Rancang Bangun Sistem *Online Judge* dan Pendeteksian Plagiarisme Menggunakan Arsitektur *Serverless*

Romadlon Rahmatulloh[1], Ricky Eka Putra[2]

[1,2] Teknik Informatika, Universitas Negeri Surabaya
[1]romadlonrahmatulloh16051204028@mhs.unesa.ac.id
[2]rickyeka@unesa.ac.id

***Abstrak*— Beberapa perguruan tinggi telah menjadikan Informatika sebagai jurusan yang dapat dipilih oleh calon mahasiswa. Salah satu kemampuan dasar yang harus dimiliki oleh mahasiswa Informatika adalah kemampuan menulis kode program. Perkembangan ilmu pengetahuan dan teknologi akan semakin menuntut mahasiswa untuk dapat menulis kode program yang semakin rumit. Proses evaluasi kode program yang dilakukan secara manual oleh dosen pengajar tidak lagi dapat mengikuti perkembangan ini, karena mempertimbangkan jumlah mahasiswa dan banyaknya baris kode yang harus diperiksa. Oleh karena itu dibutuhkan sistem *online judge* yang dapat mengevaluasi kode program hasil pekerjaan mahasiswa. Pada penelitian ini, sistem *online judge* dibangun menggunakan *Judge0 API*. Algortima *Sherlock N-Overlap* pada sistem ini mampu mendeteksi plagiarisme lebih efektif melebihi *tools* JPlag dan SIM, dengan nilai *harmonic average* lebih dari 0.8 dengan *threshold* sebesar 10 sampai dengan 90 pada skenario *known similarity*. Sedangkan skenario *Unknown Similarity* dengan *threshold* sebesar 50 sampai dengan 90 meghasilkan nilai *harmonic average* lebih dari 0.89. Sistem ini dibangun menggunakan arsitektur *serverless*. Sistem dibagi menjadi 2 bagian agar beban sistem untuk melayani permintaan pengguna lebih ringan. Sehingga sistem online judge ini dapat menangani permintaan pengguna hingga rata-rata 15.000 pengguna dengan sangat baik.**

***Kata Kunci*—Online Judge, Judge0 API, deteksi plagiarisme, Sherlock N-Overlap, AWS Lambda.**

I. PENDAHULUAN

Informatika merupakan salah satu disiplin ilmu yang mulai banyak ditekuni di Indonesia. Mulanya, hanya beberapa perguruan tinggi saja yang membuka Informatika sebagai jurusan sendiri. Melihat kebutuhan teknologi yang akan terus berkembang, semakin kesini sudah banyak perguruan tinggi yang mulai membuka jurusan Informatika meskipun tidak ada latar belakang kampus teknologi. Salah satu kemampuan dasar yang harus dimiliki mahasiswa jurusan Teknik Informatika adalah kemampuan mahasiswa dalam membuat program komputer (*coding*) [1] [2] [3]. Untuk itu, tugas dan ujian yang diberikan untuk mahasiswa jurusan Teknik Informatika akan selalu berhubungan dengan pemrograman, hal ini bertujuan untuk mengukur sejauh mana pengetahuan dan kemampuan mahasiswa mengenai kemampuan dasar yang harus dimilikinya.

Pada pelaksanaan *coding* yang pernah diikuti penulis, masih terdapat beberapa permasalahan, salah satunya adalah teknik evaluasi hasil pekerjaan kode program masih dilakukan secara manual. Proses evaluasi yang masih dilakukan secara manual memiliki peluang *human error* yang lebih besar jika mempertimbangkan jumlah mahasiswa dan baris kode yang dituliskan. Kendala teknis evaluasi yang masih manual ini akan menjadi semakin rumit ketika menemui permasalahan plagiarisme. Plagiarisme kode program adalah menyalin isi kode program, sebagian atau keseluruhan untuk digunakan kembali dalam *coding* dengan atau tanpa modifikasi [3] [4]. Kegiatan plagiarisme tidak pernah dibenarkan dalam hal apapun, termasuk juga dalam menuntut ilmu karena dapat mengaburkan penilaian kemampuan dasar mahasiswa.

Permasalahan lainnya adalah penggunaan media dalam pengumpulan kode program. Seringnya, penulis menemukan mahasiswa mengumpulkan kode program dalam benuk lembaran kertas atau langsung mengumpulkan kepada dosen bersangkutan melalui *e-mail* dll. Hal ini tentu sangat tidak efektif karena dosen harus melakukan eksekusi kode untuk setiap file program yang ditulis mahasiswanya.

Perkembangan Teknologi Informasi membawa dampak besar bagi pendidikan di Indonesia. Peran teknologi yang dimanfaatkan secara benar akan memudahkan tugas dosen maupun mahasiswanya, tak terkecuali pada kasus diatas. Salah satu pemanfaatan teknologi yang mampu menyelesaikan masalah ini adalah Online Judge [1] [3] [4] [5]. Online Judge adalah perangkat lunak yang digunakan untuk melakukan pengeksekusian kode program dan dapat menilai kesesuaian hasil keluaran program dengan ekspektasi yang diharapkan [1] [3] [5] [4]. Perangkat lunak ini akan menilai kesesuaian kode program yang dieksekusinya dengan parameter *input*, *output* dan beberapa konfigurasi lainnya sehingga pengguna dapat melihat hasil penilaian secara langsung.

Beberapa penelitian telah menambahkan fitur deteksi plagiarisme pada *Online Judge* yang dibuat. Pada [4] Dalam pembuatan sistem online judge, Minh dkk. menggunakan *DOMJudge* sebagai *Online Judge system* utamanya. *DOMJudge* adalah sebuat perangkat lunak *open source* yang digunakan untuk kontes pemrograman, seperti *International Collegiate Programming Regional Contest* dan *World Championship Programming Contests*. *DOMjudge* telah ditambahkan sebuah sistem plagiarisme menggunakan teknik *preprocessing* dan menggunakan *k-Grams*, *Hashing* dan *Suffix Array* untuk membandingkan dua kode program. Sistem tersebut menghasilkan cara lebih baik daripada metode konvensional. Sistem yang telah dibuat berhasil membuat tempat latihan untuk para siswa dalam mempelajari pemrograman. Sistem yang dibuat juga bekerja dengan baik

dalam pemeriksaan hasil pemrograman untuk membantu mencegah dan menghilangkan kecurangan dalam ujian.

Pada [6] Franca dkk. melakukan modifikasi pada *tools* Sherlock. Mereka menambahkan langkah Normalisasi dan mengubah koefisien similaritynya dengan Koefisien Overlap pada *tools* Sherlock. Mereka membandingkan hasil deteksi plagiarisme oleh *tools* Sherlock yang dimodifikasinya dengan *tools* lainnya yaitu Sim, Moss dan JPlag. Skenario pengujian yang dilakukan mereka dengan melakukan 2 kali uji coba, yaitu menguji kode program yang sudah diketahui plagiarismenya dan menguji kode program yang belum diketahui plagiarismenya. Hasilnya menunjukan dari kedua pengujian tersebut Sherlock N-Overlap memiliki *harmonic mean* yang terbaik dibandingkan *tools* yang lainnya. Sehingga pada penelitian ini menggunakan Sherlock N-Overlap sebagai deteksi plagiarisme pada kode program.

Pada perancangan sistem *online judge*, sistem dituntut untuk bisa menangani banyak permintaan karena mahasiswa akan mengakses sistem secara bersamaan. Sistem yang dirancang harus mampu menangani banyak permintaan dalam waktu yang bersamaan dan sistem harus selalu aktif untuk pengguna yang ingin mengaksesnya. Salah satu alternatif yang dapat memenuhi kebutuhan ini yaitu dengan membangun sistem dengan arsitektur *Serverless*. Ada banyak *platform serverless* yang tersedia, seperti AWS Lambda, Google Cloud Function, dan Microsoft Azure Function. AWS Lambda memiliki kemampuan memproses permintaan lebih cepat dan stabil [7] [8] [9]. Sehingga pada penelitian ini menggunakan AWS Lambda sebagai platform *serverless*.

Penelitian ini merancang sistem *online judge* yang dapat digunakan untuk sebagai media evaluasi kode program. Sistem yang dirancang juga dapat mendeteksi plagiarisme pada kode program dengan menggunakan algoritma *Sherlock N-Overlap*. Sistem dibagun pada arsitektur *serverless* dengan menggunakan platform AWS Lambda agar sistem tetap dapat menangani banyak permintaan tanpa ada kendala.

II. METODOLOGI

A. Deskripsi Umum Aplikasi

Sistem *online judge* yang diusulkan adalah aplikasi yang ditujukan untuk mempermudah penilaian tugas, ujian dan praktikum pada mata kuliah yang berkaitan dengan pemrograman di Jurusan Teknik Informatika Universitas Negeri Surabaya.

Pada aplikasi ini, dosen dapat mengunggah soal-soal tugas, ujian dan praktikum beserta *test case*-nya. *Test case* digunakan untuk memvalidasi kode program. Tiap soal dapat diisikan lebih dari satu *test case* agar dapat dilakukan uji coba untuk melihat kompleksitas jawaban (menyesuaikan soal atau kasus yang ada). Mahasiswa dapat mengumpulkan kode program sebagai jawaban dari masing-masing soal. Jawaban tersebut akan dicek menggunakan *Judge0 API* untuk mengetahui kesesuaian terhadap *test case* soal yang dikerjakan.

Tiap kali mahasiswa mengunggah jawaban, sistem deteksi plagiarisme akan dipanggil untuk membandingkan jawaban kode program yang baru diunggah dengan jawaban kode program yang sudah diunggah sebelumnya pada soal yang sama. Sistem *online judge* akan mencatat setiap perbandingan kedalam *database*. Sehingga setiap kode program yang diunggah memiliki jejak persentase plagiat dengan kode program milik mahasiswa lainnya.

Selain digunakan untuk tugas, ujian dan praktikum, sistem *online judge* juga dapat digunakan untuk kontes pemrograman seperti *Code Jam* yang secara tahunan digelar untuk mahasiswa Jurusan Teknik Informatika Universitas Negeri Surabaya. Sistem *online judge* yang diusulkan akan mempermudah tahap penjurian karena juri tidak perlu lagi mengeksekusi kode program secara manual.

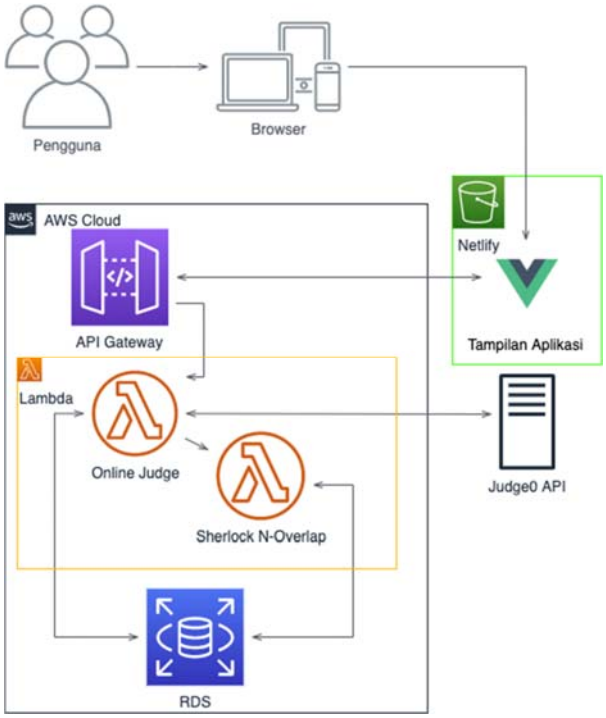
Setiap *event* seperti tugas, ujian, praktikum dan kontes, terdapat *scoreboard* untuk melihat rekapitulasi nilai tiap mahasiswa. Penilaian ini akan disesuaikan dengan beberapa kasus. Jika jawaban benar dan tidak melewati batas eksekusi pada setiap *test case* jawaban akan mendapatkan nilai penuh, akan tetapi jika hanya melewati beberapa *test case* yang dimasukkan maka nilai yang didapat merupakan hasil perkalian banyaknya *test case* yang dilalui dibagi dengan banyaknya *test case* soal dengan nilai penuh yang dimasukkan pada soal. Contoh jika soal diatur untuk mendapat nilai 40, jumlah test case 2 dan kode program yang di eksekusi hanya dapat melalui 1 test case saja maka nilai yang didapat adalah 20. Jika kode program yang di unggah tidak dapat melalui semua test case dan TLE (*Time Limit Exceed*) untuk jawaban yang melebihi batas waktu eksekusi akan diberi skor 0. Sistem juga menilai dari jumlah *compile* dan *running* kode program yang dilakukan mahasiswa. Pada sistem *online judge,* dosen akan mengatur jumlah maksimum *compile dan running* yang dapat dilakukan oleh mahasiswa. Jika mahasiswa melakukan *compile* dan *running* melebihi maksimum yang telah dosen tentukan, maka mahasiswa tidak akan mendapat nilai *compile* dan *running*. Selain penilaian *test case* dan *compile running*, sistem *online judge* juga menilai kode program mahasiswa berdasarkan persentase plagiarisme kode tersebut dengan kode mahasiswa lainnya. Dosen akan menentukan batas maksimum persentase plagiarisme yang dilakukan mahasiswa. Jika persentase plagiarisme yang dilakukan mahasiswa melebihi batas maksimum, maka mahasiswa tersebut akan kehilangan nilai plagiarisme. Dengan adanya Sistem online judge ini, dosen akan dimudahkan untuk mendapatkan nilai dari mahasiswa kelasnya dalam tugas, ujian atau praktikum bahasa pemrograman.

B. Arsitektur Sistem

Gbr. 1 menunjukan rancangan sistem *online judge* pada penelitian ini. Sistem yang diusulkan akan dibuat pada platform *website* seperti pada [1] [4] [5]. Untuk memaksimalkan pengalaman terbaik pengguna, sistem dipisah menjadi 2 bagian yaitu *Web Application* dan *Web Service*.

Bagian pertama akan dibangun menggunakan teknologi *Web Application*. *Web Application* adalah sebuah program yang berjalan pada sebuah *browser* sebagai *client* untuk menjalankan fungsi tertentu [10]. *Web Application* dibangun

menggunakan *Framework VueJS*. *VueJs* mendukung teknologi bernama *Single Page Application* (SPA) dimana *browser* hanya perlu mengunduh *resource* halaman sekali untuk dapat mengakses seluruh isi *Web Application*. Penggunaan *Single Page Application* membuat tampilan *website* menjadi lebih interaktif karena pengguna tidak perlu memuat halaman lagi untuk menuju suatu halaman yang ada pada *Web Application*. Sehingga penggunaan *VueJs* akan memaksimalkan pengalaman pengguna menjadi lebih matang [11]. Antarmuka aplikasi yang telah dibangun menggunakan *Framework Vue.js* akan di-*deploy* pada *Netlify*. *Netlify* merupakan penyedia platform yang lengkap untuk mengotomatisasi proyek web modern [12]. Salah satunya adalah mendukung *Framework VueJS* dalam mengotomatisasi *release* agar dapat diakses pengguna. *Netlify* menggunakan teknologi *Serverless* dimana pengguna hanya perlu menghubungkan Git sebagai tempat penyimpanan *project*-nya untuk melakukan *release* setiap ada *update* pada aplikasinya.



Gbr. 1 Rancangan Arsitektur

Bagian kedua akan dibangun menggunakan teknologi *Web Service*. *Web Service* bertugas sebagai penyedia data yang akan ditampilkan pada *Web Application* [10]. *Web Service* pada penelitian ini menggunakan platform *Serverless* AWS Lambda [13]. AWS Lambda adalah layanan komputasi *serverless* yang disediakan oleh Amazon Web Service (AWS) [13]. AWS Lambda dapat memproses permintaan dengan sangat cepat dan stabil seperti pada penelitian [7] [8] [9]. Sehingga memungkinkan pengguna dapat berinteraksi seperti memuat data, menerima data dan memproses dengan cepat. Data yang sudah di proses akan disimpan pada *database* RDS. Sistem *online judge* dan Algoritma Pendeteksian Plagiarisme dibangun pada AWS Lambda. Bahasa yang digunakan pada AWS Lambda adalah python versi 3.7.

C. Mekanisme Eksekusi Kode Program

Gbr. 2 Proses eksekusi kode program

Sistem yang diusulkan menggunakan aplikasi pihak ketiga bernama *Judge0 API*. Perangkat lunak *Judge0 API* tersedia *opensource* [14] sebagai alat bantu untuk mengeksekusi kode program. Gbr. 2 menunjukan ilustrasi alur eksekusi kode program. Pertama pengguna akan menuliskan kode program pada aplikasi *front end* yang dibangun dengan *Framework Vue.js*. Setelah menuliskan keseluruhan kode program, aplikasi akan mengirimkan data kode program ke server sistem *online judge*.

Setelah sistem *online judge* menerima kode program, sistem akan meneruskan kode program ke aplikasi pihak ketiga bernama *Judge0 API* dengan beberapa parameter. *Judge0 API* akan mengeksekusi kode program dan menilai kesesuaian kode program dengan *test case* soal. Pada *Judge0 API* terdapat beberapa parameter yang dikonfigurasi, diantaranya parameter *language_id, cpu_time_limit, memory_limit, source_code, stdin* dan *expected_output*.

Ketika kode program dieksekusi dan berjalan dengan benar, selanjutnya masukan dan luarannya akan dinilai. Jika masukkan dan luarannya sesuai, maka aplikasi pihak ketiga akan mengembalikan hasil eksekusi dengan pesan sukses ke server *online judge*. Jika masukan dan luaran tidak sesuai dengan yang diharapkan, *Judge0 API* akan mengembalikan hasil eksekusi dengan pesan *wrong answer*. Apabila terdapat kesalahan lainnya, maka aplikasi *Judge0 API* akan mengembalikan hasil eksekusi dengan pesan *error*.

D. Sherlock N-Overlap

Sherlock N-Overlap akan menjadi algoritma utama dalam mendeteksi persentase plagiarisme kode program jawaban. *Sherlock N-Overlap* akan diletakkan sesuai dengan ilustrasi pada Gbr. 1. Dimana *Sherlock N-Overlap* akan berdiri menjadi satu fungsi pada platform AWS Lambda. Fungsi ini akan dipanggil ketika mahasiswa mengunggah jawaban kode program pada sistem *online judge*.

Sherlock N-overlap adalah algoritma *Sherlock* yang dimodifikasi oleh Allyson, Danilo, Jose dan Giovanni dengan menambahkan metode normalisasi sebelum masuk ke algoritma *Sherlock* yang asli dan mengubah rumus *similarity*-

nya dengan koefisien overlap [6]. Dalam algoritma *Sherlock N-Overlap*, tidak ada perubahan yang dilakukan pada parameter konfigurasi, jadi *Sherlock N-overlap* berjalan dengan cara yang sama.

$$Sim(A,B) = 100 \times \frac{a}{a+b+c} \quad (1)$$

Pada *Sherlock* yang asli menggunakan koefisien kesamaan Jacquard seperti pada (1). Pada (1) dimana *a* adalah jumlah *signature* yang ditemukan di A, *b* jumlah *signature* yang ditemukan di B dan *c* adalah jumlah *signature* serupa yang ditemukan di A dan B, A dan B merupakan kode program yang akan dibandingkan. Koefisien kesamaan ini sensitif terhadap peningkatan nilai penyebut dan mengurangi kesamaan antara pasangan kode program. Jenis koefisien ini tidak efektif dalam mendeteksi kesamaan jika dimasukkannya kode yang tidak berguna pada file yang bertujuan untuk menyembunyikan kesamaan. Untuk meminimalkan masalah ini, koefisien *similarity* pada *Sherlock* menggunakan koefisien *Overlap*.

$$Overlap(A,B) = \frac{|A \cap B|}{\min(|A|,|B|)} \quad (2)$$

Persamaan (2) merupakan koefisien Overlap, koefisien *Overlap* yang formulasinya dinyatakan pada (2) mampu mengidentifikasi jumlah *signature* serupa (pembilang) dibandingkan dengan yang lebih kecil dari dua set *signature* (penyebut). Rumus ini mengurangi sensitivitas terhadap dimasukkannya struktur dan perintah kode program yang tidak berguna.

Untuk mendukung perbandingan pada *Sherlock N-overlap*, beberapa aturan *preprocessing* dibuat. Pertama normalisasi 1, 2 dan 3 dibuat. Normalisasi dilakukan secara berurutan mulai dari 1, 2 dan 3. Eksperimen dengan *Sherlock N-overlap* pada [6] telah menyatakan bahwa kode program yang menggunakan normalisasi 1 hingga 3 memberikan hasil yang lebih baik daripada tanpa *preprocessing*. Namun, hasil tersebut masih lebih rendah daripada *tools* lain yang digunakan untuk perbandingan. Oleh karena itu, aturan normalisasi 4 dan 5 ditambahkan untuk mendapatkan *preprocessing* yang terbaik dan tetap membiarkan karakteristik gaya dan logika pemrograman individu tetap utuh.

1. Aturan Normalisasi 1
   a. Bersihkan baris dan ruang kosong.
   b. Bersihkan semua komentar.
   c. Bersihkan library.
   d. Inklusi (atau eliminasi) *whitespace* di antara ekspresi, deklarasi variabel, dan struktur lainnya.
2. Aturan Normalisasi 2
   Bersihkan semua karakter yang dilampirkan dalam tanda kutip.
3. Aturan Normalisasi 3
   Bersihkan semua nilai literal dan variabel.
4. Aturan Normalisasi 4
   a. Bersihkan semua *reserved word* (for, while, do, int dan sebagainya)
   b. Penambahan aturan khusus untuk menyisipkan atau menghapus kekosongan diantara karakter.
5. Aturan Normalisasi 5
   Mengubah *reserved word* dengan synonymnya.

Pada aturan normalisasi 5, semua *reserverd word* yang berasal dari bahasa C dan C++ akan diubah dengan sinonimnya berdasarkan Tabel I.

TABEL I
RESERVERD WORD DAN SINONIMNYA

Reserverd Word	Sinonim
for, while, do	loop
int, float, char, double, long, bool	declarationType
printf, puts, cout	stdout
scanf, cin	stdin
auto, signed, const, extern, register, unsigned	defVar
if, else, switch, case, default	conditional
struct, typedef	structures

III. HASIL DAN PEMBAHASAN

A. *Uji Aplikasi pada Pengguna*

Sistem *online judge* yang dibuat telah diimplementasikan pada kegiatan belajar mengajar matakuliah struktur data. Sistem *online judge* ini diterapkan untuk evaluasi tugas praktikum kelas TI 19 A dengan jumlah mahasiswa sebanyak 51 orang, dan kelas TI 19 B dengan jumlah mahasiswa sebanyak 53 orang. Berdasarkan kuisioner yang diberikan, 80.6%responden menyatakan sistem ini memudahkan pengguna dalam evaluasi tugas praktikum dan berjalan sesuai dengan fungsi utamanya.

B. *Uji Plagiarisme*

Pengujian ini bertujuan untuk mengukur kemampuan Algoritma *Sherlock N-Overlap* dalam mendeteksi presentase kemiripan pada kode program. Pengujian ini dilakukan untuk mendapatkan nilai *precision*, *recall* dan *harmonic average. Precision* adalah tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh algoritma. *Recall* adalah tingkat keberhasilan algoritma dalam menemukan kembali sebuah informasi. Untuk mendapatkan nilai *precision* dan *recall* terlebih dahulu memisah data berdasarkan *Confussion Matrix*. *Confussion matrix* memisah data menjadi 4 kategori yaitu *True Positif* (TP), *False Positif* (FP), *True Negatif* (TN) dan *False Negatif* (FN). Nilai *True Negative* (TN) merupakan jumlah data bersifat negatif yang terdeteksi dengan benar, sedangkan *False Positive* (FP) merupakan data bersifat negatif namun terdeteksi sebagai data bersifat positif. Sementara itu, *True Positive* (TP) merupakan data bersifat positif yang terdeteksi benar. *False Negative* (FN) merupakan kebalikan dari *True Positive*, sehingga data bersifat positif, namun terdeteksi sebagai data yang bersifat negatif.

Uji plagiarisme dibagi menjadi dua macam pengujian, yaitu berdasarkan kode program yang telah diketahui fakta kesamaannya (*known similarity*) dan kode program yang belum diketahui fakta kesamaannya (*unknown similarity*)

1) Known Similarity: Uji plagiarisme menggunakan skenario *known similarity* dilakukan dengan bantuan tiga *programmer* untuk melakukan penulisan kode program dalam bahasa C++. Terdapat dua jenis *assignment* berdasarkan jumlah baris kode yang harus dikerjakan. *Assignment* A terdiri sekitar 90-100 baris kode dan *Assignment* B terdiri sekitar 40-50 baris kode. Tiga *programmer* tersebut kemudian melakukan modifikasi terhadap kode program yang telah dikerjakan sesuai dengan aturan seperti pada [6]. Aturan modifikasi sebagai berikut:

1. Menyalin kode original tanpa merubah apapun
2. Memodifikasi bagian komentar
3. Mengubah *identifier*
4. Mengubah posisi variable dan fungsi
5. Mengubah *variable scope*
6. Mengubah indentation
7. Menambahkan informasi yang tidak relevan, seperti library, varible yang tidak digunakan, dan komentar
8. Mengubah operasi matematika
9. Menggabungkan semua perubahan diatas
10. Menambahkan code dari *assignment* yang lain

Sehingga, kode program yang diujikan pada skenario ini berjumlah 66 kode program (6 kode asli dan 60 kode modifikasi sesuai dengan aturan tersebut untuk tiap *assignment*).

Pada algoritma deteksi plagiarisme, *precision* adalah nilai yang merepresentasekan proporsi pasangan kode yang faktanya sama dan dinyatakan sama oleh algoritma dari semua kode yang dinyatakan sama (termasuk kode yang dinyatakan sama oleh algoritma tetapi faktanya tidak sama). Sedangkan *precision* merepresentasekan proporsi pasangan kode yang faktanya sama dan dinyatakan sama dari semua kode yang faktanya sama (termasuk kode yang faktanya sama tetapi dinyatakan tidak sama oleh algoritma). *Confusion Matrix* yang digunakan seperti pada table II.

TABEL III
CONFUSION MATRIX

Pasangan yang dianggap tidak sama dengan *threshold* tertentu	**Pasangan yang dianggap sama dengan *threshold* tertentu**
Pasangan yang sama tapi tidak dianggap sama (D) (False negative)	Pasangan yang relevan dan dianggap sama (A) (True positive)
Pasangan yang tidak sama dan tidak dianggap sama (C) (True negative)	Pasangan yang tidak sama tetapi dianggap sama (B) (False positive)

Untuk kode program yang sudah diketahui *similarity*-nya akan dihitung *precision* dan *recall* menggunakan parameter yang ada pada Tabel II. Pada Tabel II, A menyatakan pasangan kode yang dianggap sama oleh algoritma pada *threshold* tertentu dan sesuai dengan fakta sebenarnya (bahwa pasangan kode tersebut sama). B menyatakan pasang kode yang dianggap sama oleh algoritma tetapi pasang kode tersebut sebenarnya berbeda. C menyatakan pasangan kode yang dianggap tidak sama oleh algoritma dan faktanya memang tidak sama. D menyatakan pasangan kode yang sebenarnya sama tetapi pasang kode tersebut tidak dianggap sama oleh algoritma.

Setelah didapat nilai parameter dari Tabel II kemudian akan dihitung *precision* dan *recall* untuk menghitung performa algoritma. Untuk menghitung *precision* menggunakan (3).

$$P = \frac{A}{A+B} \quad (3)$$

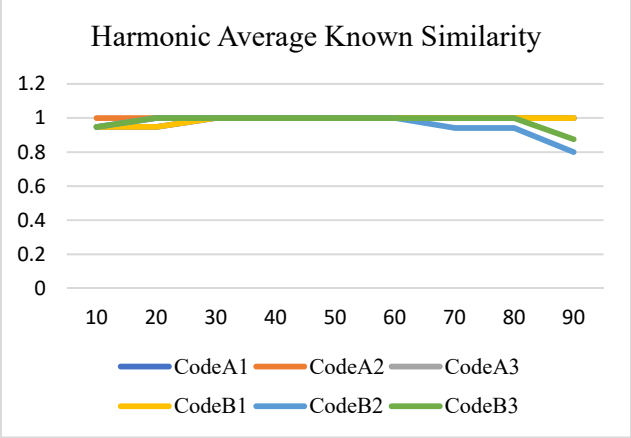
Sedangkan menghitung *recall* menggunakan (4).

$$R = \frac{A}{A+D} \quad (4)$$

Jika algoritma menghasilkan semua pasangan yang faktanya sama (R=1), tidak berarti algoritma ini ideal. Begitu juga jika algoritma menyatakan hanya pasangan yang sama (P=1) maka algoritma tidak berarti ideal. Jadi untuk mengetahui performa algoritma yaitu dengan *harmonic average*. *Harmonic average* adalah perhitungan yang mengombinasikan *precision* dan *recall*. Untuk menghitung *harmonic average* menggunakan (5).

$$F = 2 \times \frac{P \times R}{P+R} \quad (5)$$

Hasil uji plagiarisme dengan *Sherlock N-Overlap* menggunakan skenario *known similarity* didapatkan harmonic average dapat dilihat pada Gbr. 3



Gbr. 3 Hasil uji skenario *known similarity*

CodeA1 menyatakan kelompok kode program yang dihasilkan *programmer* 1 dari *assignment* A. CodeA2 menyatakan kelompok kode program yang dihasilkan *programmer* 2 dari *assignment* A. CodeA3 menyatakan kelompok kode program yang dihasilkan *programmer* 3 dari *assignment* A. CodeB1 menyatakan kelompok kode program yang dihasilkan *programmer* 1 dari *assignment* B. CodeB2 menyatakan kelompok kode program yang dihasilkan *programmer* 2 dari *assignment* B. CodeB3 menyatakan kelompok kode program yang dihasilkan *programmer* 3 dari *assignment* B. Gbr. 3 menunjukan bahwa *Sherlock N-Overlap* efektif untuk mendeteksi kesamaan antara kode program yang terdiri sekitar 40-50 baris kode dan 90-100 baris kode. *Sherlock N-Overlap* sangat baik mendeteksi plagiarisme pada *threshold* 30 sampai 60 tanpa ada penurunan nilai *harmonic average*. Akan tetapi, nilai *harmonic average* akan mengalami

penurunan pada threshold lebih dari 60 dan kurang dari 30. Hasil *harmonic average* yang ditunjukan pada Gbr. 3 selalu bernilai lebih dari 0.8 pada semua *threshold*.

*2) Unknown Similarity:*Untuk mengetahui performa kode program yang tidak diketahui fakta kesamaannya akan menjadi sulit mendapatkan nilai *true positif*, karena tidak ada fakta yang menyatakan dengan pasti. Demi menyelesaikan permasalahan ini performa akan dihitung dengan mencari pendekatan nilai *precision* dan *recall* menggunakan metode *Conformity* [6]. Metode *Conformity* mencari nilai *true positif* dan mendapatkan pendekatan nilai *precision* dan *recall* bergantung hasil tool deteksi yang lainnya. Misal akan mencari nilai *recall* dan *precision* dari algoritma *Sherlock*, *tool* deteksi plagiarisme lainnya seperti JPlag dan SIM akan digunakan bersamaan untuk mendapatkan nilai *true positif*. Artinya metode ini mendapatkan pendekatan nilai *true positif* dengan kesepakatan bersama oleh *tool* lainnya.

Parameter yang dibutuhkan untuk mencari nilai *precision* dan *recall* dengan metode *Conformity* adalah

1. NOS : Jumlah pasangan kode yang dinyatakan sama oleh Algoritma.
2. NIOS : Jumlah pasang kode yang dinyatakan sama oleh Algoritma tetapi tidak dinyatakan sama oleh Algoritma lainnya.
3. NIAS : Jumlah pasangan kode yang dinyatakan tidak sama oleh Algoritma tetapi dinyatakan sama oleh Algoritma lainnya.

Dari parameter yang didapatkan, akan dicari pendekatan nilai *true positif* (NTP) dengan (6). Persamaan (6) didapatkan dari asumsi bahwa nilai *true positif* merupakan banyaknya pasang kode yang dianggap sama oleh semua *tools*.

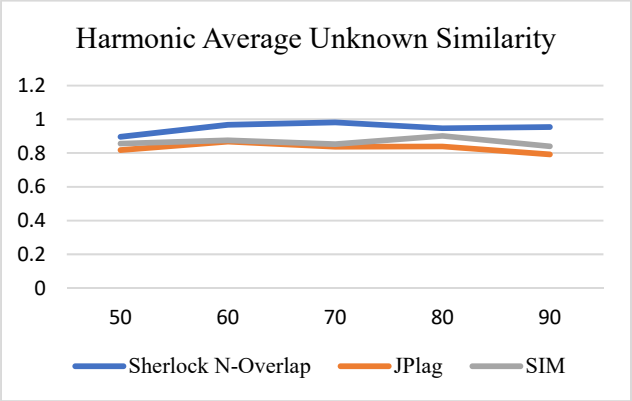
$$NTP = NOS - NIOS \quad (6)$$

Setelah didapatkan NTP akan dicari nilai *precision* dan *recall*. Persamaan (7) digunakan untuk mencari nilai *precision conformity*. Dan Persamaan (8) digunakan untuk mencari nilai *recall conformity*. Kemudian, hasil *precison* dan *recall* akan digunakan untuk menghitung *harmonic average* dengan (5).

$$P_{Conf} = \frac{NTP}{\max(NTP)+NIOS} \quad (7)$$

$$R_{Conf} = \frac{NTP}{\max(NTP)+NIAS} \quad (8)$$

Skenario uji ini dilakukan pada *source code* hasil pekerjaan praktikum dan ujian akhir semester pada matakuliah struktur data kelas TI 19 A dan TI 19 B di jurusan Teknik Informatika Universitas Negeri Surabaya. Jumlah mahasiswa pada kelas TI 19 A adalah 51 mahasiswa dan pada kelas TI 19 B sebanyak 53 mahasiswa. Hasil uji plagiarisme menggunakan skenario *known similarity* dapat dilihat pada Gbr. 4



Gbr. 4 Hasil uji skenario *unkwon similarity*

Gbr. 4 menunjukan *Sherlock N-Overlap* mampu mendeteksi kelompok kode program yang belum diketahui *similarity*-nya dengan sangat baik. Hasil nilai *harmonic average* yang didapat *Sherlock N-Overlap* melebihi nilai *harmonic average* dari tools JPlag dan SIM. Nilai *harmonic average* yang diperoleh *Sherlock N-Overlap* yang konsisten selalu mendekati nilai 1.

C. *Uji Performa Aplikasi*

Pengujian ini dilakukan dengan metode *stress testing*. *Stress testing* dilakukan untuk mengetahui batasan kapasitas penggunaan yang dapat mengakses sistem dalam waktu bersamaan [15]. *Stress testing* dilakukan menggunakan *software Apache Jmeter*. *Apache Jmeter* diinstall pada sistem operasi *Windows Server 2019* dengan RAM 30GB dan 8 CPU *Intel Xeon* 2Ghz dengan *bandwith* 1GBs.

Pada pengujian ini terdapat tiga parameter pendukung, yaitu *Number of Thread* yang mengidentifikasi jumlah pengguna, *Rump-Up periode* yang mengatur lama *Apache Jmeter* untuk meningkatkan jumlah *thread*, dan *loop count* yang mengidentifikasikan banyaknya perulangan pengujian.

Stress testing dilakukan untuk mengetahui performa pada 2 bagian sistem yaitu, pada antarmuka aplikasi yang di-*deploy* pada *Netlify* dan AWS Lambda. Stress testing ini akan menggunakan 4 skenario uji yang menggambarkan perbedaan jumlah dari tiap-tiap parameter. Detail skenario *stress testing* dapat dilihat pada Tabel III.

TABEL III
DETAIL SKENARIO STRESS TESTING

Skenario	Ramp-periode	Loop Count	Number of Thread
1	5	1	10-20000
2	15	1	10-20000
3	15	2	10-20000
4	15	5	10-20000

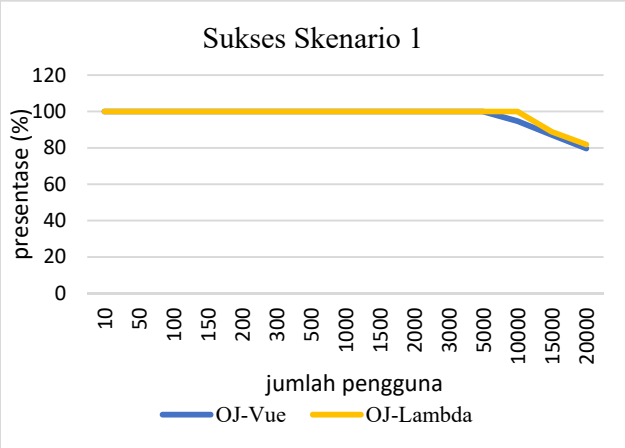
Pengujian performa dimulai dengan mengatur konfigurasi Apache JMeter sesuai dengan skenario pada table III.

Kemudian Apache JMeter akan membuat *vritual user agent* yang akan mengakses target. *Virtual user agent* yang dibuat sesuai dengan jumlah *Number of Thread* yang diatur pada pengaturan Apache Jmeter. *Virtual user agent* tersebut akan naik sesuai waktu pada Ramp-periode. Misal pada skenario 1, *Virtual user agent* akan naik sampai 20000 dalam waktu 5 detik. Kemudian *virtual user agent* akan mengakses target sebanyak *Loop Count*. Setiap *virtual user agent* akan menerima respon dari target berupa status sukses dan gagal.

Hasil uji *stress testing* pada skenario 1 dapat dilihat pada Tabel IV

TABEL IV
HASIL UJI STRESS TESTING PADA SKENARIO 1

Test ID	NOT	Sukses (%)		error (%)	
		Vue	lambda	Vue	lambda
1	10	100	100	0	0
2	100	100	100	0	0
3	500	100	100	0	0
4	1000	100	100	0	0
5	2000	100	100	0	0
6	3000	100	100	0	0
7	5000	100	100	0	0
8	10000	94,7	100	5,3	0
9	15000	87,05	88,85	12,95	11,15
10	20000	79,66	81,88	20,34	18,12



Gbr. 5 Tingkat sukses pada uji skenario 1

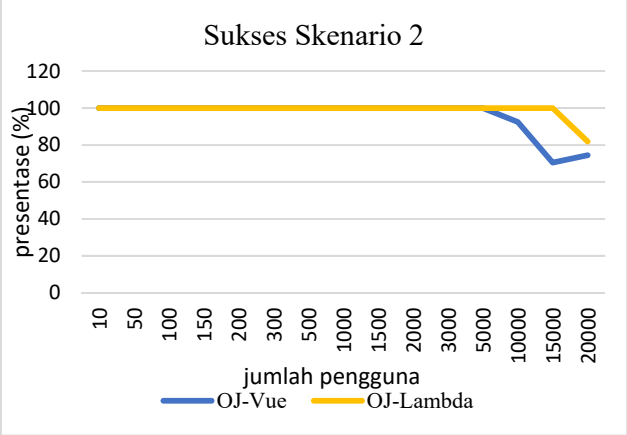
Gbr. 5 dan Tabel IV menunjukan hasil *stress testing* dengan skenario 1. Server *Vue* dapat melayani permintaan dengan sukses hingga 5.000 pengguna tanpa mengalami *error*. Sedangkan server AWS Lambda dapat melayani permintaan dengan sukses hingga 10.000 pengguna tanpa mengalami error. Sehingga pada sistem online judge ini diuji dengan skenario 1 mampu melayani permintaan pengguna dengan sangat baik hingga 15.000 pengguna.

Hasil uji pada skenario 2 dapat dilihat pada Tabel V.

TABEL V
HASIL UJI STRESS TESTING PADA SKENARIO 2

Test ID	NOT	Sukses (%)		error (%)	
		Vue	lambda	Vue	lambda
1	10	100	100	0	0
2	100	100	100	0	0
3	500	100	100	0	0
4	1000	100	100	0	0
5	2000	100	100	0	0
6	3000	100	100	0	0
7	5000	100	100	0	0
8	10000	92,5	100	7,5	0
9	15000	70,5	100	29,5	0
10	20000	74,4	81,87	25,26	18,13

Gbr. 6 dan Tabel V menunjukan hasil *stress testing* dengan skenario 2. Server *Vue* dapat melayani permintaan dengan sukses hingga 5.000 pengguna tanpa mengalami *error*. Sedangkan server AWS Lambda dapat melayani permintaan dengan sukses hingga 15.000 pengguna tanpa mengalami error. Sehingga pada sistem online judge ini diuji dengan skenario 2 mampu melayani permintaan pengguna dengan sangat baik hingga 20.000 pengguna.



Gbr. 6 Tingkat sukses pada uji skenario 2

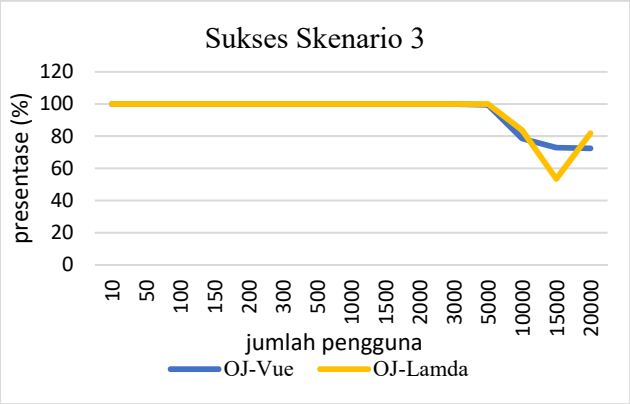
Hasil uji pada skenario 3 dapat dilihat pada Tabel VI

TABEL VI
HASIL UJI STRESS TESTING PADA SKENARIO 3

Test ID	NOT	Sukses (%)		error (%)	
		Vue	lambda	Vue	lambda
1	10	100	100	0	0
2	100	100	100	0	0
3	500	100	100	0	0
4	1000	100	100	0	0
5	2000	100	100	0	0
6	3000	100	100	0	0
7	5000	99,44	100	0,56	0
8	10000	78,46	83,73	21,54	16,27
9	15000	72,85	53,4	27,15	46,6

10	20000	72,41	81,88	27,59	18,12

Gbr. 7 dan Tabel VI menunjukan hasil *stress testing* dengan skenario 3. Server *Vue* dapat melayani permintaan dengan sukses hingga 3.000 pengguna tanpa mengalami *error*. Sedangkan server AWS Lambda dapat melayani permintaan dengan sukses hingga 5.000 pengguna tanpa mengalami error. Sehingga pada sistem online judge ini diuji dengan skenario 3 mampu melayani permintaan pengguna dengan sangat baik hingga 8.000 pengguna.



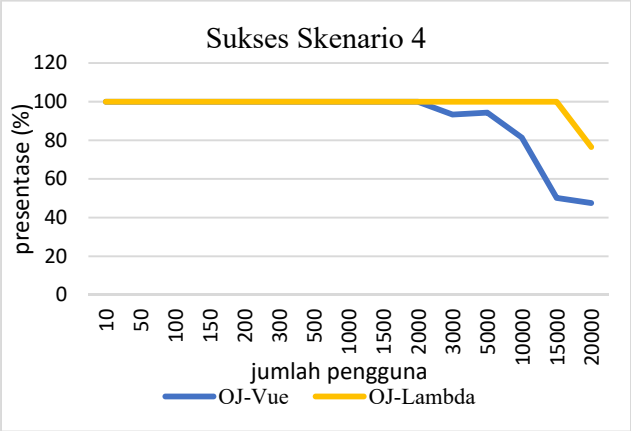
Gbr. 7 Tingkat sukses pada uji skenario 3

Hasil uji pada skenario 4 dapat dilihat pada Tabel VII

TABEL VII
HASIL UJI STRESS TESTING PADA SKENARIO 4

Test ID	NOT	Sukses (%)		error (%)	
		Vue	lambda	Vue	lambda
1	10	100	100	0	0
2	100	100	100	0	0
3	500	100	100	0	0
4	1000	100	100	0	0
5	2000	100	100	0	0
6	3000	93,33	100	6,67	0
7	5000	94,37	100	5,63	0
8	10000	81,3	100	18,7	0
9	15000	50,19	100	49,81	0
10	20000	47,79	76,53	52,51	23,47

Gbr. 8 dan Tabel VII menunjukan hasil *stress testing* dengan skenario 4. Server *Vue* dapat melayani permintaan dengan sukses hingga 2.000 pengguna tanpa mengalami *error*. Sedangkan server AWS Lambda dapat melayani permintaan dengan sukses hingga 15.000 pengguna tanpa mengalami error. Sehingga pada sistem online judge ini diuji dengan skenario 4 mampu melayani permintaan pengguna dengan sangat baik hingga 17.000 pengguna.



Gbr. 8 Tingkat sukses pada uji skenario 4

Dari hasil 4 skenario uji *stress testing* yang didapatkan, terjadi beberapa jenis error yang menyebabkan tingkat persentasi sukses menurun. *Error* yang terjadi yaitu *bad gateway*, *internal service error*, *no connection*, *failed conection time out* dan *service unavailable. Error* paling banyak terjadi adalah *no connection*. *No connection* terjadi karena Apache JMeter tidak dapat membuat koneksi ke server uji karena keterbatasan *resource* untuk membuka koneksi. Kenaikan jumlah pengguna (NOT) dalam waktu (Ramp-Periode) yang singkat akan menyebabkan tingkat sukses menurun. Begitu juga *loop count* yang dilakukan oleh pengguna akan mengakibatkan kenaikan tingkat error yang terjadi.

IV. KESIMPULAN

Berdasarkan hasil penelitian yang telah dilakukan, kesimpulan yang diperoleh dari perencanaan dan pembuatan sistem *online judge* dan pendeteksian plagiarisme adalah:

1. Berdasarkan kuisioner yang diberikan, 80.6% responden menyatakan sistem ini memudahkan pengguna dalam evaluasi tugas praktikum dan berjalan sesuai dengan fungsi utamanya. Dari sisi mahasiswa, kemudahan dalam pengujian dan pengumpulan kode program, sedangkan dari sisi dosen juga kemudahan dalam mengelola soal, jawaban serta mendapatkan nilai mahasiswa secara otomatis.
2. Sistem mampu mendeteksi plagiarisme secara otomatis pada kode program mahasiswa. Penggunaan *Sherlock N-Overlap* dalam mendeteksi plagiarisme pada kode program mahasiswa telah berhasil mendeteksi dengan baik. Pada uji plagiarisme menggunakan skenario *known similarity* dengan *threshold* sebesar 10 sampai dengan 90 selalu didapatkan nilai *harmonic average* lebih dari 0.8. Pada uji plagiarisme dengan skenario *Unknown Similarity* dengan *threshold* sebesar 50 sampai dengan 90 selalu meghasilkan nilai *harmonic average* lebih dari 0.89. Hasil ini membuktikan bahwa penggunaan *Sherlock N-Overlap* efektif dalam mendeteksi plagiarisme pada kode program melebihi *tools* JPlag dan SIM.
3. Dari hasil uji *stress testing*, sistem berhasil melayani banyak permintaan dari pengguna dalam waktu bersamaan. Dengan

dibaginya sistem menjadi 2 bagian membuat beban sistem untuk melayani permintaan pengguna lebih ringan. *AWS Lambda* mampu menangani permintaan pengguna hingga rata-rata 12.500 pengguna tanpa ada penurunan performa. VueJS mampu menangani permintaan hingga rata-rata 3.750 pengguna tanpa ada penurunan performa. Sehingga sistem online judge ini dapat menangani permintaan pengguna hingga rata-rata 15.000 pengguna dengan sangat baik.

V. SARAN

Perlu integrasi sistem *online judge* dengan dengan sistem akademik yang bersangkutan, sehingga sistem akan lebih mudah mendapatkan data mahasiswa dan mengirim nilai ke sistem akademik pusat. Sistem juga perlu untuk dikembangkan lebih lanjut mengenai deteksi plagiarisme pada kode program mahasiswa yang dapat memisah kode program yang memang plagiarisme dan kode program yang tidak sengaja plagiarisme.

UCAPAN TERIMA KASIH

Terima kasih disampaikan kepada Mahasiswa Jurusan Teknik Informatika Universitas Negeri Surabaya terutama kelas Struktur Data prodi S1 Teknik Informatika 2019A dan 2019B yang bersedia membantu pengujian sistem, Para Dosen Teknik Informatika yang memberi masukkan dan saran terkait sistem yang diusulkan, dan teman-teman yang bersedia membantu menyelesaikan penelitian ini.

REFERENSI

[1] R. Harimurti, A. Qoiriah, A. Iwan NH dan A. , "Pengembangan Fitur Penilaian dan Perangkingan pada Automatic Programming Assessment Tools," *Journal Information Engineering and Educational Technology,* vol. 2, no. 2, hal. 96-100, 2018.

[2] J. Maulindar dan . D. A. Cahyani, "ANALISIS FAKTOR-FAKTOR YANG MEMPENGARUHI MINAT MAHASISWA UNTUK MENJADI PROGRAMMER," *Jurnal INFORMA Politeknik Indonusa Surakarta,* vol. 5, no. 2, hal. 14-17, 2019.

[3] R. A. Sukamto, E. P. Nugroho dan M. Nursalam, "Implementasi Pendeteksi Code Clone pada Online Judge Sebagai Sarana Mengurangi Kecurangan Akademik Mahasiswa Program Studi Ilmu Komputer," *Jurnal Cybermatika,* vol. 2, no. 2, hal. 1-6, 2014.

[4] M. T. Pham dan T. B. Nguyen, "The DOMJudge Based Online Judge System with Plagiarism Detection," dalam *2019 IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF)*, Danang, 2019.

[5] H. Sun dan B. L. Min Jiao, "YOJ: An Online Judge System Designed for Programming Courses*," dalam *The 9th International Conference on Computer Science & Education (ICCSE 2014)*, Vancouver, 2014.

[6] F. B. Alyson, M. L. Danilo dan B. C. Giovanni, "Sherlock N-overlap: invasive normalization and overlap coefficient for the similarity analysis between source code," *IEEE Transactions on Computers,* vol. 68, no. 5, hal. 740-751, 2019.

[7] D. Jackson dan G. Clynch, "An Investigation of the Impact of Language Runtime on the Performance and Cost of Serverless Functions," dalam *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, Zurich, 2018.

[8] H. Lee, K. Satyam dan G. Fox, "Evaluation of Production Serverless Computing Environments Publisher: IEEE," dalam *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, San Francisco, 2018.

[9] G. McGrath dan P. R. Brenner, "Serverless Computing: Design, Implementation, and Performance Publisher: IEEE," dalam *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, Atlanta, 2017.

[10] A. Wirasandi, S. F. S. Gumilang dan M. A. Hasibuan, "Integrasi Line Bot Layanan Pesan Antar Makanan "dikampus" Menggunakan Line Front-end Framework (liff) Dengan Metode Iterative Incremental," dalam *e-Proceeding of Engineering*, Bandung, 2019.

[11] S. P. Adithama dan M. Maslim, "PEMBANGUNAN SISTEM INFORMASI PERPUSTAKAAN SEKOLAH DASAR BERBASIS WEB," *Jurnal Pengabdian Kepada Masyarakat,* vol. 3, no. 2, hal. 350-360, 2019.

[12] Netlify, "Docs Netlify," Netlify, [Online], https://docs.netlify.com. tanggal akses: 3 Mei 2020.

[13] M. Villamizar, O. Garcés, L. Ochoa, H. Castro, L. Salamanca, M. Verano, R. Casallas, S. Gil, C. Valencia, A. Zambrano dan M. Lang, "Infrastructure Cost Comparison of Running Web Applications in the Cloud Using AWS Lambda and Monolithic and Microservice Architectures," dalam *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, Cartagena, 2016.

[14] Judge0, "Judge0 API docs," [Online], https://api.judge0.com/. tanggal akses: 28 Februari 2020.

[15] M. A. Putri dan H. N. Hadi, "Performance Testing Analysis on Web Application: Study Case Student Admission Web System," dalam *2017 International Conference on Sustainable Information Engineering and Technology (SIET)*, Malang, 2017.