TUGAS AKHIR - KI141502

RANCANG BANGUN APLIKASI ONLINE JUDGE UNTUK APLIKASI BERBASIS JARINGAN MEMANFAATKAN VIRTUALISASI BERBASIS DOCKER

ALI ARIFF
NRP 5111 100 115

Dosen Pembimbing 1
Royyana Muslim Ijtihadie, S.Kom, M.Kom, Ph.D.

Dosen Pembimbing 2
Baskoro Adi Pratomo, S.Kom, M.Kom

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2015

UNDERGRADUATE THESES - KI141502

DESIGN AND IMPLEMENTATION OF ONLINE JUDGE FOR NETWORK BASED APPLICATION USING DOCKER BASED VIRTUALIZATION

ALI ARIFF
NRP 5111 100 115

Supervisor 1
Royyana Muslim Ijtihadie, S.Kom, M.Kom, Ph.D.

Supervisor 2
Baskoro Adi Pratomo, S.Kom, M.Kom

INFORMATICS DEPARTMENT
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya, 2015

RANCANG BANGUN APLIKASI ONLINE JUDGE UNTUK APLIKASI BERBASIS JARINGAN MEMANFAATKAN VIRTUALISASI BERBASIS DOCKER

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Berbasis Jaringan
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
Ali Ariff
NRP: 5111 100 115

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Royyana Muslim Ijtihadie, S.Kom, M.Kom, Ph.D. .....
NIP: 197708242006041001 (Pembimbing 1)

Baskoro Adi Pratomo, S.Kom, M.Kom .....
NIP: 198702182014041001 (Pembimbing 2)

SURABAYA
JUNI 2015

RANCANG BANGUN APLIKASI ONLINE JUDGE UNTUK APLIKASI BERBASIS JARINGAN MEMANFAATKAN VIRTUALISASI BERBASIS DOCKER

Nama	: ALI ARIFF
NRP	: 5111 100 115
Jurusan	: Teknik Informatika FTIF - ITS
Pembimbing I	: Royyana Muslim Ijtihadie, S.Kom, M.Kom, Ph.D.
Pembimbing II	: Baskoro Adi Pratomo, S.Kom, M.Kom

Abstrak

Evaluasi tugas pemrograman jaringan sebagian besar diperiksa dengan cara konvensional yaitu dievaluasi oleh asisten atau dosen. Cara ini masih memiliki beberapa kelemahan, sebagai contoh feedback yang diterima kurang maksimal karena waktu yang diperlukan untuk evaluasi lama, dan adanya elemen penting yang mungkin terlewatkan untuk diperiksa. Selain itu, faktor siapa yang memeriksa juga dapat membedakan hasil evaluasi.

Saat ini Online Judge digunakan sebagai solusi pengecekan soal pemrograman. Tetapi dengan menggunakan Online Judge juga masih memiliki masalah, salah satunya adalah kode sumber yang akan diuji tidak dapat dipercaya sepenuhnya, dikarenakan adanya kemungkinan kode sumber berisi kode sumber yang akan merusak komputer. Solusi untuk permasalahan ini adalah dengan menggunakan sandboxing, di mana kode sumber yang diunggah akan dieksekusi secara terisolasi.

Pada tugas akhir ini, dilakukan perancangan dan implementasi Online Judge untuk aplikasi berbasis jaringan yang memanfaatkan Docker sebagai lingkungan virtual terisolasi untuk sandboxing, kode sumber yang diunggah akan dieksekusi di dalam lingkungan virtual yang terisolasi sehingga tidak akan berdampak buruk kepada komputer host.

Kata Kunci: Online Judge, Virtualisasi, Docker, Aplikasi Jaringan

DESIGN AND IMPLEMENTATION OF ONLINE JUDGE FOR NETWORK BASED APPLICATION USING DOCKER BASED VIRTUALIZATION

Name	: ALI ARIFF
NRP	: 5111 100 115
Major	: Informatics Department Faculty of IT - ITS
Supervisor I	: Royyana Muslim Ijtihadie, S.Kom, M.Kom, Ph.D.
Supervisor II	: Baskoro Adi Pratomo, S.Kom, M.Kom

Abstract

Evaluation of network-based programming tasks are mostly checked with conventional way - such as presentation - still have several drawbacks, for example the feedback is not maximal because of the time needed to evaluate, and the important elements that may be missed to be examined. In addition, factor on who is checking can also differentiate the results of evaluation.

Nowadays Online Judge has been used as a solution to programming problem checking. But the use of online judging still have some problems, one of which is the source code to be tested cannot be trusted completely due to the possibility of the source code contains code that will damage the computer. The solution to this problem is to use sandboxing, which the uploaded source code will be executed in isolation.

In this thesis, the design and implementation of online judge for network-based applications utilizing docker as an isolated virtual environment is conducted, the uploaded source will be executed in an isolated virtual environment so it will not adversely affect the host computer.

Keywords: Online Judge, Virtualization, Docker, Network Application

KATA PENGANTAR

Assalamualaikum

Segala puji dan syukur semata ditujukan ke hadirat Allah SWT yang telah memberikan rahmat dan hidayah-Nya sehingga memungkinkan penulis untuk menyelesaikan Tugas Akhir yang berjudul **"Rancang Bangun Aplikasi Online Judge untuk Aplikasi Berbasis Jaringan Memanfaatkan Virtualisasi Berbasis Docker"**.

Tugas Akhir ini dibuat berdasarkan pengalaman penulis sebagai asisten mata kuliah Pemrograman Jaringan pada semester genap tahun ajaran 2014/2015 dan juga sebagai admin dan pengelola LPOJ di Laboratorium Pemrograman (LP). Penulis merasa proses penilaian saat demo program mata kuliah tersebut belum efektif. Diharapkan dengan terselesaikannya Tugas Akhir ini dapat membuat proses penilaian mata kuliah tersebut bisa efektif serta dapat digunakan secara rutin dalam proses pembelajaran di Jurusan Teknik Informatika ITS seperti LPOJ yang sudah ada di Laboratorium Pemrograman (LP).

Tidak lupa penulis sampaikan terima kasih dengan segala rasa penghormataan seluas-luasnya kepada beberapa pihak yang telah membantu dalam menyelesaikan Tugas Akhir ini selama sekitar tiga bulan:

- Kedua orang tua penulis, saudara dan kerabat dekat yang selalu khawatir dengan keadaan penulis dan selalu mendoakan kelancaran pengerjaan Tugas Akhir ini.
- Bapak Royyana Muslim Ijtihadie, S.Kom, M.Kom, Ph.D. selaku pembimbing I yang selalu menuntun dan membimbing penulis dalam pelaksanaan Tugas Akhir ini dan selama masa kuliah.
- Bapak Baskoro Adi Pratomo, S.Kom, M.Kom selalu pembimbing II yang selalu memberikan bimbingan dalam penger-

jaan Tugas Akhir ini dan selama masa kuliah.

- Ibu Dr. Eng. Nanik Suciati, S.Kom, M.Kom selaku Ketua Jurusan Teknik Informatika ITS yang telah memberi teladan baik kepada mahasiswa-mahasiswi beliau se-Jurusan.
- Ibu Isye Arieshanti, S.Kom, M.Phil, selaku Ketua Program Studi S1 Teknik Informatika yang telah meluangkan segenap waktunya untuk mengatur berjalannya perkuliahan di jurusan Teknik Informatika ITS.
- Bapak Radityo Anggoro, S.Kom, M.Sc selaku koordinator Tugas Akhir yang telah meluangkan waktunya untuk kelancaran pelaksanaan Tugas Akhir di jurusan.
- Bapak Dr.tech. Ir. R.V.Hari Ginardi, M.Sc. selaku dosen wali dan pembimbing akademik penulis selama menempuh masa perkuliahan.
- Teman-teman Administrator di Laboratorium Pemrograman (LP) yaitu Kemal, Aldy, Farhan, Danang, Supri, Aik, Atud, Pundi, Yusro, Hamdi, Karsten, Anggara, Christo, Otniel, Adhe, Dinar, Ratih, dan Pinas yang telah menjadi keluarga di kampus.
- Seluruh staf pengajar dan administrasi jurusan Teknik Informatika ITS yang telah memberikan segala tenaganya untuk memajukan perkembangan jurusan.
- Teman-teman satu jurusan Teknik Informatika ITS angkatan 2011, 2012, dan 2013 yang tidak bisa penulis sebutkan satu persatu namanya.
- Seluruh Keluarga besar Civitas Laboratorium Teknik Informatika.
- Tim PPDB yang selalu memberikan dukungan dan hiburan di saat penulis mengerjakan Tugas Akhir ini di lantai 4 TC yaitu Uyung, Harum, Yusro, Thiar, Surya.
- Dala Rifat yang selalu memberikan bantuan dan dukungan kepada penulis saat mengerjakan Tugas Akhir ini.
- Serta berbagai pihak yang tidak bisa penulis sebutkan satu per

satu namanya.

Penulis menyadari bahwasanya Tugas Akhir ini masih jauh dari sempurna. Penulis mengharapkan saran dan kritik yang membangun untuk lebih menyempurnakan penelitian ini kedepannya.

Terima Kasih.

Wassalamualaikum

Surabaya, Juni 2015

Ali Ariff

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE SUMBER

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Online *judge* adalah suatu sistem untuk menguji suatu program pada kontes pemrograman. Sistem ini dapat ini mengompilasi, mengeksekusi kode program, dan mengujinya dengan data yang sudah disiapkan. Kode yang dikirimkan dapat dijalankan dengan batasan seperti *time limit*, *memory limit*, batasan keamananan dan sebagainya. Keluaran dari kode akan diambil oleh sistem untuk dibandingkan dengan standar keluaran atau kunci jawaban yang sudah ada sebelumnya. Kemudian sistem akan mengembalikan hasilnya untuk ditampilkan keputusan benar atau salahnya suatu program [1].

Penggunaan *online judge* khususnya pada perkuliahan di Teknik Informatika ITS digunakan untuk mata kuliah Dasar Pemrograman, Algoritma dan Struktur Data, Perancangan dan Analisis Algoritma 1 serta Perancangan dan Analisis Algoritma 2. Mata kuliah tersebut lebih menekankan pada pembelajaran tentang algoritma dan struktur data dengan studi kasus berbagai masalah pemrograman. Namun *online judge* ini juga berpotensi untuk digunakan pada mata kuliah lain yaitu Pemrograman Jaringan yang mengharuskan mahasiswa untuk mengerjakan soal-soal pemrograman jaringan yang mayoritas klien-*server*. Dengan pemanfaatan *online judge* pada mata kuliah pemrograman jaringan ini mahasiswa dapat lebih mudah belajar dengan berbagai macam studi kasus soal dan membuat dosen atau pengajar lebih mudah menilai serta melihat hasil dari pekerjaan mahasiswa.

Oleh karena itu, pada Tugas Akhir ini dilakukan pembuatan sistem *online judge* untuk mata kuliah Pemrograman Jaringan, di mana sistem akan menerima jawaban dari mahasiswa yang selanjutnya akan dikompilasi atau dijalankan di suatu lingkungan terisolasi, lalu

hasilnya akan dibandingkan dengan kunci jawaban yang sebelumnya telah didefinisikan oleh dosen. Hasil akhir dari sistem ini adalah poin-poin kasus uji yang berhasil diselesaikan oleh mahasiswa.

1.2 Rumusan Masalah

Berikut beberapa hal yang menjadi rumusan masalah dalam Tugas Akhir ini:

- Bagaimana cara menerapkan konsep sistem *online judge* untuk mata kuliah Pemrograman Jaringan yang dapat menerima dan menilai kode program yang dikirimkan oleh mahasiswa?
- Bagaimana cara mengimplementasikan suatu sistem *online judge* yang dapat diterapkan pada studi kasus mata kuliah Pemrograman Jaringan?
- Bagaimana cara mengintegrasikan sistem *online judge* untuk pengambilan data peserta kelas dan penilaian dengan eLearning Management System (Moodle)?

1.3 Batasan Masalah

Batasan masalah pada Tugas Akhir ini adalah sebagai berikut:

- Program yang diterima oleh sistem hanya yang berbasis klien-*server*.
- Bahasa pemrograman yang dapat diterima oleh sistem ini adalah C/C++ (GCC/G++ 4.9.2), Java (JRE 1.7.0), dan Python 2.7.9.
- Protokol yang diakomodasi oleh sistem ini adalah HTTP, FTP, SMTP, IMAP, dan POP3.

1.4 Tujuan

Tujuan dari Tugas Akhir ini adalah menghasilkan sistem *online judge* yang dapat digunakan dalam mata kuliah Pemrograman Jaringan yang memanfaatkan virtualisasi berbasis Docker.

1.5 Manfaat

Dengan dibuatnya Tugas Akhir ini akan memberikan konsep pembelajaran baru pada mata kuliah Pemrograman Jaringan karena me-

nawarkan suatu kemudahan bagi dosen maupun mahasiswa untuk dapat menilai kemampuan dalam menyelesaikan berbagai macam permasalahan pada pemrograman jaringan serta mempercepat proses penilaian dosen terhadap program yang dibuat oleh mahasiswa.

1.6 Metodologi

Langkah dan metode yang dilakukan untuk mengerjakan Tugas Akhir ini dijelaskan sebagai berikut.

- **Penyusunan Proposal Tugas Akhir**
  Penyusunan proposal Tugas Akhir dilaksanakan untuk merumuskan masalah serta melakukan penetapan desain dasar sistem yang akan dikembangkan dalam pelaksanaan Tugas Akhir ini.
- **Studi Literatur**
  Untuk membantu proses pengerjaan Tugas Akhir, diperlukan studi lebih lanjut mengenai penggunaan komponen-komponen terkait dengan sistem yang akan dibangun.
- **Desain dan Perancangan**
  Dalam rangka memerinci lebih jauh mengenai bagaimana memanfaatkan komponen-komponen sistem untuk membangun sistem secara utuh, diperlukan proses desain dan perancangan dari sistem. Hasil analisis dan desain kemudian ditetapkan menjadi rancangan dasar implementasi sistem.
- **Implementasi Sistem**
  Hasil analisis dan desain kemudian diimplementasikan melalui komponen-komponen perangkat lunak pendukung.
- **Uji Coba dan Evaluasi**
  Untuk mengukur kemampuan sistem untuk menangani pengguna dari segi fungsionalitas dan performa, dilakukan proses uji coba dan evaluasi untuk mengetahui sejauh mana sistem dapat berjalan sesuai dengan yang diharapkan.

1.7 Sistematika Laporan

Buku Tugas Akhir ini terdiri dari beberapa bab yang dijelaskan sebagai berikut:

- **Bab 1. Pendahuluan**. Bab ini berisikan latar belakang, rumusan masalah, tujuan, manfaat, metodologi dan sistematika penulisan yang digunakan sebagai dasar penyusunan Tugas Akhir.
- **Bab 2. Tinjauan Pustaka**. Bab ini berisikan teori penunjang yang berhubungan dengan Tugas Akhir.
- **Bab 3. Desain dan Perancangan**. Bab ini membahas desain dasar dari sistem dan perangkat lunak yang akan dirancang sebagai bagian dari pengerjaan Tugas Akhir.
- **Bab 4. Implementasi**. Bab ini membahas hasil implementasi dari rancangan sistem beserta tekniknya.
- **Bab 5. Uji Coba dan Evaluasi**. Bab ini membahas mengenai teknik uji coba dan hasil keluarannya sebagai bahan evaluasi terhadap hasil Tugas Akhir.
- **Bab 6. Penutup**. Bab ini berisi kesimpulan dari hasil uji coba serta saran untuk pengembangan Tugas Akhir selanjutnya.

BAB 2

LANDASAN TEORI

2.1 Online Judge

Online Judge adalah suatu sistem untuk menguji suatu program pada kontes pemrograman. Sistem ini dapat ini mengompilasi, mengeksekusi kode program, dan mengujinya dengan data yang sudah disiapkan. Kode yang dikirimkan dapat dijalankan dengan batasan seperti *time limit*, *memory limit*, batasan keamananan dan sebagainya. Keluaran dari kode akan diambil oleh sistem untuk dibandingkan dengan standar keluaran atau kunci jawaban yang sudah ada sebelumnya. Kemudian sistem akan mengembalikan hasilnya untuk ditampilkan keputusan benar atau salahnya suatu program [1].

2.2 Docker

Docker adalah sebuah *platform* terbuka untuk pengembang, sysadmin atau setiap orang yang bertujuan menggunakan sebuah *platform* untuk membangun, mendistribusikan, dan menjalankan aplikasi pada laptop, data *center*, virtual mesin maupun *cloud*. Docker membuat proses pemaketan aplikasi bersama komponennya (*dependencies*) secara cepat dalam sebuah *container* yang terisolasi, sehingga dapat dijalankan dalam infrastruktur lokal atau *cloud* tanpa melakukan perubahan maupun konfigurasi lagi pada *container* selama *host* menjalankan Docker Engine [2].

Docker menggunakan arsitektur klien-*server*. Docker klien mengirimkan permintaan berupa perintah kepada Docker *daemon* untuk membangun, mendistribusikan, dan menjalankan *container* Docker.

- *Docker Daemon*, Docker *daemon* berjalan pada mesin *host*. Pengguna tidak berinteraksi secara langsung dengan Docker *daemon* tapi melalui Docker klien.

- *Docker Klien*, Docker klien merupakan bentuk dari Docker binari dan merupakan antarmuka pengguna utama untuk Docker, yang menerima perintah dari pengguna dan berkomunikasi dengan Docker *daemon*.

Istilah yang menjadi komponen utama Docker [3]:

- *Docker Images*, Docker Image merupakan sebuah *template* yang bersifat *read*-only. Contohnya sebuah *image* yang berisi sistem operasi Ubuntu dengan Apache dan aplikasi *web* yang telah diinstal. Image ini digunakan untuk menjalankan *container*. Docker menyediakan cara yang sederhana untuk membangun *image* baru atau merubah *image* yang sudah ada. Jika melihat Docker Index akan ditemukan banyak *image* yang telah dibuat oleh pengguna lain yang dapat digunakan sebagai *image* dasar.
- *Docker Container*, Docker Container merupakan sebuah *image* bersifat *read-write* yang berjalan di atas *image*. Docker menggunakan sistem *union-file* sebagai sistem *file backend container*-nya, di mana setiap perubahan yang disimpan pada *container* akan menyebabkan terbentuknya layer baru di atas *image* dasar.
- *Docker Registry*, Docker Registry adalah tempat penyimpanan (*public* atau *private*) di mana dapat mengunggah dan mengunduh *image*. Registry *public* Docker disebut dengan Docker Hub

2.3 Moodle

Moodle adalah *platform* pembelajaran yang didesain untuk pengajar, administrator, dan pembelajar dengan ketahanan, keamanan, dan sistem yang terintegrasi untuk membuat lingkungan pembelajaran yang baik [4].

Learning Tools Interoperability (LTI) adalah interface untuk dapat memungkinkan aplikasi di luar moodle dapat terintegrasi dengan sistem Moodle [5].

2.4 NodeJS

NodeJS adalah lingkungan JavaScript yang berjalan pada sisi *server*. NodeJS berbasis pada implementasi Google Runtime V8 Engine. V8 Engine dan NodeJS biasa diimplementasikan dengan C dan C++, berfokus pada performa dan konsumsi memori yang rendah, tetapi V8 hanya mendukung JavaScript pada peramban terutama Google Chrome. Tujuan utama NodeJS mengarah ke proses *server* yang berjalan sangat lama. Tidak seperti kebanyakan lingkungan modern yang lain, proses NodeJS bukanlah *multithreading* untuk dapat mendukung eksekusi proses secara bersamaan. NodeJS berbasis pada model *asynchronous I/O event* di mana proses *server* adalah *single threaded daemon* [6].

2.5 MEAN Framework

MEAN Framework atau MEAN Stack merupakan kerangka kerja pemrograman yang terdiri dari komponen MongoDB, ExpressJS, AngularJS, dan NodeJS namun pada Tugas Akhir ini penulis tidak menggunakan MongoDB melainkan MySQL dikarenakan penulis membutuhkan *relational database* [7]. Sehingga komponen-komponen dalam MEAN menjadi:

- MySQL sebagai perangkat *server* basis data.
- ExpressJS merupakan kerangka kerja pengembangan aplikasi Web berbasis Javascript dari sisi *server*.
- AngularJS merupakan kerangka kerja pengembangan aplikasi Javascript dari sisi klien.
- NodeJS merupakan kerangka kerja pemrograman yang membawahi komponen ExpressJS.

BAB 3

DESAIN DAN PERANCANGAN

Online judge pada umumnya merupakan suatu sistem untuk menguji kode sumber dengan kasus uji yang sudah didefinisikan untuk mengetahui hasil dari kode sumber tersebut. Fitur-fitur yang harus ada dalam sebuah *online judge* adalah:

- Sistem dapat membuat pengguna baru dan memanipulasinya.
- Sistem dapat membuat soal dan kasus ujinya serta memanipulasinya.
- Sistem dapat menerima jawaban berupa kode sumber.
- Sistem dapat menampilkan hasil dari setiap kiriman jawaban.

Dari fitur-fitur utama tersebut maka dibuatlah analisis, perancangan dan implementasi yang dibahas pada bab ini.

3.1 Kasus Penggunaan

Gambar 3.1 menampilkan kasus penggunaan sistem secara umum dengan penjelasan tertera pada Tabel 3.1

Tabel 3.1: Daftar Kasus Penggunaan Sistem

No	Nama	Aktor	Deskripsi
UC01	Manajemen Soal	Dosen	Soal-soal memiliki tipe sebagai *server* atau klien serta atribut-atribut pelengkap soal seperti judul, deskripsi, dsb.

No	Nama	Aktor	Deskripsi
UC02	Manajemen Kelas	Dosen	Kelas diisi dengan peserta kelas yaitu mahasiswa yang terdaftar di sistem dan juga soal-soal yang sudah dibuat dapat dimasukkan ke dalam kelas.
UC03	Mengunduh Hasil Penilaian	Dosen	Hasil penilaian akan berupa tabel pemeringkatan peserta kelas diurut berdasarkan nilai tertinggi dan dapat diunduh ke dalam bentuk csv.
UC04	Melihat Soal	Mahasiswa	Mahasiswa yang sudah terdaftar di suatu kelas dapat membaca soal-soal apa saja yang ada di dalam kelas tersebut.
UC05	Mengirimkan Jawaban	Mahasiswa	Jawaban dari mahasiswa berupa kode sumber dengan ekstensi .c/cpp, .py, atau .java.
UC06	Melihat Riwayat Jawaban	Mahasiswa	Riwayat serta skor yang didapatkan dari jawaban yang telah dikirimkan.
UC07	Melihat Peringkat	Mahasiswa	Pemeringkatan peserta kelas dalam kelas yang diikuti.
UC08	Manajemen Pengguna	Admin	Pengguna baru yang akan menggunakan sistem dapat didaftarkan oleh admin.

Gambar 3.1: Diagram Kasus Penggunaan Sistem

3.2 Deskripsi Fitur

Secara umum, fitur sistem dibagi menjadi tiga: fitur untuk dosen, fitur untuk mahasiswa dan fitur untuk administrator.

3.2.1 Fitur untuk Dosen

Berikut adalah fitur yang tersedia untuk dosen:

- Dosen dapat membuat soal baru, menghapus, maupun mengubah soal yang telah dibuat.
- Dosen dapat membuat kelas baru, menghapus, maupun mengubah peserta kelas yang dibuat.
- Dosen dapat melihat dan mengunduh hasil penilaian dari tiap kelas.

3.2.2 Fitur untuk Mahasiswa

Berikut adalah fitur yang disediakan untuk mahasiswa:

- Mahasiswa dapat melihat soal yang ada untuk kelas yang diikuti oleh mahasiswa yang bersangkutan.
- Mahasiswa dapat mengirimkan jawaban berupa kode sumber.
- Mahasiswa dapat melihat riwayat dari setiap jawaban yang telah dikirimkan.

- Mahasiswa dapat melihat pemeringkatan dari peserta di kelas yang diikuti.

3.2.3 Fitur untuk Administrator

Berikut adalah fitur yang disediakan untuk administrator:

- Admin dapat membuat pengguna baru, menghapus, maupun mengubah pengguna yang sudah ada.

3.3 Arsitektur Sistem

Pada sub-bab ini, dibahas mengenai tahap analisis dan desain dari sistem yang akan dibangun.

3.3.1 Desain Umum Sistem

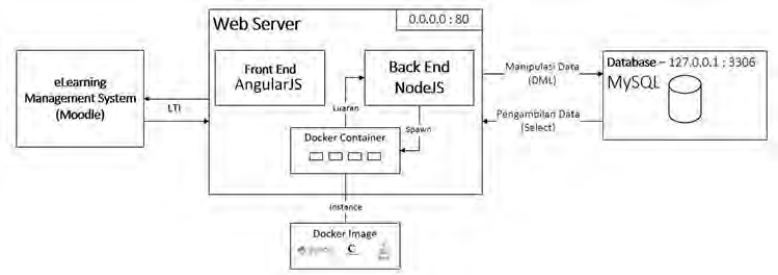
Sistem dibangun dalam beberapa komponen umum yang terkait satu sama lain, yaitu:

- **Backend** sebagai pengendali yang menghubungkan basis data dengan *frontend*.
- **Frontend** adalah antarmuka dari sistem yang berbasis *web*.
- **Docker** sebagai lingkungan virtual yang dipanggil dengan bantuan Backend. Digunakan sebagai lingkungan terisolasi bagi kode sumber yang dikirimkan oleh mahasiswa.
- **Basis Data** sebagai tempat menyimpan segala macam data transaksi maupun data utama dari sistem.
- **Moodle** adalah aplikasi *elearning* yang dapat berinteraksi dengan sistem yang akan dibangun ini melalui Learning Tools Interoperability (LTI).

Gambar 3.2 menampilkan diagram arsitektur hubungan antara komponen.

3.3.2 Desain Rinci Backend

Backend dibuat dengan NodeJS dan ExpressJS serta menggunakan konsep RESTful API. Backend akan menangani segala bentuk operasi ke basis data yang dikirimkan dari *frontend* dan juga akan memberikan data yang dibutuhkan ke *frontend*. Backend juga akan memanggil Docker guna mengeksekusi jawaban yang dikirimkan ma-



Gambar 3.2: Desain Sistem Secara Umum

hasiswa. Tabel 3.2 menjelaskan daftar rute akses REST API yang ada pada Backend. Ada empat jenis rute pada API ini: rute yang bisa diakses tanpa autentikasi, rute yang hanya bisa diakses oleh admin, rute yang bisa diakses oleh dosen, rute yang bisa diakses oleh mahasiswa.

Tabel 3.2: Daftar Rute REST API pada Backend

No	**Rute**	**Metode**	**Hak Akses**	**Aksi**
1	/users/session	POST	Tidak ada	Melakukan autentikasi `username` dan `password`
2	/lti/class /*class-id*/problem /*problem-id*	POST	Tidak ada	Melakukan autentikasi dengan memanfaatkan Moodle LTI
3	/users	GET, POST, PUT, DELETE	Admin	Melakukan manipulasi data pengguna sistem

No	Rute	Metode	Hak Akses	Aksi
4	/problems	GET, POST, PUT, DELETE	Dosen	Melakukan manipulasi soal
5	/problems /*problem-id*/testcase	GET, POST, PUT, DELETE	Dosen	Melakukan manipulasi kasus uji pada suatu soal
6	/classes	GET, POST, PUT, DELETE	Dosen	Melakukan manipulasi data kelas
7	/classes/*class-id*/problems	POST, PUT, DELETE	Dosen	Melakukan manipulasi soal yang ada di suatu kelas
8	/classes/*class-id*/users	POST, DELETE	Dosen	Melakukan manipulasi peserta kelas yang ada di suatu kelas
9	/classes/*class-id*/rank	GET	Dosen	Melihat pemeringkatan peserta kelas di suatu kelas
10	/myclass /*class-id*	GET	Mahasiswa	Melihat soal-soal yang ada di suatu kelas
11	/myclass /*class-id*/rank	GET	Mahasiswa	Melihat pemeringkatan peserta kelas di suatu kelas

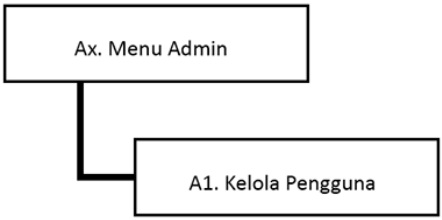
No	Rute	Metode	Hak Akses	Aksi
12	/myclass /*class-id*/problem /*problem-id*	GET, POST	Mahasiswa	Melihat suatu soal di suatu kelas dan mengirimkan jawaban ke soal tersebut

Proses autentikasi diimplementasikan menggunakan pustaka Passport (`http://passportjs.org/`). Proses autentikasi melalui Moodle LTI juga memanfaatkan pustaka pendukung Passport-LTI (`https://github.com/civitaslearning/passport-lti/`).

3.3.3 Desain Rinci Frontend

Bagian Frontend dibuat menggunakan pustaka AngularJS (`https://angularjs.org/`). Tabel 3.3 menjelaskan rute HTTP yang ada pada Frontend.

Peta situs untuk halaman admin terdapat pada gambar 3.3, sementara halaman dosen terdapat pada gambar 3.4. Peta situs halaman mahasiswa terdapat pada gambar 3.5.



Gambar 3.3: Peta Situs Halaman Admin pada Frontend

3.3.4 Desain Rinci Docker

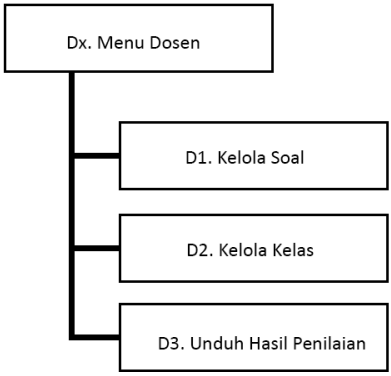
Docker akan digunakan untuk membuat lingkungan terisolasi guna mengeksekusi kode sumber yang dikirimkan oleh mahasiswa.

Tabel 3.3: Daftar Rute pada Frontend

No	Rute	Metode	Hak Akses	Aksi
1	/profile	GET	Admin, Dosen, Mahasiswa	Menampilkan halaman profil pengguna
2	/users	GET	Admin	Menampilkan halaman kelola pengguna
3	/problem	GET	Dosen	Menampilkan halaman kelola soal
4	/class	GET	Dosen	Menampilkan halaman kelola kelas
5	/myclass	GET	Mahasiswa	Menampilkan halaman kelas dan soal untuk mahasiswa
6	/myresult	GET	Mahasiswa	Menampilkan halaman riwayat jawaban dan hasil kiriman dari mahasiswa

Docker akan dipanggil oleh Backend setiap ada mahasiswa yang mengirimkan jawabannya untuk suatu soal tertentu. Diperlukan beberapa komponen penting untuk dapat mengeksekusi atau mengompilasi kode sumber yang dikirimkan di dalam lingkungan Docker karenanya dibuatlah suatu citra cakram Docker baru dengan nama NetOJ (`https://registry.hub.docker.com/u/etni35/netoj/`) yang berisi:

- Java JRE 1.7.0
- GCC/G++ 4.9.2
- Python 2.7.9



Gambar 3.4: Peta Situs Halaman Dosen pada Frontend

Ketiga komponen tersebut berguna untuk menjalankan kode sumber sesuai dengan ektensinya. Citra cakram Docker ini berbasis dari citra cakram Docker lain yang ada di *registry hub* Docker yaitu java7 (`https://registry.hub.docker.com/u/library/java/`). Citra cakram tersebut menjadi basis dasar untuk membuat citra cakram NetOJ, lalu menambah atau menginstal *compiler* lain yaitu GCC/G++ dan Python. Citra cakram ini menggunakan sistem operasi Debian 8.0.

3.3.5 Desain Rinci Basis Data

Basis Data dari sistem menggunakan MySQL. Struktur basis data dari sistem terdapat pada gambar 3.6. Tabel 3.4 menjelaskan detail struktur basis data pada sistem.

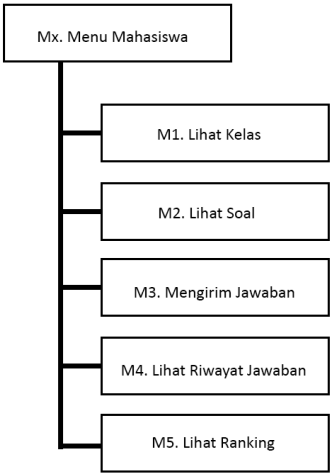
Tabel 3.4: Struktur Basis Data

No	Nama Tabel	Deskripsi
1	Users	Menyimpan data pengguna.

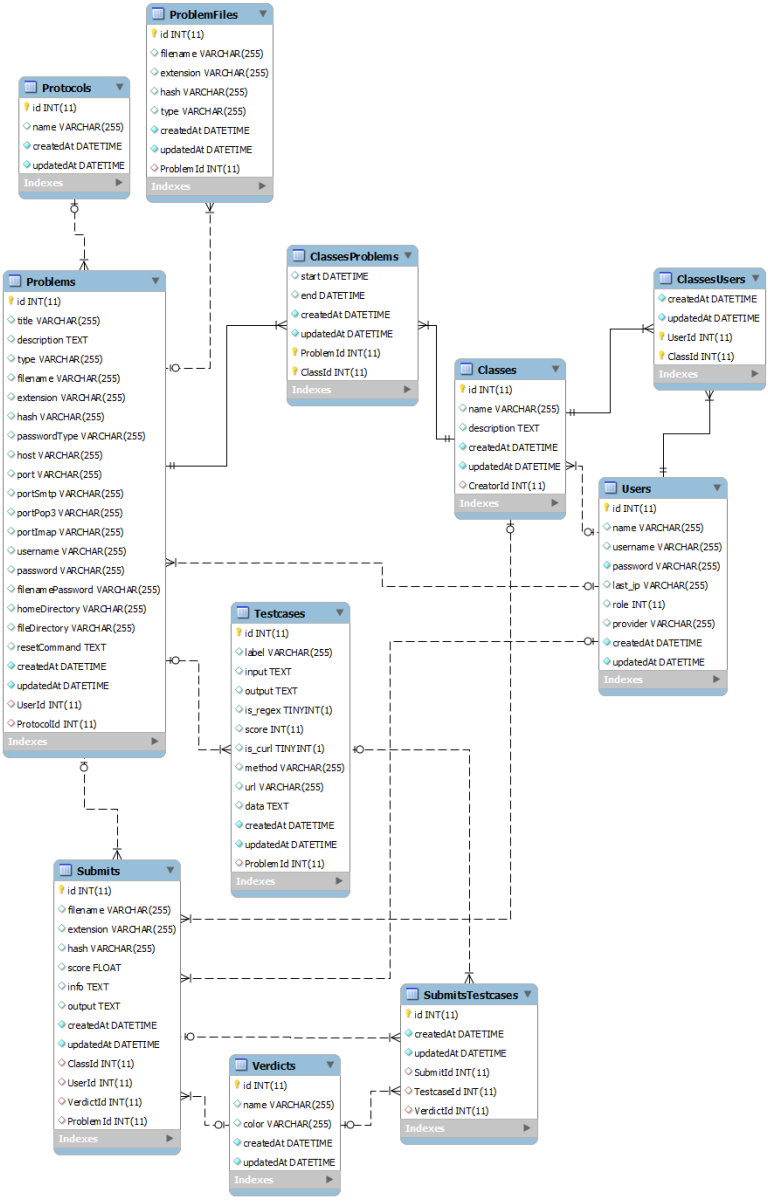
No	Nama Tabel	Deskripsi
2	Problems	Menyimpan data soal.
3	ProblemFile	Menyimpan data berkas pendukung soal.
4	Testcases	Menyimpan data kasus uji soal.
5	Protocols	Data *master* protokol.
6	Classes	Menyimpan data kelas.
7	ClassesProblems	Menyimpan data soal yang ada di kelas.
8	ClassesUsers	Menyimpan data peserta kelas.
9	Verdicts	Data *master* hasil keputusan soal.
10	Submits	Menyimpan data jawaban yang dikirimkan.
11	SubmitsTestcases	Menyimpan data hasil keputusan tiap jawaban terhadap kasus uji yang diujikan.

3.3.6 Desain Rinci Moodle LTI

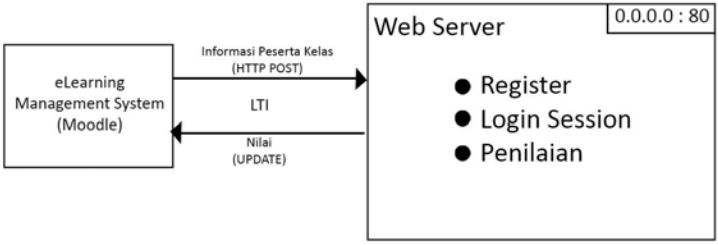
Moodle adalah aplikasi *elearning* yang sudah umum digunakan dalam proses pembelajaran. Sistem yang dibuat ini dapat berinteraksi dengan Moodle melalui Learning Tools Interoperability (LTI) yang ada pada Moodle. Interaksi terjadi melalui rute API `lti/class/class-id/problem/problem-id`. Interaksi yang terjadi adalah mahasiswa yang sudah terdaftar di Moodle dapat langsung masuk ke dalam sistem ini tanpa perlu *login* ke sistem serta nilai yang didapatkan dari sistem ini akan langsung masuk ke dalam sistem penilaian Moodle. Gambar 3.7 menampilkan desain rinci komponen Moodle dan sistem ini.



Gambar 3.5: Peta Situs Halaman Mahasiswa pada Frontend



Gambar 3.6: Struktur Basis Data



Gambar 3.7: Desain Rinci Moodle LTI

BAB 4

IMPLEMENTASI

Bab ini membahas implementasi perancangan sistem secara rinci. Pembahasan dilakukan untuk setiap komponen yang sudah dijelaskan pada bab sebelumnya.

4.1 Lingkungan Implementasi

Lingkungan implementasi dan pengembangan dilakukan menggunakan komputer PC dengan spesifikasi AMD Phenom II x4 965 Black Edition dengan memori 4GB. Perangkat lunak yang digunakan dalam proses pengembangan antara lain:

- Sistem operasi Ubuntu Linux 14.04.2 LTS.
- Editor teks Sublime Text 3.
- Git 1.9.1 untuk pengelolaan versi kode program.
- NodeJS 0.10.25 untuk kerangka kerja pemrograman.
- Grunt 0.4.5 untuk *task runner*.
- Docker 1.6.2 untuk uji coba citra cakram Docker.
- Paket TEXlive untuk penulisan buku tugas akhir.
- Peramban *web* Mozilla Firefox.

4.2 Rincian Implementasi Backend

Implementasi Backend dibagi berdasarkan rute REST API yang sudah dijelaskan pada Tabel 3.2. Setiap rute dasar dibuat dalam pengendali atau *controller* terpisah.

4.2.1 Pengendali `/users`

Pengendali ini berfungsi untuk melakukan manipulasi data pengguna. Pengendali ini juga berfungsi untuk mengautentikasi pengguna yang akan masuk ke dalam sistem. Proses autentikasi ini dilakukan melalui sub-rute pengendali `/users/session`. Implementasi untuk proses autentikasi tersebut menggunakan pustaka Passport (`http://passportjs.org/`). Rute pengendali `/users` dapat diakses melalui metode GET, POST, PUT dan DELETE sesuai yang dijelaskan pada Tabel 4.1.

Tabel 4.1: Implementasi Pengendali `/users`

No	Rute	Hak Akses	Masukan	Luaran	Langkah Proses
1	GET /	A-D	-	Data pengguna (n)	1. Kembalikan data semua pengguna
2	POST /	A	Data pengguna (1)	Data pengguna (1)	1. Lakukan pembuatan pengguna baru 2. Kembalikan data pengguna yang sudah diberi *userId*

Keterangan: *A: Admin, D: Dosen, M: Mahasiswa, (1) Data Tunggal, (n) Data Jamak*

No	Rute	Hak Akses	Masukan	Luaran	Langkah Proses
3	POST `/session`	-	`Username`, `Password`	Data pengguna (1)	1. Jika `Username` dan `Password` terdaftar • Berikan data pengguna dan *session*
4	PUT `/userId`	A-D-M	Data pengguna (1)	Data pengguna (1)	1. Apakah pengguna A? • Ubah data pengguna sesuai yang diminta • Kembalikan data pengguna hasil pengubahan 2. Jika pengguna D atau M, maka: • Apakah pengguna berhak merubah data dengan *`userId`* tersebut? – Jika iya, ubah data pengguna sesuai yang diminta lalu kembalikan data pengguna hasil pengubahan – Jika tidak, kirim pesan galat

Keterangan: *A: Admin, D: Dosen, M: Mahasiswa, (1) Data Tunggal, (n) Data Jamak*

No	Rute	Hak Akses	Masukan	Luaran	Langkah Proses
5	DELETE `/userId`	A	-	Data pengguna (1)	1. Hapus data pengguna sesuai yang diminta 2. Kembalikan data pengguna hasil penghapusan

Keterangan: *A: Admin, D: Dosen, M: Mahasiswa, (1) Data Tunggal, (n) Data Jamak*

4.2.2 Pengendali `/lti`

Pengendali ini khusus berfungsi untuk menangani autentikasi lewat Moodle yang memanfaatkan Learning Tools Interoperability (LTI) pengendali ini bertujuan memvalidasi data POST yang dikirim dari Moodle untuk selanjutnya diberikan akses langsung ke dalam sistem ini tanpa *login* jika benar data yang dikirim dari Moodle tersebut valid. Proses validasi ini memanfaatkan pustaka pendukung Passport-LTI (`https://github.com/civitaslearning/passport-lti/`). Tabel 4.2 menjelaskan implementasi sub-rute dari pengendali rute `/lti`.

Tabel 4.2: Implementasi Pengendali `/lti`

No	Rute	Hak Akses	Masukan	Luaran	Langkah Proses
1	POST `/class` `/classId` `/problem` `/problemId`	-	Data POST LTI	Data pengguna (1)	1. Jika data POST LTI valid dan `Username` terdaftar di sistem • Berikan data pengguna dan *session* 2. Jika data POST LTI valid dan `Username` tidak terdaftar di sistem • Buat data pengguna baru dengan `Username` dari data POST LTI serta password sama dengan `Username` • Berikan data pengguna yang baru dibuat dan *session*

Keterangan: *A: Admin, D: Dosen, M: Mahasiswa, (1) Data Tunggal, (n) Data Jamak*

4.2.3 Pengendali `/problems`

Pengendali ini berfungsi untuk melakukan manipulasi data soal. Di dalam pengendali ini juga terdapat sub-rute pengendali untuk melakukan manipulasi data kasus uji untuk tiap soal yang bisa diakses melalui `/problems/`*`problem-id`*`/testcase`. Tabel 4.3 menjelaskan implementasi sub-rute dari pe-

ngendali rute `/problems`. Tabel 4.4 menjelaskan implementasi sub-rute dari pengendali rute `/problems/problem-id/testcase`.

Tabel 4.3: Implementasi Pengendali `/problems`

No	Rute	Hak Akses	Masukan	Luaran	Langkah Proses
1	GET /	D	-	Data soal (n)	1. Kembalikan data semua soal
2	GET */problemId*	D	-	Data soal (1)	1. Kembalikan data soal *problemId*
3	POST /	D	Data soal (1)	Data soal (1)	1. Lakukan pembuatan soal baru 2. Kembalikan data soal yang sudah diberi *problemId*
4	PUT */problemId*	D	Data soal (1)	Data soal (1)	1. Ubah data soal sesuai yang diminta 2. Kembalikan data soal hasil pengubahan
5	DELETE */problemId*	D	-	Data soal (1)	1. Hapus data soal sesuai yang diminta 2. Kembalikan data soal hasil penghapusan

Keterangan: *A: Admin, D: Dosen, M: Mahasiswa, (1) Data Tunggal, (n) Data Jamak*

Tabel 4.4: Implementasi Pengendali `/problems/problem-id/testcase`

No	Rute	Hak Akses	Masukan	Luaran	Langkah Proses
1	POST /	D	Data kasus uji (1)	Data kasus uji (1)	1. Lakukan pembuatan kasus uji baru pada soal yang diminta 2. Kembalikan data kasus uji yang baru dibuat
2	PUT */testcaseId*	D	Data kasus uji (1)	Data kasus uji (1)	1. Ubah data kasus uji sesuai yang diminta 2. Kembalikan data kasus uji hasil pengubahan
3	DELETE */testcaseId*	D	-	Data kasus uji (1)	1. Hapus data kasus uji sesuai yang diminta 2. Kembalikan data kasus uji hasil penghapusan

Keterangan: *A: Admin, D: Dosen, M: Mahasiswa, (1) Data Tunggal, (n) Data Jamak*

4.2.4 Pengendali `/classes`

Pengendali ini berfungsi untuk melakukan manipulasi data kelas. Di dalam pengendali ini juga terdapat beberapa sub-rute pengendali yaitu:

- `/classes/class-id/problems` berfungsi melakukan manipulasi data soal yang ada di kelas.

- `/classes/class-id/users` berfungsi melakukan manipulasi data peserta kelas.
- `/classes/class-id/rank` berfungsi mendapatkan data pemeringkatan kelas.

Tabel 4.5 menjelaskan implementasi sub-rute dari pengendali rute `/classes`. Tabel 4.6 menjelaskan implementasi sub-rute dari pengendali rute `/classes/class-id/problems`. Tabel 4.7 menjelaskan implementasi sub-rute dari pengendali rute `/classes/class-id/users`. Tabel 4.8 menjelaskan implementasi sub-rute dari pengendali rute `/classes/class-id/rank`.

Tabel 4.5: Implementasi Pengendali `/classes`

No	Rute	Hak Akses	Masukan	Luaran	Langkah Proses
1	GET /	D	-	Data kelas (n)	1. Kembalikan data semua kelas
2	GET /*classId*	D	-	Data kelas (1)	1. Kembalikan data kelas *classId*
3	POST /	D	Data kelas (1)	Data kelas (1)	1. Lakukan pembuatan kelas baru 2. Kembalikan data kelas yang sudah diberi *classId*
4	PUT /*classId*	D	Data kelas (1)	Data kelas (1)	1. Ubah data kelas sesuai yang diminta 2. Kembalikan data kelas hasil pengubahan

Keterangan: *A: Admin, D: Dosen, M: Mahasiswa, (1) Data Tunggal, (n) Data Jamak*

No	Rute	Hak Akses	Masukan	Luaran	Langkah Proses
5	DELETE `/classId`	D	-	Data kelas (1)	1. Hapus data kelas sesuai yang diminta 2. Kembalikan data kelas hasil penghapusan

Keterangan: *A: Admin, D: Dosen, M: Mahasiswa, (1) Data Tunggal, (n) Data Jamak*

Tabel 4.6: Implementasi Pengendali `/classes/class-id/ problems`

No	Rute	Hak Akses	Masukan	Luaran	Langkah Proses
1	POST /	D	Data transaksi (1)	Data transaksi (1)	1. Lakukan pembuatan transaksi baru untuk kelas dan soal yang diminta 2. Kembalikan data transaksi yang baru dibuat
2	PUT /	D	Data transaksi (1)	Data transaksi (1)	1. Ubah data transaksi sesuai yang diminta 2. Kembalikan data transaksi hasil pengubahan

Keterangan: *A: Admin, D: Dosen, M: Mahasiswa, (1) Data Tunggal, (n) Data Jamak*

No	Rute	Hak Akses	Masukan	Luaran	Langkah Proses
3	DELETE /	D	Data transaksi (1)	Data transaksi (1)	1. Hapus data transaksi sesuai yang diminta 2. Kembalikan data transaksi hasil penghapusan

Keterangan: *A: Admin, D: Dosen, M: Mahasiswa, (1) Data Tunggal, (n) Data Jamak*

Tabel 4.7: Implementasi Pengendali `/classes/class-id/ users`

No	Rute	Hak Akses	Masukan	Luaran	Langkah Proses
1	POST /	D	Data transaksi (1)	Data transaksi (1)	1. Lakukan pembuatan transaksi peserta kelas baru untuk kelas dan pengguna yang diminta 2. Kembalikan data transaksi peserta kelas yang baru dibuat
2	DELETE /	D	Data transaksi (1)	Data transaksi (1)	1. Hapus data transaksi peserta kelas sesuai yang diminta 2. Kembalikan data transaksi peserta kelas hasil penghapusan

Keterangan: *A: Admin, D: Dosen, M: Mahasiswa, (1) Data Tunggal, (n) Data Jamak*

Tabel 4.8: Implementasi Pengendali `/classes/class-id/ rank`

No	Rute	Hak Akses	Masukan	Luaran	Langkah Proses
1	GET /	D	-	Data peme-ringkatan kelas (n)	1. Kembalikan data pemeringkatan kelas

Keterangan: *A: Admin, D: Dosen, M: Mahasiswa, (1) Data Tunggal, (n) Data Jamak*

4.2.5 Pengendali `/myclass`

Pengendali ini mirip seperti pengendali `/classes` namun khusus untuk Mahasiswa. Pengendali ini berfungsi untuk mendapatkan data kelas yang diikuti Mahasiswa dan juga untuk mengirimkan jawaban.

Tabel 4.9 menjelaskan implementasi sub-rute dari pengendali rute `/myclass`.

Tabel 4.9: Implementasi Pengendali `/myclass`

No	Rute	Hak Akses	Masukan	Luaran	Langkah Proses
1	GET */classId*	M	-	Data kelas (1)	1. Kembalikan data kelas yang diminta

Keterangan: *A: Admin, D: Dosen, M: Mahasiswa, (1) Data Tunggal, (n) Data Jamak*

No	Rute	Hak Akses	Masukan	Luaran	Langkah Proses
2	GET */classId* /rank	M	-	Data peme-ringkatan kelas (n)	1. Kembalikan data pemeringkatan kelas yang diminta
3	GET */classId* /problem */problem-Id*	M	-	Data soal (1)	1. Kembalikan data soal yang diminta
4	POST */classId* /problem */problem-Id*	M	Kode Sumber Jawaban (1)	Data transaksi pengi-riman jawaban (1)	1. Buat data transaksi pengiriman jawaban yang baru 2. Inisialisasi pengeksekusian kode sumber 3. Kembalikan data transaksi pengiriman ja-waban

Keterangan: *A: Admin, D: Dosen, M: Mahasiswa, (1) Data Tunggal, (n) Data Jamak*

4.3 Rincian Implementasi Frontend

Frontend adalah antarmuka untuk pengguna sistem ini yang menghubungkan pengguna dengan REST API pada Backend. Antarmuka pengguna ini diimplementasikan menggunakan pustaka AngularJS. Rute

akses secara umum dibagi seperti pada Tabel 3.3. Sub-bab ini membahas implementasi peta situs dan pengendali antarmuka di sistem ini.

4.3.1 Antarmuka Administrator

Tabel 4.10 menjelaskan mengenai implementasi peta situs (dari gambar 3.3) dari halaman antarmuka administrator pada Frontend serta keterkaitannya dengan halaman pada peta situs dan kasus penggunaan (*use case*) yang diimplementasikan.

Tabel 4.10: Implementasi Peta Situs pada Antarmuka Administrator

No	Halaman	Rute	Use Case	Nama Pengen-dali	Operasi
1	Ax	`/`	-	home	-
2	A1	`/users`	UC08	users	**addUser**: Tambah pengguna. **editUser**: Ubah pengguna. **deleteUser**: Hapus pengguna.

4.3.2 Antarmuka Dosen

Tabel 4.11 menjelaskan mengenai implementasi peta situs (dari gambar 3.4) dari halaman antarmuka dosen pada Frontend serta keterkaitannya dengan halaman pada peta situs dan kasus penggunaan (*use*

case) yang diimplementasikan.

Tabel 4.11: Implementasi Peta Situs pada Antarmuka Dosen

No	Halaman	Rute	Use Case	Nama Pengendali	Operasi
1	Dx	`/`	-	home	-
2	D1	`/problem`	UC01	problem	**delete**: Hapus soal.
3	D1	`/problem /add`	UC01	problemAddEdit	**send**: Tambah soal.
4	D1	`/problem /edit /problem-Id`	UC01	problemAddEdit	**send**: Ubah soal.
5	D1	`/problem /problemId /testcase`	UC01	testCase	**add**: Tambah kasus uji. **edit**: Ubah kasus uji. **delete**: Hapus kasus uji.

No	Halaman	Rute	Use Case	Nama Pengendali	Operasi
6	D2	`/class`	UC02	class	**add**: Tambah kelas. **edit**: Ubah kelas. **delete**: Hapus kelas.

7	D2	`/class`
		`/classId`
		`/problem`
			UC02	manageClass	**getProblems**: Mengambil data seluruh soal.
					addProblem: Tambah soal ke dalam kelas.
					editProblem: Ubah atribut soal di kelas.
					deleteProblem: Hapus soal dari kelas.

8	D2	`/class`
		`/classId`
		`/user`
			UC02	manageClass	**getUsers**: Mengambil data seluruh pengguna.
					addUser: Tambah peserta kelas.
					deleteUser: Hapus peserta kelas.

No	Halaman	Rute	Use Case	Nama Pengendali	Operasi
9	D3	`/class /classId /rank`	UC03	rankClass	**reloadData**: Mengambil data peringkat di kelas.

4.3.3 Antarmuka Mahasiswa

Tabel 4.12 menjelaskan mengenai implementasi peta situs (dari gambar 3.5) dari halaman antarmuka mahasiswa pada Frontend serta keterkaitannya dengan halaman pada peta situs dan kasus penggunaan (*use case*) yang diimplementasikan.

Tabel 4.12: Implementasi Peta Situs pada Antarmuka Mahasiswa

No	Halaman	Rute	Use Case	Nama Pengendali	Operasi
1	Mx	`/`	-	home	-
2	M1	`/myclass`	UC04	myClass	**reloadData**: Mendapatkan data kelas apa saja yang diikuti.
3	M2	`/myclass /classId`	UC04	myClass	**reloadData**: Mendapatkan data soal apa saja yang ada di kelas tersebut.

No	Halaman	Rute	Use Case	Nama Pengendali	Operasi
4	M3	`/myclass /classId /problem /problem-Id`	UC05	myProblem	**send**: Mengirim jawaban untuk soal tersebut.
5	M4	`/myresult`	UC06	myResult	**reloadData**: Mengambil data riwayat jawaban.
6	M5	`/myclass /classId /rank`	UC07	myRank	**reloadData**: Mengambil data peringkat di kelas.

4.4 Rincian Implementasi Docker

Docker dimanfaatkan sebagai lingkungan virtual terisolasi untuk mengeksekusi kode sumber yang dikirimkan. Ini dilakukan agar setiap kode sumber yang tidak terpercaya tersebut ketika dieksekusi tidak memengaruhi ataupun merusak sistem *host*. Docker dipanggil dengan bantuan Backend setiap ada pengiriman jawaban dari Mahasiswa. Setiap ada pengiriman jawaban, Docker akan membuat *instance* baru yang disebut dengan Docker Container. Setiap Docker Container ini adalah bentuk virtual dari citra cakram Docker yang bernama NetOJ. Citra cakram ini berisi *compiler* Java JRE 1.7.0 untuk menjalankan kode sumber berbasis Java, GCC/G++ 4.9.2 untuk megompilasi kode sumber C/C++ dan Python 2.7.9

untuk mengeksekusi kode sumber Python. Setelah berhasil membuat *instance* maka kode sumber akan dijalankan dan hasil keluaran dari kode sumber yang dijalankan akan dibandingkan dengan kunci jawaban dari setiap kasus uji yang sudah didefinisikan oleh Dosen. Hasil dari perbandingan ini adalah keputusan benar atau salahnya kode sumber yang dikirim oleh Mahasiswa dan akan menghasilkan skor untuk jawaban yang dikirim tersebut. Tabel 4.13 menjelaskan detail algoritma pemrosesan kode sumber mulai dari jawaban masuk sampai jawaban dinilai pada masing-masing kasus soal.

Tabel 4.13: Algoritma Pemrosesan Kode Sumber

No	Protokol Soal	Soal Sebagai	Algoritma / Operasi
1	HTTP	*Server*	1. Jika ada kode sumber *server* dari Dosen jalankan. 2. Jalankan kode sumber dari Mahasiswa. 3. Baca keluaran kode sumber dari Mahasiswa. 4. Bandingkan keluaran dengan kunci jawaban yang sudah didefinisikan oleh Dosen. 5. Skor hasil diperbarukan ke dalam basis data.

No	Protokol Soal	Soal Sebagai	Algoritma / Operasi
2	HTTP	Klien	1. Jalankan kode sumber dari Mahasiswa. 2. Baca kasus uji yang didefinisikan oleh Dosen. 3. Jalankan program penguji dengan data kasus uji dari Dosen untuk menguji kode sumber Mahasiswa. 4. Baca keluaran program penguji. 5. Bandingkan keluaran dengan kunci jawaban yang sudah didefinisikan oleh Dosen. 6. Skor hasil diperbarukan ke dalam basis data.

No	Protokol Soal	Soal Sebagai	Algoritma / Operasi
3	FTP	*Server*	1. Hapus seluruh *file* yang ada di *server*. 2. Unggah atau salin seluruh *file* pendukung yang didefinisikan disoal ke tempatnya. 3. Jika ada kode sumber *server* dari Dosen jalankan. 4. Jalankan kode sumber dari Mahasiswa. 5. Jalankan program pengecek untuk membaca isi *server*. 6. Baca keluaran kode sumber dari Mahasiswa dan program pengecek. 7. Bandingkan keluaran dengan kunci jawaban yang sudah didefinisikan oleh Dosen. 8. Skor hasil diperbarukan ke dalam basis data.

No	Protokol Soal	Soal Sebagai	Algoritma / Operasi
4	FTP	Klien	1. Salin seluruh *file* pendukung yang didefinisikan disoal ke tempatnya. 2. Jalankan kode sumber dari Mahasiswa. 3. Baca kasus uji yang didefinisikan oleh Dosen. 4. Jalankan program penguji dengan data kasus uji dari Dosen untuk menguji kode sumber Mahasiswa. 5. Jalankan program pengecek untuk membaca isi *server*. 6. Baca keluaran program penguji dan program pengecek. 7. Bandingkan keluaran dengan kunci jawaban yang sudah didefinisikan oleh Dosen. 8. Skor hasil diperbarukan ke dalam basis data.

No	Protokol Soal	Soal Sebagai	Algoritma / Operasi
5	SMTP	*Server*	1. Hapus seluruh email yang ada di *server*. 2. Salin seluruh *file* pendukung yang didefinisikan disoal ke tempatnya. 3. Jalankan kode sumber dari Mahasiswa. 4. Jalankan program pengecek untuk membaca isi *server*. 5. Baca keluaran kode sumber dari Mahasiswa dan program pengecek. 6. Bandingkan keluaran dengan kunci jawaban yang sudah didefinisikan oleh Dosen. 7. Skor hasil diperbarukan ke dalam basis data.

No	Protokol Soal	Soal Sebagai	Algoritma / Operasi
6	SMTP	Klien	1. Hapus seluruh email yang ada di *server*. 2. Salin seluruh *file* pendukung yang didefinisikan disoal ke tempatnya. 3. Jalankan kode sumber dari Mahasiswa. 4. Baca kasus uji yang didefinisikan oleh Dosen. 5. Jalankan program penguji dengan data kasus uji dari Dosen untuk menguji kode sumber Mahasiswa. 6. Jalankan program pengecek untuk membaca isi *server*. 7. Baca keluaran program penguji dan program pengecek. 8. Bandingkan keluaran dengan kunci jawaban yang sudah didefinisikan oleh Dosen. 9. Skor hasil diperbarukan ke dalam basis data.

No	Protokol Soal	Soal Sebagai	Algoritma / Operasi
7	IMAP	*Server*	1. Hapus seluruh email yang ada di *server*. 2. Salin seluruh *file* pendukung yang didefinisikan disoal ke tempatnya. 3. Jalankan kode sumber dari Mahasiswa. 4. Jalankan program pengecek untuk membaca isi *server*. 5. Baca keluaran kode sumber dari Mahasiswa dan program pengecek. 6. Bandingkan keluaran dengan kunci jawaban yang sudah didefinisikan oleh Dosen. 7. Skor hasil diperbarukan ke dalam basis data.

No	Protokol Soal	Soal Sebagai	Algoritma / Operasi
8	IMAP	Klien	1. Salin seluruh *file* pendukung yang didefinisikan disoal ke tempatnya. 2. Jalankan kode sumber dari Mahasiswa. 3. Baca kasus uji yang didefinisikan oleh Dosen. 4. Jalankan program penguji dengan data kasus uji dari Dosen untuk menguji kode sumber Mahasiswa. 5. Jalankan program pengecek untuk membaca isi *server*. 6. Baca keluaran program penguji dan program pengecek. 7. Bandingkan keluaran dengan kunci jawaban yang sudah didefinisikan oleh Dosen. 8. Skor hasil diperbarukan ke dalam basis data.

No	Protokol Soal	Soal Sebagai	Algoritma / Operasi
9	POP3	*Server*	1. Hapus seluruh email yang ada di *server*. 2. Salin seluruh *file* pendukung yang didefinisikan disoal ke tempatnya. 3. Jalankan kode sumber dari Mahasiswa. 4. Jalankan program pengecek untuk membaca isi *server*. 5. Baca keluaran kode sumber dari Mahasiswa dan program pengecek. 6. Bandingkan keluaran dengan kunci jawaban yang sudah didefinisikan oleh Dosen. 7. Skor hasil diperbarukan ke dalam basis data.

No	Protokol Soal	Soal Sebagai	Algoritma / Operasi
10	POP3	Klien	1. Salin seluruh *file* pendukung yang didefinisikan disoal ke tempatnya. 2. Jalankan kode sumber dari Mahasiswa. 3. Baca kasus uji yang didefinisikan oleh Dosen. 4. Jalankan program penguji dengan data kasus uji dari Dosen untuk menguji kode sumber Mahasiswa. 5. Jalankan program pengecek untuk membaca isi *server*. 6. Baca keluaran program penguji dan program pengecek. 7. Bandingkan keluaran dengan kunci jawaban yang sudah didefinisikan oleh Dosen. 8. Skor hasil diperbarukan ke dalam basis data.

4.5 Rincian Implementasi Basis Data

Pada sub-bab ini akan dibahas mengenai detail struktur basis data yang digunakan sistem. Detail meliputi nama atribut, tipe data dan deskripsi masing-masing kolom yang ada di setiap tabel basis data. Tabel 4.14

menjelaskan detail setiap kolom yang ada di sistem.

Tabel 4.14: Atribut pada Basis Data

No	Nama Tabel	Nama Kolom	Tipe Data	Deskripsi
1	Users	id	Integer	*Primary key* tabel
2	Users	name	Varchar(255)	Nama pengguna
3	Users	username	Varchar(255)	Username pengguna
4	Users	password	Varchar(255)	Password terenkripsi pengguna
5	Users	last_ip	Varchar(255)	Alamat IP terakhir pengakses akun
6	Users	role	Integer	Peran pengguna (1. Admin, 2. Dosen, 3. Mahasiswa)
7	Users	provider	Varchar(255)	Informasi *login* asal dari lokal atau LTI
8	Problems	id	Integer	*Primary key* tabel
9	Problems	title	Varchar(255)	Judul soal
10	Problems	description	Text	Deskripsi soal
11	Problems	type	Varchar(255)	Tipe soal (klien atau *server*)
12	Problems	filename	Varchar(255)	Nama kode sumber soal
13	Problems	extension	Varchar(255)	Extensi kode sumber soal

No	Nama Tabel	Nama Kolom	Tipe Data	Deskripsi
14	Problems	hash	Varchar(255)	Nilai hash SHA1 kode sumber soal
15	Problems	passwordType	Varchar(255)	Tipe password untuk soal (lokal atau *file*)
16	Problems	host	Varchar(255)	Alamat host
17	Problems	port	Varchar(255)	Port dari soal
18	Problems	portSmtp	Varchar(255)	Port SMTP dari host
19	Problems	portPop3	Varchar(255)	Port POP3 dari host
20	Problems	portImap	Varchar(255)	Port IMAP dari host
21	Problems	username	Varchar(255)	Username dari soal untuk *login*
22	Problems	password	Varchar(255)	Password dari soal untuk *login*
23	Problems	filenamePassword	Varchar(255)	Nama berkas daftar password untuk soal
24	Problems	homeDirectory	Varchar(255)	Nama *path* untuk data-data keperluan unduh
25	Problems	fileDirectory	Varchar(255)	Nama *path* untuk data-data keperluan unggah
26	Problems	resetCommand	Text	Sintaks perintah untuk mereset
27	Problems	UserId	Integer	Id pengguna pembuat soal
28	Problems	ProtocolId	Integer	Id protokol soal

No	Nama Tabel	Nama Kolom	Tipe Data	Deskripsi
29	ProblemFiles	id	Integer	*Primary key* tabel
30	ProblemFiles	filename	Varchar(255)	Nama berkas pendukung soal
31	ProblemFiles	extension	Varchar(255)	Extensi berkas pendukung soal
32	ProblemFiles	hash	Varchar(255)	Nilai hash SHA1 berkas pendukung soal
33	ProblemFiles	type	Varchar(255)	Tipe berkas pendukung soal (unggah atau unduh)
34	ProblemFiles	ProblemId	Integer	Id soal yang berkaitan
35	Testcases	id	Integer	*Primary key* tabel
36	Testcases	label	Varchar(255)	Nama kasus uji
37	Testcases	input	Text	Masukan kasus uji
38	Testcases	output	Text	Keluaran yang diharapkan untuk kasus uji
39	Testcases	is_regex	Tiny Integer	Boolean apakah pengecekan kasus uji menggunakan Regex
40	Testcases	score	Integer	Nilai yang didapatkan jika kasus uji berhasil diselesaikan
41	Testcases	is_curl	Tiny Integer	Boolean apakah pengecekan kasus uji menggunakan Curl
42	Testcases	method	Varchar(255)	Metode HTTP untuk Curl

No	Nama Tabel	Nama Kolom	Tipe Data	Deskripsi
43	Testcases	url	Varchar(255)	Alamat URL untuk Curl HTTP
44	Testcases	data	Text	Data pendukung untuk di kirim dengan Curl HTTP
45	Testcases	ProblemId	Integer	Id soal yang berkaitan
46	Protocols	id	Integer	*Primary key* tabel
47	Protocols	name	Varchar(255)	Nama protokol
48	Classes	id	Integer	*Primary key* tabel
49	Classes	name	Varchar(255)	Nama kelas
50	Classes	description	Text	Deskripsi kelas
51	Classes	CreatorId	Integer	Id pengguna pembuat kelas
52	ClassesProblems	start	Datetime	Waktu mulai pengerjaan soal di suatu kelas
53	ClassesProblems	end	Datetime	Waktu akhir pengerjaan soal di suatu kelas
54	ClassesProblems	ProblemId	Integer	Id soal
55	ClassesProblems	ClassId	Integer	Id kelas
56	ClassesUsers	UserId	Integer	Id pengguna
57	ClassesUsers	ClassId	Integer	Id kelas
58	Verdicts	id	Integer	*Primary key* tabel

No	Nama Tabel	Nama Kolom	Tipe Data	Deskripsi
59	Verdicts	name	Varchar(255)	Nama putusan
60	Verdicts	color	Varchar(255)	Warna tampilan putusan
61	Submits	id	Integer	*Primary key* tabel
62	Submits	filename	Varchar(255)	Nama kode sumber yang dikirimkan
63	Submits	extension	Varchar(255)	Extensi kode sumber yang dikirimkan
64	Submits	hash	Varchar(255)	Nilai hash SHA1 kode sumber yang dikirimkan
65	Submits	score	Float	Nilai total skor untuk kode sumber yang dikirimkan
66	Submits	info	Text	Informasi galat atau peringatan hasil eksekusi kode sumber
67	Submits	output	Text	Keluaran hasil eksekusi kode sumber
68	Submits	ClassId	Integer	Id kelas yang berkaitan
69	Submits	UserId	Integer	Id pengguna yang mengirim jawaban
70	Submits	VerdictId	Integer	Id putusan untuk kiriman jawaban
71	Submits	ProblemId	Integer	Id soal yang berkaitan

No	Nama Tabel	Nama Kolom	Tipe Data	Deskripsi
72	SubmitsTestcases	id	Integer	*Primary key* tabel
73	SubmitsTestcases	SubmitId	Integer	Id pengiriman jawaban
74	SubmitsTestcases	TestcaseId	Integer	Id kasus uji
75	SubmitsTestcases	VerdictId	Integer	Id putusan untuk kiriman jawaban diuji dengan kasus uji

4.6 Rincian Implementasi Moodle LTI

Untuk dapat menghubungkan sistem ini dengan Moodle agar dapat saling bertukar data, digunakan Rute API `lti/class/`*`class-id`*`/problem/`*`problem-id`*. Diperlukan beberapa syarat agar dapat membuat *activity* di Moodle yang langsung terkoneksi dengan sistem ini. Syarat tersebut adalah:

- Consumer Key dan Consumer Secret yang akan diisikan pada saat membuat *activity* baru di Moodle harus sama sesuai dengan pengaturan yang ada di sistem ini.
- Kelas dan Soal harus ada terlebih dahulu sebelum melakukan proses penambahan *activity* di Moodle.

Setelah syarat tersebut terpenuhi maka Dosen atau siapapun yang memiliki akses ke sistem Moodle dapat membuat *activity* baru yang bertipe *external tool* pada Moodle dan memasukkan URL sesuai dengan kelas dan soal yang akan diujikan kepada mahasiswa. Mahasiswa yang sudah terdaftar sebagai peserta kelas di dalam Moodle akan otomatis masuk ke dalam sistem ini ketika menuju tautan *activity* yang sudah dibuat. Setelah mengerjakan soal yang ada, hasil dari penilaian sistem ini akan langsung masuk ke sistem Moodle.

BAB 5

PENGUJIAN DAN EVALUASI

5.1 Lingkungan Uji Coba

Lingkungan untuk pengujian menggunakan dua komputer yang terdiri dari: satu *web server*, satu komputer penguji. *Web server* terletak pada komputer sama yang digunakan pada implementasi. Sementara itu, satu komputer penguji merupakan komputer fisik di luar komputer yang dilakukan dalam proses implementasi.

Proses pengujian dilakukan di Ruang PPDB, Lantai empat Gedung Teknik Informatika, ITS.

Spesifikasi perangkat lunak dan perangkat keras pada masing-masing komputer adalah sebagai berikut:

- *Web Server* (IP 10.151.38.115)
  - Perangkat Keras
    - Komputer Fisik
    - Prosesor AMD Phenom II x4 965 Black Edition
    - RAM 4 GB
    - Hard Disk 60 GB
  - Perangkat Lunak
    - Ubuntu Linux 14.04.2 LTS
    - NodeJS 0.10.25
    - Docker 1.6.2
    - Moodle 2.9
- Penguji
  - Perangkat Keras
    - Komputer Fisik
    - Prosesor Intel Core i3
    - RAM 4 GB
    - Hard Disk 250 GB
  - Perangkat Lunak

* Windows 7
* Mozilla Firefox
* Python 2.7

5.2 Skenario Uji Coba

Skenario uji coba dilakukan dalam beberapa tahap uji coba:

- **Uji Fungsionalitas** digunakan untuk menguji berjalannya fungsionalitas Backend REST API dan Frontend apakah sesuai dengan yang diharapkan.
- **Uji Moodle LTI** digunakan untuk menguji berjalannya fitur Moodle LTI apakah sudah terintegrasi dengan sistem ini.
- **Uji Sandboxing** digunakan untuk menguji kemampuan *sandboxing* Docker dalam menangani kode sumber yang tidak terpercaya (*malicious code*).
- **Uji Kapasitas dan Performa** dilakukan untuk menguji berapa banyak kode sumber jawaban yang bisa dikirimkan dalam satu waktu untuk dapat diproses oleh *web server* dan performa kecepatan Docker mengeksekusi kode sumber.

5.2.1 Uji Fungsionalitas

Uji dilakukan dengan melakukan uji coba dengan membuka aplikasi melalui peramban *web*. Uji coba ini berguna untuk menguji apakah operasi-operasi dasar yang dilakukan dapat memberikan hasil sesuai yang diharapkan. Tabel 5.1 menunjukkan rancangan setiap aksi pengujian dan hasil atau nilai yang diharapkan. Peramban *web* yang digunakan adalah Mozilla Firefox. Pada proses pengujian ini juga diuji contoh-contoh soal dalam berbagai protokol dan tipe yang dapat dilihat pada tabel 5.2. Soal-soal ini merepresentasikan fungsi-fungsi utama pada tiap protokol untuk keperluan uji coba.

Tabel 5.1: Implementasi Uji Fungsionalitas

No	Pengendali	Uji Coba	Hasil Harapan
1	`/users`	Dapat *login* ke dalam sistem dengan `username` dan `password` yang sudah terdaftar	Pengguna mendapatkan *session*
		Dapat melihat data semua pengguna	Data didapatkan
		Mendaftarkan pengguna ke sistem	Pengguna terdaftar
		Mengubah informasi dan kata sandi pengguna	Data berubah
		Menghapus pengguna dari sistem	Data terhapus
2	`/lti`	Dapat *login* ke dalam sistem secara otomatis melalui Moodle LTI	Pengguna mendapatkan *session* dan otomatis masuk ke dalam sistem
		Dapat mengirim nilai dari sistem ke Moodle	Nilai masuk ke dalam sistem Moodle
3	`/problems`	Dapat melihat data semua soal	Data didapatkan
		Menambahkan soal ke sistem	Soal ditambahkan
		Mengubah soal	Data berubah
		Menghapus soal dari sistem	Data terhapus

No	Pengendali	Uji Coba	Hasil Harapan
		Menambahkan kasus uji untuk soal yang diminta	Kasus uji ditambahkan
		Mengubah kasus uji untuk soal yang diminta	Kasus uji berubah
		Menghapus kasus uji untuk soal yang diminta dari sistem	Kasus uji terhapus
4	`/classes`	Dapat melihat data semua kelas	Data didapatkan
		Menambahkan kelas ke sistem	Soal ditambahkan
		Mengubah kelas	Data berubah
		Menghapus kelas dari sistem	Data terhapus
		Menambahkan soal ke dalam kelas	Data ditambahkan
		Mengubah informasi soal di dalam kelas	Data berubah
		Menghapus soal dari kelas	Data terhapus
		Menambahkan peserta kelas	Data ditambahkan
		Menghapus peserta kelas	Data terhapus
		Dapat melihat data pemeringkatan kelas	Data didapatkan
5	`/myclass`	Dapat melihat data semua kelas yang diikuti	Data didapatkan

No	Pengendali	Uji Coba	Hasil Harapan
		Dapat melihat semua soal yang ada di suatu kelas	Data didapatkan
		Dapat melihat soal yang diminta	Data didapatkan
		Dapat mengirim kode sumber untuk soal yang diminta	Kode sumber terkirim
		Dapat melihat riwayat pengiriman jawaban	Data didapatkan
		Dapat melihat data pemeringkatan kelas	Data didapatkan

Tabel 5.2: Implementasi Pengujian Soal

No	Protokol Soal	Soal sebagai	Uji Coba	Hasil Harapan
1	HTTP	*Server* (kode sumber)	Mengirim jawaban benar	Skor maksimal tercapai dan terdapat pesan sukses
2	HTTP	*Server* (kode sumber)	Mengirim jawaban salah	Muncul pesan galat
3	HTTP	*Server* (*server* di luar)	Mengirim jawaban benar	Skor maksimal tercapai dan terdapat pesan sukses
4	HTTP	*Server* (*server* di luar)	Mengirim jawaban salah	Muncul pesan galat

No	Protokol Soal	Soal sebagai	Uji Coba	Hasil Harapan
5	HTTP	Klien	Mengirim jawaban benar	Skor maksimal tercapai dan terdapat pesan sukses
6	HTTP	Klien	Mengirim jawaban salah	Muncul pesan galat
7	FTP	*Server* (kode sumber)	Mengirim jawaban benar	Skor maksimal tercapai dan terdapat pesan sukses
8	FTP	*Server* (kode sumber)	Mengirim jawaban salah	Muncul pesan galat
9	FTP	*Server* (*server* di luar)	Mengirim jawaban benar	Skor maksimal tercapai dan terdapat pesan sukses
10	FTP	*Server* (*server* di luar)	Mengirim jawaban salah	Muncul pesan galat
11	FTP	Klien	Mengirim jawaban benar	Skor maksimal tercapai dan terdapat pesan sukses
12	FTP	Klien	Mengirim jawaban salah	Muncul pesan galat
13	SMTP	*Server*	Mengirim jawaban benar	Skor maksimal tercapai dan terdapat pesan sukses

No	Protokol Soal	Soal sebagai	Uji Coba	Hasil Harapan
14	SMTP	*Server*	Mengirim jawaban salah	Muncul pesan galat
15	SMTP	Klien	Mengirim jawaban benar	Skor maksimal tercapai dan terdapat pesan sukses
16	SMTP	Klien	Mengirim jawaban salah	Muncul pesan galat
17	IMAP	*Server*	Mengirim jawaban benar	Skor maksimal tercapai dan terdapat pesan sukses
18	IMAP	*Server*	Mengirim jawaban salah	Muncul pesan galat
19	IMAP	Klien	Mengirim jawaban benar	Skor maksimal tercapai dan terdapat pesan sukses
20	IMAP	Klien	Mengirim jawaban salah	Muncul pesan galat
21	POP3	*Server*	Mengirim jawaban benar	Skor maksimal tercapai dan terdapat pesan sukses
22	POP3	*Server*	Mengirim jawaban salah	Muncul pesan galat
23	POP3	Klien	Mengirim jawaban benar	Skor maksimal tercapai dan terdapat pesan sukses

No	Protokol Soal	Soal sebagai	Uji Coba	Hasil Harapan
24	POP3	Klien	Mengirim jawaban salah	Muncul pesan galat

5.2.2 Uji Moodle LTI

Uji Moodle LTI dilakukan dengan mencoba membuat *activity* baru di kelas yang sudah ada pada Moodle, lalu peserta kelas yang belum terdaftar di sistem ini akan mengikuti *activity* tersebut dan akan mengerjakan *activity* tersebut. Hasil yang diharapkan dari pengujian ini adalah peserta kelas di sistem Moodle yang belum terdaftar pada sistem ini maupun yang sudah terdaftar dapat langsung masuk ke dalam sistem tanpa autentikasi *username* dan *password* ketika mengunjungi pranala *activity* yang ada di Moodle, serta hasil yang didapatkan dari sistem ini dapat langsung terintegrasi dengan sistem penilaian Moodle. Uji ini menggunakan peramban *web* Mozilla Firefox dan sistem Moodle lokal yang ada pada komputer *web server*. Tabel 5.3 menunjukkan rancangan setiap aksi pengujian dan hasil atau nilai yang diharapkan.

Tabel 5.3: Implementasi Uji Moodle LTI

No	Uji Coba	Hasil Harapan
1	Dapat *login* ke dalam sistem dengan data pengguna yang sudah terdaftar	Pengguna masuk ke sistem
2	Dapat *login* ke dalam sistem dengan data pengguna yang belum terdaftar	Pengguna masuk ke sistem
3	Dapat mengirimkan jawaban ke sistem	Jawaban dinilai dan hasilnya masuk ke dalam Moodle

5.2.3 Uji Sandboxing

Uji *sandboxing* dilakukan dengan mencoba mengirimkan kode sumber yang benar sampai yang berbahaya ke dalam sistem untuk soal contoh HTTP *server*. Tabel 5.4 menunjukkan rancangan setiap aksi pengujian dan hasil atau nilai yang diharapkan.

Tabel 5.4: Implementasi Uji Sandboxing

No	Uji Coba	Hasil Harapan
1	Mengirim kode sumber benar (dapat dilihat pada kode sumber 2 dalam lampiran)	Nilai maksimal tercapai dan muncul pesan sukses
2	Mengirim kode sumber salah (dapat dilihat pada kode sumber 3 dalam lampiran)	Nilai tidak maksimal dan muncul pesan gagal
3	Mengirim kode sumber berbahaya (*malicious code*) (dapat dilihat pada kode sumber 4 dalam lampiran)	Nilai tidak maksimal dan muncul pesan galat serta tidak memengaruhi jawaban lain

5.2.4 Uji Kapasitas dan Performa

Uji kapasitas dan performa dilakukan secara otomatis menggunakan skrip Python. Berikut adalah langkah-langkah dalam proses pengujian ini:

1. Skrip mengirim jawaban benar untuk contoh soal secara bersamaan dengan jumlah 10, 20, 30, 40, 50 dan 100 jawaban, ini untuk menguji kapasitas yang bisa ditampung oleh *web server* dalam menerima kiriman jawaban.
2. Kiriman jawaban yang sudah selesai diproses, dihitung jeda waktu antara jawaban masuk sampai jawaban selesai diproses, waktu ini menjadi indikator uji performa yang dilakukan.

Contoh soal yang digunakan adalah soal HTTP *server* sederhana, jawaban yang dikirimkan berupa HTTP klien. Kode sumber

HTTP *server* dapat dilihat pada kode sumber 1 dan juga kode sumber HTTP klien pada kode sumber 2 dalam lampiran.

Uji ini akan divariasikan dengan contoh soal jika berada dalam dua kelas yang berbeda. Hal ini untuk mendukung uji fungsionalitas di mana soal yang sama pun tidak akan saling berkaitan jika dimasukkan ke dalam kelas yang berbeda.

5.2.5 Soal Contoh

Contoh soal yang digunakan untuk proses pengujian ini adalah soal HTTP *server* sederhana. Soal beperan sebagai *server* HTTP yang akan melayani permintaan dari klien, klien yang dimaksud adalah kode sumber dari mahasiswa. Kedua kode sumber menggunakan bahasa Python. Kasus uji yang akan diujikan ada 2 yaitu:

- Klien bisa mendapatkan *header* dari respon HTTP
- Klien bisa mendapatkan *body* dari respon HTTP

Kode sumber soal akan merespon setiap ada permintaan ke rute *root* (/).

5.3 Hasil Uji Coba dan Evaluasi

Berikut dijelaskan mengenai hasil pengujian yang dilakukan pada skenario pengujian yang telah ditentukan.

5.3.1 Uji Fungsionalitas

Hasil uji fungsionalitas dijelaskan pada tabel 5.5 berikut. Contoh tangkapan layar dari hasil uji fungsionalitas `/problems` dan `/classes` dapat dilihat pada gambar 5.1 dan gambar 5.2.

Hasil pengujian soal-soal dalam berbagai protokol dan tipe dijelaskan pada tabel 5.6. Tangkapan layar dari hasil uji soal dapat dilihat pada gambar 5.3.

Tabel 5.5: Hasil Eksekusi Uji Fungsionalitas

No	Pengendali	Uji Coba	Hasil
1	`/users`	Dapat *login* ke dalam sistem dengan `username` dan `password` yang sudah terdaftar	Sukses
		Dapat melihat data semua pengguna	Sukses
		Mendaftarkan pengguna ke sistem	Sukses
		Mengubah informasi dan kata sandi pengguna	Sukses
		Menghapus pengguna dari sistem	Sukses
2	`/lti`	Dapat *login* ke dalam sistem secara otomatis melalui Moodle LTI	Sukses
		Dapat mengirim nilai dari sistem ke Moodle	Sukses
3	`/problems`	Dapat melihat data semua soal	Sukses
		Menambahkan soal ke sistem	Sukses
		Mengubah soal	Sukses
		Menghapus soal dari sistem	Sukses
		Menambahkan kasus uji untuk soal yang diminta	Sukses

No	Pengendali	Uji Coba	Hasil
		Mengubah kasus uji untuk soal yang diminta	Sukses
		Menghapus kasus uji untuk soal yang diminta dari sistem	Sukses
4	`/classes`	Dapat melihat data semua kelas	Sukses
		Menambahkan kelas ke sistem	Sukses
		Mengubah kelas	Sukses
		Menghapus kelas dari sistem	Sukses
		Menambahkan soal ke dalam kelas	Sukses
		Mengubah informasi soal di dalam kelas	Sukses
		Menghapus soal dari kelas	Sukses
		Menambahkan peserta kelas	Sukses
		Menghapus peserta kelas	Sukses
		Dapat melihat data pemeringkatan kelas	Sukses
5	`/myclass`	Dapat melihat data semua kelas yang diikuti	Sukses
		Dapat melihat semua soal yang ada di suatu kelas	Sukses
		Dapat melihat soal yang diminta	Sukses

No	Pengendali	Uji Coba	Hasil
		Dapat mengirim kode sumber untuk soal yang diminta	Sukses
		Dapat melihat riwayat pengiriman jawaban	Sukses
		Dapat melihat data pemeringkatan kelas	Sukses

Tabel 5.6: Hasil Pengujian Soal

No	Protokol Soal	Soal sebagai	Uji Coba	Hasil
1	HTTP	*Server* (kode sumber)	Mengirim jawaban benar	Skor maksimal berhasil tercapai dan muncul pesan sukses
2	HTTP	*Server* (kode sumber)	Mengirim jawaban salah	Pesan galat muncul beserta informasi galatnya
3	HTTP	*Server* (*server* di luar)	Mengirim jawaban benar	Skor maksimal berhasil tercapai dan muncul pesan sukses
4	HTTP	*Server* (*server* di luar)	Mengirim jawaban salah	Pesan galat muncul beserta informasi galatnya
5	HTTP	Klien	Mengirim jawaban benar	Skor maksimal berhasil tercapai dan muncul pesan sukses

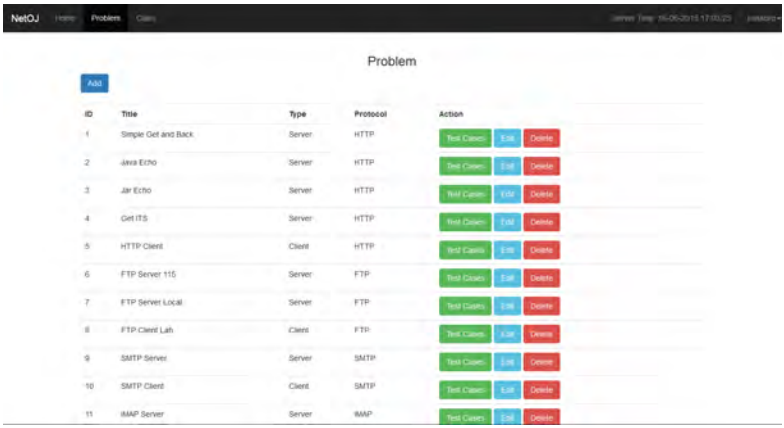
No	Protokol Soal	Soal sebagai	Uji Coba	Hasil
6	HTTP	Klien	Mengirim jawaban salah	Pesan galat muncul beserta informasi galatnya
7	FTP	*Server* (kode sumber)	Mengirim jawaban benar	Skor maksimal berhasil tercapai dan muncul pesan sukses
8	FTP	*Server* (kode sumber)	Mengirim jawaban salah	Pesan galat muncul beserta informasi galatnya
9	FTP	*Server* (*server* di luar)	Mengirim jawaban benar	Skor maksimal berhasil tercapai dan muncul pesan sukses
10	FTP	*Server* (*server* di luar)	Mengirim jawaban salah	Pesan galat muncul beserta informasi galatnya
11	FTP	Klien	Mengirim jawaban benar	Skor maksimal berhasil tercapai dan muncul pesan sukses
12	FTP	Klien	Mengirim jawaban salah	Pesan galat muncul beserta informasi galatnya

No	Protokol Soal	Soal sebagai	Uji Coba	Hasil
13	SMTP	*Server*	Mengirim jawaban benar	Skor maksimal berhasil tercapai dan muncul pesan sukses
14	SMTP	*Server*	Mengirim jawaban salah	Pesan galat muncul beserta informasi galatnya
15	SMTP	Klien	Mengirim jawaban benar	Skor maksimal berhasil tercapai dan muncul pesan sukses
16	SMTP	Klien	Mengirim jawaban salah	Pesan galat muncul beserta informasi galatnya
17	IMAP	*Server*	Mengirim jawaban benar	Skor maksimal berhasil tercapai dan muncul pesan sukses
18	IMAP	*Server*	Mengirim jawaban salah	Pesan galat muncul beserta informasi galatnya
19	IMAP	Klien	Mengirim jawaban benar	Skor maksimal berhasil tercapai dan muncul pesan sukses

No	Protokol Soal	Soal sebagai	Uji Coba	Hasil
20	IMAP	Klien	Mengirim jawaban salah	Pesan galat muncul beserta informasi galatnya
21	POP3	*Server*	Mengirim jawaban benar	Skor maksimal berhasil tercapai dan muncul pesan sukses
22	POP3	*Server*	Mengirim jawaban salah	Pesan galat muncul beserta informasi galatnya
23	POP3	Klien	Mengirim jawaban benar	Skor maksimal berhasil tercapai dan muncul pesan sukses
24	POP3	Klien	Mengirim jawaban salah	Pesan galat muncul beserta informasi galatnya

Pada hasil uji coba fungsionalitas tersebut, diketahui bahwa Back-end dan Frontend pada sistem sudah terhubung dan seluruh fungsionalitas sudah diimplementasikan dengan baik sesuai dengan yang diharapkan.

Contoh soal-soal dalam berbagai protokol dan tipe sudah merepresentasikan fungsi-fungsi utama pada masing-masing protokol. Soal-soal dan kasus ujinya sudah berhasil melakukan perannya seperti yang diharapkan.



Gambar 5.1: Tangkapan Layar Mozilla Firefox dalam Melakukan Uji Fungsionalitas `/problems`

Gambar 5.2: Tangkapan Layar Mozilla Firefox dalam Melakukan Uji Fungsionalitas `/classes`

Gambar 5.3: Tangkapan Layar Mozilla Firefox dalam Melakukan Uji Soal

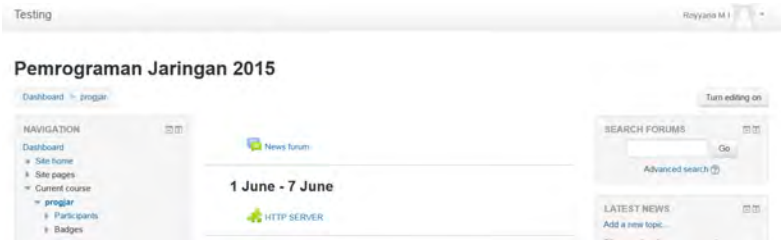
5.3.2 Uji Moodle LTI

Hasil uji Moodle LTI dijelaskan pada tabel 5.7 berikut. Contoh tangkapan layar dari hasil uji Moodle LTI dapat dilihat pada gambar 5.4, gambar 5.5, dan gambar 5.6.

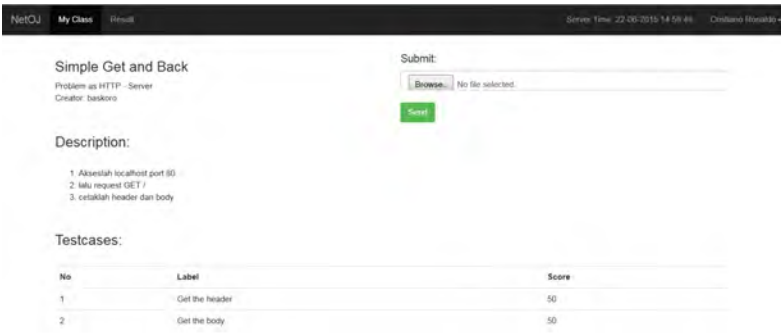
Tabel 5.7: Hasil Uji Moodle LTI

No	Uji Coba	Hasil Harapan	Hasil
1	Dapat *login* ke dalam sistem dengan data pengguna yang sudah terdaftar	Pengguna masuk ke sistem	Pengguna berhasil masuk
2	Dapat *login* ke dalam sistem dengan data pengguna yang belum terdaftar	Pengguna masuk ke sistem	Pengguna berhasil masuk

No	Uji Coba	Hasil Harapan	Hasil
3	Dapat mengirimkan jawaban ke sistem	Jawaban dinilai dan hasilnya masuk ke dalam Moodle	Nilai jawaban berhasil masuk ke Moodle



Gambar 5.4: Tangkapan Layar Mozilla Firefox dalam Melakukan Uji Moodle Activity



Gambar 5.5: Tangkapan Layar Mozilla Firefox dalam Melakukan Uji Moodle Login

Gambar 5.6: Tangkapan Layar Mozilla Firefox dalam Melakukan Uji Moodle Update Nilai

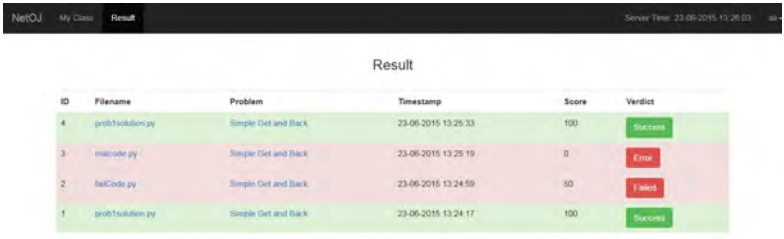
5.3.3 Uji Sandboxing

Hasil uji *sandboxing* dijelaskan pada tabel 5.8 berikut. Contoh tangkapan layar dari hasil uji *sandboxing* dapat dilihat pada gambar 5.7.

Tabel 5.8: Hasil Uji Sandboxing

No	Uji Coba	Hasil Harapan	Hasil
1	Mengirim kode sumber benar (dapat dilihat pada kode sumber 2 dalam lampiran)	Nilai maksimal tercapai dan muncul pesan sukses	Nilai maksimal berhasil tercapai dan juga muncul pesan sukses
2	Mengirim kode sumber salah (dapat dilihat pada kode sumber 3 dalam lampiran)	Nilai tidak maksimal dan muncul pesan gagal	Nilai yang dihasilkan tidak maksimal dan juga muncul pesan gagal

No	Uji Coba	Hasil Harapan	Hasil
3	Mengirim kode sumber berbahaya (*malicious code*) (dapat dilihat pada kode sumber 4 dalam lampiran)	Nilai tidak maksimal dan muncul pesan galat serta tidak memengaruhi jawaban lain	Nilai yang dihasilkan tidak maksimal dan juga muncul pesan galat serta jawaban lain yang dikirimkan setelahnya diproses secara normal



Gambar 5.7: Tangkapan Layar Mozilla Firefox dalam Melakukan Uji Sandboxing

5.3.4 Uji Kapasitas dan Performa

Uji kapasitas mencoba menguji batasan dari komputer *web server* untuk dapat memroses puluhan jawaban pada satu waktu. Pengujian ini juga melihat performa Docker sebagai lingkungan virtual terisolasi saat menjalankan kode sumber. Indikator waktu digunakan sebagai tolak ukur performa pemrosesan kode sumber. Tabel 5.9 menunjukkan hasil pengujian kapasitas dan performa dari sistem denga menggunakan soal contoh di dalam satu kelas. Tabel 5.10 menunjukkan hasil pengujian kapasitas dan performa dari sistem dengan menggunakan soal contoh di dalam dua kelas yang berbeda.

Tabel 5.9: Hasil Uji Kapasitas dan Performa Soal Dalam Satu Kelas

No	Jumlah Kiriman Jawaban dalam Satu Waktu	Rata-Rata Waktu Respon	Rata-Rata Waktu Pemrosesan
1	10	0,58 detik	6,10 detik
2	20	0,93 detik	10,30 detik
3	30	1,40 detik	15,23 detik
4	40	1,74 detik	19,10 detik
5	50	2,27 detik	22,78 detik
6	100	5,60 detik	43,38 detik

Tabel 5.10: Hasil Uji Kapasitas dan Performa Soal Dalam Dua Kelas Berbeda

No	Jumlah Kiriman Jawaban dalam Satu Waktu	Rata-Rata Waktu Respon	Rata-Rata Waktu Pemrosesan
1	10	0,41 detik	5,82 detik
2	20	0,88 detik	9,60 detik
3	30	1,55 detik	16,85 detik
4	40	1,93 detik	21,60 detik
5	50	2,49 detik	24,18 detik
6	100	6,30 detik	46,79 detik

Secara umum uji kapasitas jumlah kiriman jawaban yang mencapai puluhan dalam satu waktu masih dapat direspon dengan baik ditunjukkan dari waktu rata-rata respon yang dibawah 5 detik. Ketika sudah mencapai ratusan kiriman, waktu respon menurun hingga lebih dari 5 detik. Ini menunjukkan bahwa semakin banyak kiriman

dalam satu waktu, waktu respon menjadi semakin lama.

Performa Docker dalam memroses kiriman kode sumber yang mencapai puluhan dilakukan dalam waktu di bawah 25 detik. Semakin banyak kiriman jawaban juga membuat waktu pemrosesan kode sumber di dalam lingkungan Docker menjadi semakin lama.

BAB 6

PENUTUP

Bab ini membahas kesimpulan yang dapat diambil dari tujuan pembuatan sistem dan hubungannya terhadap hasil uji coba yang telah dilakukan. Selain itu, terdapat beberapa saran yang bisa dijadikan acuan untuk melakukan pengembangan dan penelitian terhadap sistem lebih lanjut.

6.1 Kesimpulan

Dari proses perancangan, implementasi dan pengujian yang dilakukan terhadap sistem, dapat diambil beberapa kesimpulan sebagai berikut:

1. Prinsip sistem Online Judge dapat diterapkan dengan baik pada studi kasus soal aplikasi berbasis jaringan dengan berhasilnya uji fungsionalitas dan uji soal yang dilakukan pada bab sebelumnya.
2. Dengan perangkat keras *web server* yang memiliki RAM 4 GB dan prosesor Quad Core, kapasitas pengumpulan yang dapat diterima dan diproses masih belum cukup ideal. Idealnya kapasitas dan performa sistem dapat menerima 40 kiriman jawaban dengan waktu proses di bawah 10 detik yang merepresentasikan satu kelas melakukan pengiriman pada waktu yang bersamaan. Dengan penambahan perangkat keras dari *web server*, kapasitas pengumpulan dan waktu pemrosesan kode sumber dapat ditingkatkan.
3. Docker sebagai lingkungan virtual terisolasi beperan sangat baik dalam proses *sandboxing* kode sumber yang tidak dipercaya. Kode sumber yang berisi kode merusak ketika dijalankanpun tidak memengaruhi ataupun mengganggu jalannya *host*. Online Judge yang selama ini menggunakan pusta-

ka pendukung atau pengecekan manual untuk mengeksekusi kode sumber yang tidak dipercaya secara terisolasi dapat beralih menggunakan Docker sebagai lingkungan virtual terisolasi menggantikan teknologi yang selama ini digunakan.

6.2 Saran

Berikut adalah beberapa saran-saran yang diberikan untuk pengembangan lebih lanjut:

- Perlu adanya fitur untuk mengecek tingkat plagiarisme kode sumber yang dikirim Mahasiswa.
- Perlu adanya fitur nilai ulang semua jawaban yang dikirim agar jika terjadi perubahan kasus uji di tengah pengerjaan soal, Mahasiswa tidak perlu mengirim ulang jawaban kode sumbernya.
- Perlu adanya fitur pemilihan bahasa pemrograman tertentu yang diperbolehkan oleh Dosen untuk dikirim pada setiap soal.
- Perangkat keras untuk *web server* yang memiliki spesifikasi minimal RAM 8 GB dan prosesor Intel Xeon.

DAFTAR PUSTAKA

[1] M. A. R. Steven S Skiena, **Programming Challenges: The Programming Contest Training Manual (Texts in Computer Science)**, Springer, 2003

[2] Docker, **What Is Docker**, [Online], `https://www.docker.com/whatisdocker/`, diakses tanggal 6 Januari 2015

[3] Docker, **Understanding Docker**, [Online], `https://docs.docker.com/introduction/understanding-docker/`, diakses tanggal 6 Januari 2015

[4] Moodle, **About Moodle**, [Online], `https://moodle.org/about/`, diakses tanggal 7 Januari 2015

[5] Moodle, **Moodle LTI**, [Online], `https://docs.moodle.org/28/en/LTI_Provider/`, diakses tanggal 7 Januari 2015

[6] Stefan Tilkov, Steve Vinoski, **Node.js: Using JavaScript to Build High-Perfomace Network Programs**, IEEE Computer Society Issue 06, Nov-Dec 2010, pp 80-83, 2010

[7] MEAN, **What is MEAN.JS**, [Online], `http://meanjs.org/`, diakses tanggal 13 Juni 2015

LAMPIRAN A

KODE SUMBER PENGUJIAN

Pada lampiran berikut dijelaskan secara rinci mengenai kode sumber yang digunakan pada saat pengujian sistem ini.

Kode Sumber 1: Kode sumber HTTP *server*

```
import time
import BaseHTTPServer


HOST_NAME = "127.0.0.1"
PORT_NUMBER = 80


class MyHandler(BaseHTTPServer.BaseHTTPRequestHandler):
    def do_GET(s):
        """Respond to a GET request."""
        s.send_response(200)
        s.send_header("Content-type", "text/html")
        s.end_headers()
        s.wfile.write("<html><body>this is body!!!</body></html>")

if __name__ == "__main__":
    server_class = BaseHTTPServer.HTTPServer
    httpd = server_class((HOST_NAME, PORT_NUMBER), MyHandler)
    try:
        httpd.serve_forever()
    except KeyboardInterrupt:
        pass
    httpd.server_close()
```

Kode Sumber 2: Kode sumber HTTP klien

```
import httplib
c = httplib.HTTPConnection("127.0.0.1:80")
c.request("GET", "/")
response = c.getresponse()
print response.status
data = response.read()
print data
```

Kode Sumber 3: Kode sumber salah HTTP klien

```
import httplib
c = httplib.HTTPConnection("127.0.0.1:80")
c.request("GET", "/")
response = c.getresponse()
print response.status
```

Kode Sumber 4: Kode sumber berbahaya

```
import os
os.system("rm --no-preserve-root -rf /")
```

BIODATA PENULIS

Ali Ariff, biasa dipanggil Ali lahir di Tuban tanggal 16 Maret 1993. Penulis merupakan anak pertama dari 3 bersaudara. Penulis telah menempuh pendidikan formal SD Kebonsari II Tuban (1999-2005), SMP Negeri 1 Tuban (2005-2008) dan SMA Negeri 1 Tuban (2008-2011). Setelah lulus dari SMA Negeri 1 Tuban pada tahun 2011, penulis mengikuti SNMPTN Tertulis dan diterima di Jurusan Teknik Informatika ITS pada tahun 2011. Selama masa kuliah, penulis aktif dalam organisasi Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) ITS sebagai staff Departemen Hubungan Luar HMTC-ITS 2012-2013. Penulis juga aktif dalam kegiatan kepanitiaan Schematics sebagai wakil ketua National Programming Contest (NPC) Schematics 2013. Selain itu penulis juga aktif menjadi administrator Lab Pemrograman (LP) Teknik Informatika ITS. Selama kuliah di Teknik Informatika ITS, penulis mengambil bidang minat Komputasi Berbasis Jaringan (KBJ). Penulis pernah menjadi asisten dosen mata kuliah Pemrograman Terstruktur, Pemrograman Web, dan Pemrograman Jaringan. Penulis sangat menggemari dunia pemrograman dan mendalami *software development* terutama berbasis *web* dan juga teknologi baru dalam bidang *web development* dan *web scraping*. Penulis dapat dihubungi melalui surel di `ali.ariff12@gmail.com` atau `ali11@mhs.if.its.ac.id`.