.error(error);
392+            disableEditor(false);
393+        },
394+    });
395+
396+    recordStop();
397+});
398+
399+$("#editor_execute").click(function () {
400+    disableEditor(true);
401+    handlers.execute();
402+    // let rec = { ...recording };
403+
404+    $.ajax({
405+        type: "POST",
406+        url: shj.site_url + "submit/save/execute",
407+        data: {
408+            shj.csrf_token: shj.csrf_token,
409+            code_editor: editor.getValue(),
410+            editor_input: $("textarea#editor_input").val(),

```

```

411| +         problem_id: $("select#problems").val(),
412| +         language: $("select#languages").val(),
413| +         rec_data: JSON.stringify({
414| +             ...befRecording,
415| +             [recording.startTime]: recording.events
416| +         }),
417| +         // rec_data: JSON.stringify(rec),
418| +         // rec_start: convTimeToEpoch(rec.startTime),
419| +         // rec_end: convTimeToEpoch(rec.startTime + rec.events[rec.events.length - 1].time),
420| +     },
421| +     cache: false,
422| +     success: function (data) {
423| +         data = JSON.parse(data);
424| +         $("#ajax_status").html(data.message);
425| +         if (data.status) {
426| +             (function update() {
427| +                 $.ajax({
428| +                     url:
429| +                         shj.site_url +
430| +                         "submit/get_output/" +
431| +                         $("select#problems").val(),
432| +                     cache: false,
433| +                     success: function (data) {
434| +                         data = JSON.parse(data);
435| +                         $("#textarea#editor_output").val(data.content);
436| +                         // ----
437| +                         $("#textarea#editor_output").trigger(
438| +                             "output_change",
439| +                             data.content
440| +                         );
441| +                         // ----
442| +                         if (!data.status) {
443| +                             setTimeout(update, 1000);
444| +                         } else {
445| +                             $("#ajax_status").html("Completed");
446| +                             disableEditor(false);
447| +                         }
448| +                     },
449| +                     error: function (error) {
450| +                         console.error(error);
451| +                         disableEditor(false);
452| +                     },
453| +                     });
454| +                 })();
455| +             } else {
456| +                 disableEditor(false);
457| +             }
458| +         },
459| +         error: function (error) {
460| +             console.error(error);
461| +             disableEditor(false);
462| +         },
463| +     });
464| + });
465| +
466| + loadCode($(".select#problems").val());
467| +
468| + // ##### Recording #####
469| + // Local Variable
470| + const Range = ace.Range;
471| +
472| + // Local Variable
473| + const Range = ace.Range;
474| +
475| + let hidden = "hidden";
476| + let visibilityChange = "visibilitychange";
477| + let isRetrieved = false;
478| +
479| + // Saved Recording from before...
480| + let befRecording = {};
481| +
482| + // Saved Event
483| + const recording = {
484| +     events: [],
485| +     startTime: -1,
486| +     // startValue: "",
487| +     // startSelection: [],
488| +
489| +     init: () => {
490| +         recording.events = [];
491| +         recording.startTime = Date.now();
492| +         // recording.startValue = editor.getValue();
493| +         // recording.startSelection = getSelection(editor);
494| +     },
495| + };
496| +
497| + // What's recording does the system will record
498| + const include = {
499| +     editor: true,
500| +     web: true,
501| +     pdf: true,
502| +     input: true,
503| +     output: true,
504| +     // action: true, // always true
505| +     others: false, // default value for other recording
506| + };
507| +
508| + // #####
509| + // #####

```

```

510+ // ###### Editor Event #####
511+
512+ const handlers = {
513+   // ##### Editor Event #####
514+   // Detected Every Command that executed in editor
515+   editor_change: (e) =>
516+     recordEvent(e.action, {
517+       data: e.lines,
518+       start: e.start,
519+       end: e.end,
520+     }),
521+   // Detected on cursor change
522+   editor_cursor: () => recordEvent("cursor_selection", getSelection(editor)),
523+   // Detected on selection
524+   editor_selection: () => recordEvent("sel_selection", getSelection(editor)),
525+
526+   // ##### Windows Event #####
527+   // Detected Leaving Focus in almost all browser (still active page, but on different windows or on iFrame PDF viewer)
528+   focus: () => {
529+     removeListener.focus();
530+     addListener.blur();
531+
532+     recordEvent("focus");
533+   },
534+   // Detected on Focus in almost all browser (active page)
535+   blur: () => {
536+     removeListener.blur();
537+     addListener.focus();
538+
539+     recordEvent("blur");
540+   },
541+   // Detected Leaving Page in almost all browser (page not visible anymore)
542+   visibility: (evt) => {
543+     let v = true; // page is visible
544+     let h = false; // page is hidden
545+
546+     let evtMap = {
547+       focus: v,
548+       focusin: v,
549+       pageshow: v,
550+       blur: h,
551+       focusout: h,
552+       pagehide: h,
553+     };
554+
555+     let isVisible = true;
556+
557+     evt = evt || window.event;
558+
559+     if (evt.type in evtMap) isVisible = evtMap[evt.type];
560+     else isVisible = document[hidden] ? h : v;
561+
562+     if (isVisible) {
563+       // detect focus or blur if visible again
564+       addListener.focus();
565+       addListener.pdf_focus();
566+     } else {
567+       // no need to detect focus or blur if not visible
568+       removeListener.focus();
569+       removeListener.blur();
570+       removeListener.pdf_focus();
571+       removeListener.pdf.blur();
572+     }
573+
574+     // console.log("change visibility:", !document[hidden], getCurrentTime());
575+     recordEvent("visibility", isVisible);
576+   },
577+
578+   // ##### PDF Viewer #####
579+   // Detected when user click/focus on the pdf viewer IDE
580+   pdf_focus: () => {
581+     removeListener.pdf_focus();
582+     addListener.pdf.blur();
583+
584+     recordEvent("pdf_focus");
585+   },
586+   // Detected when user click outside/blur of the pdf viewer IDE
587+   pdf.blur: () => {
588+     removeListener.pdf.blur();
589+     addListener.pdf_focus();
590+
591+     recordEvent("pdf.blur");
592+   },
593+
594+   // ##### Input Event #####
595+   input_change: (e) => recordEvent("input_change", e.currentTarget.value),
596+
597+   // ##### Output Event #####
598+   output_change: (_, data) => recordEvent("output_change", data),
599+
600+   // ##### Action Event #####
601+   save: () => recordEvent("save"),
602+   submit: () => recordEvent("submit"),
603+   execute: () => recordEvent("execute"),
604+ };
605+
606+ const addListener = {
607+   editor_change: () => editor.session.on("change", handlers.editor_change),
608+   editor_cursor: () =>

```

```

609+     editor.session.selection.on("changeCursor", handlers.editor_cursor),
610+     editor.session.selection.on("changeSelection", handlers.editor_selection),
611+     focus: () => addEvent(window, "focus", handlers.focus),
612+     blur: () => addEvent(window, "blur", handlers.blur),
613+     visibility: () => addEvent(document, visibilityChange, handlers.visibility),
614+     pdf_focus: () =>
615+       addEvent(
616+         $("#pdf_viewer")[0].contentWindow,
617+         "focusin",
618+         handlers.pdf_focus
619+       ),
620+     pdf_blur: () =>
621+       addEvent(
622+         $("#pdf_viewer")[0].contentWindow,
623+         "focusout",
624+         handlers.pdf_blur
625+       ),
626+     input_change: () => $("#editor_input").on("input", handlers.input_change),
627+     output_change: () =>
628+       $("#textarea#editor_output").on("output_change", handlers.output_change),
629+     // save: () => $("#editor_save").on("click", handlers.save),
630+     // submit: () => $("#editor_submit").on("click", handlers.submit),
631+     // execute: () => $("#editor_execute").on("click", handlers.execute),
632+   );
633+ };
634+
635+ const removeListener = {
636+   editor_change: () => editor.commands.off("afterExec", handlers.editor_change),
637+   editor_cursor: () =>
638+     editor.selection.off("changeCursor", handlers.editor_cursor),
639+   editor_selection: () =>
640+     editor.selection.off("changeSelection", handlers.editor_selection),
641+   focus: () => removeEvent(window, "focus", handlers.focus),
642+   blur: () => removeEvent(window, "blur", handlers.blur),
643+   visibility: () =>
644+     removeEvent(document, visibilityChange, handlers.visibility),
645+   pdf_focus: () =>
646+     removeEvent(
647+       $("#pdf_viewer")[0].contentWindow,
648+       "focusin",
649+       handlers.pdf_focus
650+     ),
651+   pdf_blur: () =>
652+     removeEvent(
653+       $("#pdf_viewer")[0].contentWindow,
654+       "focusout",
655+       handlers.pdf_blur
656+     ),
657+   input_change: () => $("#editor_input").off("input", handlers.input_change),
658+   output_change: () =>
659+     $("#textarea#editor_output").off("output_change", handlers.output_change),
660+   // save: () => $("#editor_save").off("click", handlers.save),
661+   // submit: () => $("#editor_submit").off("click", handlers.submit),
662+   // execute: () => $("#editor_execute").off("click", handlers.execute),
663+ };
664+
665+ // ##### Methods #####
666+ // #####
667+ // #####
668+
669+ const record = {
670+   // Code Editor
671+   editor: () => {
672+     // ##### Editor #####
673+     recording.startValue = editor.getValue();
674+     recording.startSelection = getSelection(editor);
675+
676+     // Exec command
677+     addListener.editor_change();
678+
679+     // For Cursor
680+     addListener.editor_cursor();
681+     addListener.editor_selection();
682+   },
683+   // Overall page/tabs
684+   web: () => {
685+     // ##### Web Page #####
686+     // Get from https://stackoverflow.com/questions/1060008/is-there-a-way-to-detect-if-a-browser-window-is-not-currently-
active
687+
688+     // for every type of browser.
689+     if (hidden in document) {
690+       visibilityChange = "visibilitychange";
691+     } else if ((hidden = "mozHidden") in document) {
692+       visibilityChange = "mozvisibilitychange";
693+     } else if ((hidden = "webkitHidden") in document) {
694+       visibilityChange = "webkitvisibilitychange";
695+     } else if ((hidden = "msHidden") in document) {
696+       visibilityChange = "msvisibilitychange";
697+     }
698+
699+     if (visibilityChange != null) {
700+       // addListener.focus();
701+       addListener.blur();
702+       addListener.visibility();
703+     } else if ("onfocusin" in document) {
704+       // IE 9 and lower:
705+       document.onfocusin = document.onfocusout = handlers.visibility;
706+     } else {

```

```

707+           // All others:
708+           window.onpageshow =
709+             window.onpagehide =
710+             window.onfocus =
711+             window.onblur =
712+               handlers.visibility;
713+         }
714+       },
715+     // PDF Viewer
716+     pdf: () => {
717+       let el = document.getElementById("pdf_viewer");
718+
719+       var observer = new IntersectionObserver(function () {
720+         if (el.src != "") {
721+           // addlistener.pdf_blur();
722+           addListener.pdf_focus();
723+         }
724+       });
725+
726+       observer.observe(el, { attributes: true, childList: true });
727+     },
728+     // Input field
729+     input: () => {
730+       addListener.input_change();
731+     },
732+     // Output field
733+     output: () => {
734+       addListener.output_change();
735+     },
736+     // Action Button added in the onclick event listener.
737+   };
738+
739+ // Methods to start recording.
740+ const recordStart = () => {
741+   recording.init();
742+
743+   Object.keys(record).forEach((evtName) => {
744+     const inInclude = evtName in include;
745+     if (include[evtName] || (!inInclude && include["others"])) {
746+       record[evtName]();
747+     }
748+   });
749+ };
750+
751+ const recordStop = () => {
752+   Object.values(removeListener).forEach((func) => {
753+     func();
754+   });
755+ };
756+
757+ // ##### Misc #####
758+ // #####
759+ // #####
760+
761+ // Method to record listener.
762+ const recordEvent = (event, args) => {
763+   let curTime = getCurrentTime();
764+
765+   recording.events.push({
766+     time: curTime,
767+     event,
768+     args,
769+   });
770+
771+   console.log(curTime + "ms", event, args);
772+ };
773+
774+ const addEvent = (obj, evType, fn, isCapturing) => {
775+   if (isCapturing == null) isCapturing = false;
776+   if (obj.addEventListener) {
777+     // Firefox
778+     obj.addEventListener(evType, fn, isCapturing);
779+     return true;
780+   } else if (obj.attachEvent) {
781+     // MSIE
782+     var r = obj.attachEvent("on" + evType, fn);
783+     return r;
784+   } else {
785+     return false;
786+   }
787+ };
788+
789+ const removeEvent = (obj, evType, fn, isCapturing) => {
790+   if (isCapturing == null) isCapturing = false;
791+   if (obj.removeEventListener) {
792+     // Firefox
793+     obj.removeEventListener(evType, fn, isCapturing);
794+     return true;
795+   } else if (obj.detachEvent) {
796+     // MSIE
797+     var r = obj.detachEvent("on" + evType, fn);
798+     return r;
799+   } else {
800+     return false;
801+   }
802+ };
803+
804+ const getSelection = (editor) => {
805+   var data = editor.multiSelect.toJSON();

```

```

806 +     if (!data.length) data = [data];
807 +     data = data.map(function (x) {
808 +       var a, c;
809 +       if (x.isBackwards) {
810 +         a = x.end;
811 +         c = x.start;
812 +       } else {
813 +         c = x.end;
814 +         a = x.start;
815 +       }
816 +       return Range.comparePoints(a, c)
817 +         ? [a.row, a.column, c.row, c.column]
818 +         : [a.row, a.column];
819 +     });
820 +   return data.length > 1 ? data : data[0];
821 + };
822 +
823 + const getCurrentTime = () => {
824 +   return Date.now() - recording.startTime;
825 + };

```

Kode A.11: side_bar.twig

```

1 diff --git a/application/views/templates/side_bar.twig b/application/views/templates/side_bar.twig
2 index ce5a0e2..857c6d3 100644
3 --- a/application/views/templates/side_bar.twig
4 +++ b/application/views/templates/side_bar.twig
5 @@ -12,18 +12,18 @@
6   </a>
7   </li>
8   {% if user.level == 3 %}
9 -   <li class="color-settings{{ selected=='settings' ? ' selected' }}">
10 -     <a href="{{ site_url('settings') }}" aria-labelledby="settings-label">
11 -       <i class="fa fa-gear fa-lg"></i>
12 -       <span class="sidebar_text" id="settings-label">Settings</span>
13 -     </a>
14 -   </li>
15 -   <li class="color-users{{ selected=='users' ? ' selected' }}">
16 -     <a href="{{ site_url('users') }}" aria-labelledby="users-label">
17 -       <i class="fa fa-users fa-lg"></i>
18 -       <span class="sidebar_text" id="users-label">Users</span>
19 -     </a>
20 -   </li>
21 +   <li class="color-settings{{ selected=='settings' ? ' selected' }}">
22 +     <a href="{{ site_url('settings') }}" aria-labelledby="settings-label">
23 +       <i class="fa fa-gear fa-lg"></i>
24 +       <span class="sidebar_text" id="settings-label">Settings</span>
25 +     </a>
26 +   </li>
27 +   <li class="color-users{{ selected=='users' ? ' selected' }}">
28 +     <a href="{{ site_url('users') }}" aria-labelledby="users-label">
29 +       <i class="fa fa-users fa-lg"></i>
30 +       <span class="sidebar_text" id="users-label">Users</span>
31 +     </a>
32 +   </li>
33   {% endif %}
34   <li class="color-notifications{{ selected=='notifications' ? ' selected' }}">
35     <a href="{{ site_url('notifications') }}" aria-labelledby="notifications-label">
36 @@ -49,6 +49,14 @@
37     <span class="sidebar_text" id="submit-label">Submit</span>
38   </a>
39   </li>
40 +   {% if user.level > 1 %}
41 +     <li class="color-recording{{ selected=='recording' ? ' selected' }}">
42 +       <a href="{{ site_url('recording/all') }}" aria-labelledby="recording-label">
43 +         <i class="fa fa-video-camera fa-lg"></i>
44 +         <span class="sidebar_text" id="recording-label">Recording</span>
45 +       </a>
46 +     </li>
47 +   {% endif %}
48   <li class="color-final_submissions{{ selected=='final_submissions' ? ' selected' }}">
49     <a href="{{ site_url('submissions/final') }}" aria-labelledby="final-submission-label">
50       <i class="fa fa-map-marker fa-lg"></i>
51 @@ -67,29 +75,31 @@
52     <span class="sidebar_text" id="scoreboard-label">Scoreboard</span>
53   </a>
54   </li>
55 -   <li class="color-halloffame{{ selected=='halloffame' ? ' selected' }}">
56 +   <li class="color-halloffame{{ selected=='halloffame' ? ' selected' }}">
57     <a href="{{ site.url('halloffame') }}" aria-labelledby="hall-of-fame-label">
58       <i class="fa fa-list-alt fa-lg"></i>
59       <span class="sidebar_text" id="hall-of-fame-label">Hall of Fame</span>
60     </a>
61   </li>
62   {% if user.level == 3 %}
63   <li class="color-logs{{ selected=='logs' ? ' selected' }}">
64     <a href="{{ site.url('logs') }}" aria-labelledby="24-hour-log-label">
65       <i class="fa fa-book fa-lg"></i>
66       <span class="sidebar_text" id="24-hour-log-label">24-hour Log</span>
67     </a>
68   </li>
69   {% endif %}
70 +   {% if user.level == 3 %}
71 +     <li class="color-logs{{ selected=='logs' ? ' selected' }}">
72 +       <a href="{{ site.url('logs') }}" aria-labelledby="24-hour-log-label">
73 +         <i class="fa fa-book fa-lg"></i>
74 +         <span class="sidebar_text" id="24-hour-log-label">24-hour Log</span>
75 +     </a>

```

```

76+           </li>
77+           {% endif %}
78</ul>
79<div id="sidebar_bottom">
80    <p>
81      <a href="https://github.com/ifunpar/Sharif-Judge" target="_blank">&copy; SharIF Judge {{ SHJ_VERSION }}</a>
82      <a href="https://github.com/ifunpar/Sharif-Judge" target="_blank">&copy; SharIF Judge
83      {{ SHJ_VERSION }}</a>
84      <a href="https://github.com/ifunpar/Sharif-Judge/tree/docs" target="_blank">Docs</a>
85    </p>
86    <p class="timer"></p>
87    <div id="shj Collapse" class="pointer">
88      <i id="collapse" class="fa fa-caret-square-o-left fa-lg"></i><span class="sidebar_text">Collapse Menu</span>
89      <i id="collapse" class="fa fa-caret-square-o-left fa-lg"></i>
90      <span class="sidebar_text">Collapse Menu</span>
91    </div>
92  </div>
93</div>

```

Kode A.12: Submit.php

```

1 diff --git a/application/controllers/Submit.php b/application/controllers/Submit.php
2 index 8517271..d40750a 100644
3 --- a/application/controllers/Submit.php
4 +++ b/application/controllers/Submit.php
5 @@ -270,16 +270,42 @@ class Submit extends CI_Controller
6     if(!write_file($input_path, '')){}
7     if(!write_file($output_path, '')){
8     if (!file_exists($file_path)){
9       $response = json_encode(array(content=>'', message=>'No saved file'));
10 +    $response = json_encode(array('content'=>'', 'message'=>'No saved file'));
11   }
12   else{
13     $file_content = file_get_contents($file_path);
14     if ($file_content === FALSE){
15       $response = json_encode(array(content=>'', message=>'Unable to load'));
16 +      $response = json_encode(array('content'=>'', 'message'=>'Unable to load'));
17     }
18     else{
19       addslashes($file_content);
20       $response = json_encode(array(content=>$file_content, message=>'Loaded'));
21 +      $response = json_encode(array('content'=>$file_content, 'message'=>'Loaded'));
22     }
23   }
24   echo $response;
25 }
26 //
27 +
28 +
29 /**
30 * Load recording in files from recording file
31 */
32 public function load_rec($problem_id) {
33   $user_dir = rtrim($this->assignment_root, '/').'/assignment_.'.$this->user->selected_assignment['id'].'/p'.$problem_id.'/';
34   $this->user->username;
35   $file_path = $user_dir.'/'.$RECORD_FILE_NAME.'.'.$RECORD_FILE_EXT;
36   $this->load->helper('file');
37   if (!file_exists($file_path)){
38     $response = json_encode(array('content'=>'', 'message'=>'No recording file'));
39   }
40   else{
41     $file_content = file_get_contents($file_path);
42     if ($file_content === FALSE){
43       $response = json_encode(array('content'=>'', 'message'=>'Unable to load'));
44     }
45     else{
46       addslashes($file_content);
47       $response = json_encode(array('content'=>$file_content, 'message'=>'Loaded'));
48     }
49   }
50   echo $response;
51 @@ -295,6 +321,10 @@ class Submit extends CI_Controller
52   $data = $_POST['code_editor'];
53   $problem_id = $_POST['problem_id'];
54   $language = $_POST['language'];
55 + $rec = $_POST['rec_data'];
56
57   $user_dir = rtrim($this->assignment_root, '/').'/assignment_.'.$this->user->selected_assignment['id'].'/p'.$problem_id.'/';
58   $this->user->username;
59   if (!file_exists($user_dir)){
59 @@ -302,47 +332,76 @@ class Submit extends CI_Controller
60   }
61   $file_path = $user_dir.'/'.$EDITOR_FILE_NAME.'.'.$EDITOR_FILE_EXT;
62   $input_path = $user_dir.'/'.$EDITOR_IN_NAME.'.'.$EDITOR_FILE_EXT;
63 +
64 + $rec_path = $user_dir.'/'.$RECORD_FILE_NAME.'.'.$RECORD_FILE_EXT;
65
66   $this->load->helper('file');
67   if (!write_file($file_path, $data)){
68     $response = json_encode(array(status=>FALSE, message=>'Unable to save'));
69   if (!(write_file($file_path, $data) && write_file($rec_path, $rec))){
70 +    $response = json_encode(array('status'=>FALSE, 'message'=>'Unable to save'));
71     echo $response;
72   }
73   else{
74     $response = json_encode(array(status=>TRUE, message=>'Saved'));
75     if($type === FALSE){

```

```

76+
77+     $response = json_encode(array('status'=>TRUE, 'message'=>'Saved'));
78+
79+     $this->load->model('recording_model');
80+     $this->recording_model->add_recording(array(
81+         'rec_id'          => 0,
82+         'username'        => $this->user->username,
83+         'assignment'      => $this->user->selected_assignment['id'],
84+         'problem'         => $problem_id,
85+         'upload_at'       => shj_now_str(),
86+     ));
87+
88+     if($type === FALSE){ // If only saved
89+         echo $response;
90+     }
91- else{
92+     else{ // If want to execute/submit
93+         $now = shj_now();
94+         if ( $this->queue_model->in_queue($this->user->username,$this->user->selected_assignment['id'], $this->problem['id']")){
95-             $response = json_encode(array(status=>FALSE, message=>'You have already submitted for this problem. Your last
96+             submission is still in queue.'));
96+             $response = json_encode(array('status'=>FALSE, 'message'=>'You have already submitted for this problem. Your
97             last submission is still in queue.'));
98+             echo $response;
99+         }
100-        else if ($this->user->level==0 && !$this->user->selected_assignment['open']){
101+            $response = json_encode(array(status=>FALSE, message=>'Selected assignment has been closed.'));
102+            $response = json_encode(array('status'=>FALSE, 'message'=>'Selected assignment has been closed.'));
103+            echo $response;
104+        }
105-        else if ($now < strtotime($this->user->selected_assignment['start_time'])){
106+            $response = json_encode(array(status=>FALSE, message=>'Selected assignment has not started.'));
107+            $response = json_encode(array('status'=>FALSE, 'message'=>'Selected assignment has not started.'));
108+            echo $response;
109+        }
110-        else if ($now > strtotime($this->user->selected_assignment['finish_time'])+$this->user->selected_assignment['
111+            extra_time']){
112+            $response = json_encode(array(status=>FALSE, message=>'Selected assignment has finished.'));
113+            $response = json_encode(array('status'=>FALSE, 'message'=>'Selected assignment has finished.'));
114+            echo $response;
115-        else if ( ! $this->assignment_model->is_participant($this->user->selected_assignment['participants'], $this->user->
116+            username)){
117+            $response = json_encode(array(status=>FALSE, message=>'You are not registered for submitting.'));
118+            $response = json_encode(array('status'=>FALSE, 'message'=>'You are not registered for submitting.'));
119+            echo $response;
120+        }
121-    else{
122+        if($type === 'submit'){
123+            $this->submit($data, $problem_id, $language, $user_dir);
124+            $this->_submit($data, $problem_id, $language, $user_dir, $rec);
125+        }
126-        else if($type === 'execute'){
127+            $editor_input = $_POST['editor_input'];
128+            if (!write_file($input_path, $editor_input)){
129+                $response = json_encode(array(status=>FALSE, message=>'Unable to write input file'));
130+                $response = json_encode(array('status'=>FALSE, 'message'=>'Unable to write input file'));
131+                echo $response;
132-    @@ -362,13 +421,19 @@ class Submit extends CI_Controller
133+    /**
134+     * Add code to queue for judging
135+     */
136-    private function _submit($data, $problem_id, $language, $user_dir){
137+    private function _submit($data, $problem_id, $language, $user_dir, $rec){
138+        $file_type = $this->language_to_type(strtolower(trim($language)));
139+        $file_ext = $this->language_to_ext(strtolower(trim($language)));
140+        $file_name = EDITOR_FILE_NAME;
141+        $file_fname = $file_name.'.'.$this->user->selected_assignment['total_submits']+1;
142+        $file_path = $user_dir.'/'.$file_fname.'.'.$file_ext;
143+
144+        $rec_file_name = RECORD_FILE_NAME;
145+        $rec_file_fname = $rec_file_name.'-'.$this->user->selected_assignment['total_submits']+1;
146+        $rec_file_path = $user_dir.'/'.$rec_file_fname.'.RECORD_FILE_EXT';
147+
148+        $old_file_path = $user_dir.'/'.$RECORD_FILE_NAME.'.'.$RECORD_FILE_EXT;
149+
150+        foreach($this->problems as $item)
151+            if ($item['id'] == $problem_id)
152+            {
153-    @@ -376,8 +441,8 @@ class Submit extends CI_Controller
154+        break;
155+    }
156-
157-        if (!write_file($file_path, $data)){
158-            $response = json_encode(array(status=>FALSE, message=>'Unable to submit'));
159+        if (!write_file($file_path, $data) && rename($old_file_path, $rec_file_path)){
160+            $response = json_encode(array('status'=>FALSE, 'message'=>'Unable to submit'));
161+        }
162+    else{
163+        $this->load->model('submit_model');
164-    @@ -394,6 +459,17 @@ class Submit extends CI_Controller
165+        'pre_score' => 0,
166+        'time' => shj_now_str(),
167+    );
168+    $this->load->model('recording_model');
169+

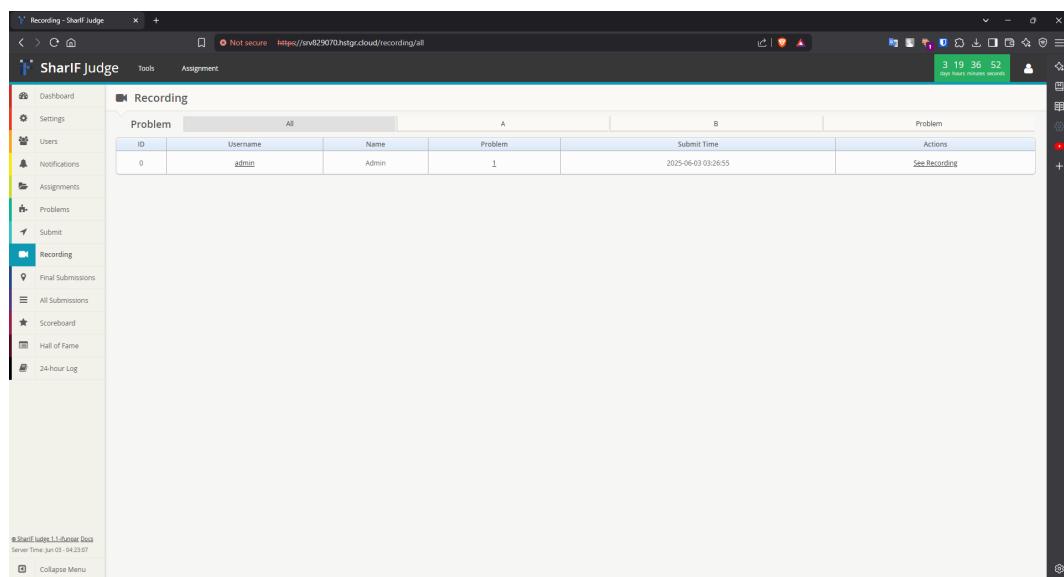
```

```

170+     $this->recording_model->add_recording(array(
171+         'rec_id'      => $submit_info['submit_id'],
172+         'username'    => $submit_info['username'],
173+         'assignment'  => $submit_info['assignment'],
174+         'problem'     => $submit_info['problem'],
175+         'upload_at'   => shj_now_str(),
176+     ));
177+     $this->recording_model->remove_saveonly_recording($submit_info['assignment'], $submit_info['problem'], $submit_info['username']);
178+
179     if ($this->problem['is_upload_only'] == 0)
180     {
181         $this->queue_model->add_to_queue($submit_info);
182 @@ -404,12 +480,11 @@ class Submit extends CI_Controller
183         $this->submit_model->add_upload_only($submit_info);
184     }
185-
186     $response = json_encode(array(status=>TRUE, message=>"Submitted"));
187+     $response = json_encode(array('status'=>TRUE, 'message'=>"Submitted"));
188 }
189 echo $response;
190 }
191-
192 /**
193 */
194 /**
195 @@ -424,7 +499,7 @@ class Submit extends CI_Controller
196     $output_path = $user_dir.'/'.$EDITOR_OUT_NAME.'.'.$EDITOR_FILE_EXT;
197-
198     if (!write_file($file_path, $data)){
199         $response = json_encode(array(status=>FALSE, message=>'Unable to execute', debug=>$file_path));
200+        $response = json_encode(array('status'=>FALSE, 'message'=>'Unable to execute', 'debug'=>$file_path));
201     }
202     else{
203         $submit_info = array(
204 @@ -442,15 +517,15 @@ class Submit extends CI_Controller
205             if($this->queue_model->add_to_queue_exec($submit_info)){
206                 if (!write_file($output_path, 'Queueing...')){
207                     $response = json_encode(array(status=>FALSE, message=>'Unable to write output file'));
208+                    $response = json_encode(array('status'=>FALSE, 'message'=>'Unable to write output file'));
209                 }
210                 else{
211                     process_the_queue();
212                     $response = json_encode(array(status=>TRUE, message=>'Executing'));
213+                    $response = json_encode(array('status'=>TRUE, 'message'=>'Executing'));
214                 }
215             }
216             else{
217                 $response = json_encode(array(status=>FALSE, message=>'Still in queue'));
218+                $response = json_encode(array('status'=>FALSE, 'message'=>'Still in queue'));
219             }
220         }
221     }
222     echo $response;
223 @@ -467,21 +542,21 @@ class Submit extends CI_Controller
224     $file_path = $user_dir.'/'.$EDITOR_OUT_NAME.'.'.$EDITOR_FILE_EXT;
225-
226     if (!file_exists($file_path)){
227         $response = json_encode(array(status=>FALSE, content=>''));
228+        $response = json_encode(array('status'=>FALSE, 'content'=>''));
229     }
230     else{
231         $this->load->helper('file');
232         $file_content = file_get_contents($file_path);
233         if ($file_content === FALSE){
234             $response = json_encode(array(status=>FALSE, content=>''));
235+            $response = json_encode(array('status'=>FALSE, 'content'=>''));
236         }
237         else{
238             $complete_status = strpos($file_content, 'Total Execution Time');
239             if($complete_status === FALSE){
240                 $response = json_encode(array(status=>FALSE, content=>$file_content));
241+                $response = json_encode(array('status'=>FALSE, 'content'=>$file_content));
242             }
243             else{
244                 $response = json_encode(array(status=>TRUE, content=>$file_content));
245+                $response = json_encode(array('status'=>TRUE, 'content'=>$file_content));
246             }
247         }
248     }
249 }
```

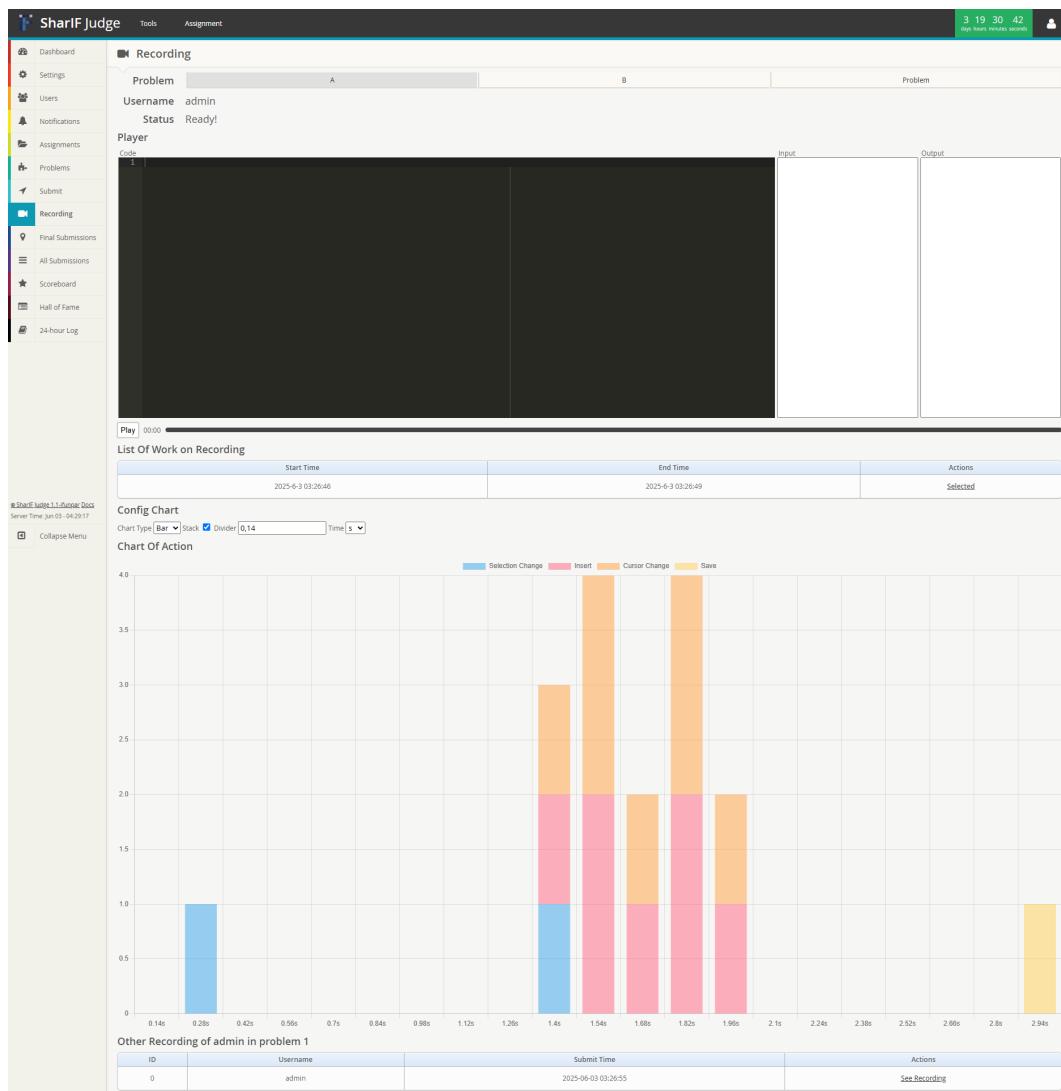
LAMPIRAN B

HASIL HALAMAN



The screenshot shows the Sharf Judge software interface. The title bar reads "Recording - Sharf Judge". The address bar shows the URL "https://svd839070.ngrok.io/recording/all" with a "Not secure" warning. The top right corner displays a timer: "3 19 36 52 days hours minutes seconds". The left sidebar has a dark blue header with the Sharf Judge logo and includes links: Dashboard, Settings, Users, Notifications, Problems, Submit, Recording (which is highlighted in blue), Final Submissions, All Submissions, Scoreboard, Hall of Fame, and 24-hour Log. The main content area is titled "Recording" and shows a table with one row of data. The table columns are: Problem (with dropdown options "All", "A", and "B"), ID, Username, Name, Problem, Submit Time, and Actions. The single entry in the table is: ID 0, Username admin, Name Admin, Problem 1, Submit Time 2025-06-03 03:26:55, and Actions "See Recording". At the bottom left, there is a note: "© Sharf Judge 1.1.5 (User) Server Time: Jun 03 04:23:07". At the bottom center, there is a "Collapse Menu" button.

Gambar B.1: Halaman Daftar Rekaman



Gambar B.2: Halaman Rekaman

LAMPIRAN C

FILE DOCKER EKSPERIMENT

Kode C.1: File *docker-compose* yang digunakan untuk Experiment

```
1 version: "3"
2 services:
3   codeigniter-3:
4     build: .
5     ports:
6       - "81:80"
7     volumes:
8       - ./var/www/html
9     depends_on:
10      - db
11
12   db:
13     image: mysql:5.7
14     ports:
15       - "3306:3306"
16     environment:
17       MYSQL_TABLE: judge
18       MYSQL_USER: sharif
19       MYSQL_PASSWORD: judge
20       MYSQL_ROOT_PASSWORD: root
21     command: --sql_mode=STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION
22     volumes:
23       - ./mysql:/var/lib/mysql
24
25   phpmyadmin:
26     image: phpmyadmin/phpmyadmin
27     ports:
28       - "8080:80"
29     environment:
30       PMA_HOST: db
31       MYSQL_ROOT_PASSWORD: freehost
32     depends_on:
33       - db
```

Kode C.2: File *Dockerfile* yang digunakan untuk Experiment

```
1 # Menggunakan image PHP 7.3 sebagai base image
2 FROM php:7.3-apache
3
4 # Install dependensi dan ekstensi PHP yang dibutuhkan untuk CodeIgniter
5 RUN apt-get update && apt-get install -y \
6     libpng-dev \
7     libjpeg-dev \
8     libldap2-dev \
9     libcurl4 \
10    libcurl4-openssl-dev \
11    libzip-dev \
12    libfreetype6-dev \
13    zip \
14    unzip \
15    default-jdk \
16    g++ \
17    python2 \
18    python3
19
20 # Install ekstensi GD dan mysqli
21 RUN docker-php-ext-configure gd --with-freetype-dir=/usr/include/ --with-jpeg-dir=/usr/include/ \
22     && docker-php-ext-install gd mysqli
23
24 RUN docker-php-ext-install curl
25
26 RUN docker-php-ext-configure ldap --with-libdir=lib/x86_64-linux-gnu/ \
27     && docker-php-ext-install ldap
28
29 RUN docker-php-ext-install fileinfo
30 RUN docker-php-ext-install mbstring
31 RUN docker-php-ext-install zip
32
33 RUN cp /usr/local/etc/php/php.ini-production /usr/local/etc/php/php.ini && \
34     sed -i -e "s/^memory_limit.*memory_limit=4G/g" /usr/local/etc/php/php.ini && \
35     sed -i -e "s/^max_input_vars.*max_input_vars=3000000/g" /usr/local/etc/php/php.ini && \
36     sed -i -e "s/^post_max_size.*post_max_size=50M/g" /usr/local/etc/php/php.ini && \
37     sed -i -e "s/^upload_max_filesize.*upload_max_filesize=50M/g" /usr/local/etc/php/php.ini
```

```
38
39 # Aktifkan mod_rewrite untuk Apache
40 RUN a2enmod rewrite
41
42 # Copy kode CodeIgniter ke dalam container
43 COPY . /var/www/html/
44
45 # Set direktori kerja
46 WORKDIR /var/www/html/
47
48 # Make Folder tester writeable by PHP
49 RUN chmod 777 /var/www/html/restricted/tester
50 RUN chmod 777 /var/www/html/application/cache/Twig
51
52 # Expose port 80
53 EXPOSE 80
54
55 # Jalankan Apache server
56 CMD ["apache2-foreground"]
```