TUGAS AKHIR – KS 1501

PENGEMBANGAN SISTEM PLUGIN BERBASIS DEPENDENCY INJECTION PADA LEARNING MANAGEMENT SYSTEM BERBASIS WEB UNTUK SISTEM PENDETEKSI KEMIRIPAN PADA SOURCE CODE

Teguh Sasmito
5211 100 015

Dosen Pembimbing
Radityo Prasetianto Wibowo, S.Kom, M.Kom

JURUSAN SISTEM INFORMASI
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2015

FINAL PROJECT – KS 1501

PLUGIN SYSTEM DEVELOPMENT BASED ON DEPENDENCY INJECTION IN WEB-BASED LEARNING MANAGEMENT SYSTEM FOR SIMILARITY DETECTION SYSTEM ON SOURCE CODE

Teguh Sasmito
5211 100 015

Academic Promotors
Radityo Prasetianto Wibowo, S.Kom, M.Kom

INFORMATION SYSTEMS DEPARTMENT
Information Technology Faculty
Sepuluh NopemberInstitut of Technology
Surabaya 2015

LEMBAR PENGESAHAN

PENGEMBANGAN SISTEM PLUGIN BERBASIS DEPENDENCY INJECTION PADA LEARNING MANAGEMENT SYSTEM BERBASIS WEB UNTUK SISTEM PENDETEKSI KEMIRIPAN PADA SOURCE CODE

TUGAS AKHIR

Disusun untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Jurusan Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:

Teguh Sasmito
5211 100 015

Surabaya, Juli 2015
KETUA
JURUSAN SISTEM INFORMASI

Dr. Eng. Febriliyan Samopa, S.Kom, M.Kom
NIP 197302191998021001

LEMBAR PERSETUJUAN

PENGEMBANGAN SISTEM PLUGIN BERBASIS DEPENDENCY INJECTION PADA LEARNING MANAGEMENT SYSTEM BERBASIS WEB UNTUK SISTEM PENDETEKSI KEMIRIPAN PADA SOURCE CODE

TUGAS AKHIR

Disusun untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Jurusan Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
Teguh Sasmito
5211 100 015

Disetujui Tim Penguji: Tanggal Ujian: 14 Juli 2015
Periode Wisuda: September 2015

Radityo Prasetianto W., S.Kom, M.Kom **(Pembimbing)**

Faizal Johan Atletiko, S.Kom, M.T **(Penguji 1)**

Arif Wibisono, S.Kom, M.Sc **(Penguji 2)**

PENGEMBANGAN SISTEM PLUGIN BERBASIS DEPENDENCY INJECTION PADA LEARNING MANAGEMENT SYSTEM BERBASIS WEB UNTUK SISTEM PENDETEKSI KEMIRIPAN PADA SOURCE CODE

Nama Mahasiswa	**:**	**Teguh Sasmito**
NRP	**:**	**5211100015**
Jurusan	**:**	**Sistem Informasi FTIf – ITS**
Dosen Pembimbing	**:**	**Radityo Prasetianto Wibowo, S.Kom, M.Kom**

ABSTRAK

Plagiarisme merupakan tindakan meniru hasil karya orang lain tanpa mencantumkan identitas penulis aslinya. Tindak plagiarisme sering terjadi di kalangan mahasiswa, tak terkecuali saat membuat source code pada sebuah program. Contoh plagiarisme yang sering terjadi saat membuat source code yaitu menyalin atau memodifikasi source code dari hasil karya mahasiswa lain. Namun pada kenyataannya, sering terjadi kekeliruan dalam pendeteksian plagiarisme. Kekeliruan tersebut adalah saat suatu source code yang dibuat oleh mahasiswa dianggap merupakan tindak plagiarisme dikarenakan source code tersebut sama persis dengan mahasiswa yang lain, namun pada kenyataannya mahasiswa tersebut membuat source code dari awal tanpa menjiplak dari sumber lain. Begitu juga sebaliknya.

Pada penelitian ini akan dijelaskan bagaimana mendeteksi persentase kemiripan pada source code yang dibuat mahasiswa pada Learning Management System menggunakan Multiple Provider. Multiple Provider yang dimaksud adalah penggunaan algoritma Levenshtein Distance dan Rabin Karp. Algoritma Levenshtien Distance termasuk dalam kategori

metode Substring Matching dimana pendeteksian dilakukan dengan cara mencocokkan dua string ke dalam matriks dua dimensi. Sedangkan algoritma Rabin Karp adalah algoritma pencarian kata yang mencari sebuah pola berupa substring dalam sebuah teks menggunakan hashing.

Hasil akhir dari penelitian ini adalah algoritma Levenshtein Distance mampu mendeteksi kemiripan pada source lebih tepat dari Rabin Karp. Sementara algoritma Rabin Karp berhasil menghasilkan persentase kemiripan lebih besar dari algoritma Levenshtein Distance.

Kata Kunci: Learning Management System, kemiripan, source code, Levenshtien Distance, Rabin Karp

PLUGIN SYSTEM DEVELOPMENT BASED ON DEPENDENCY INJECTION IN WEB-BASED LEARNING MANAGEMENT SYSTEM FOR SIMILARITY DETECTION SYSTEM ON SOURCE CODE

Nama Mahasiswa	**:**	**Teguh Sasmito**
NRP	**:**	**5211100015**
Jurusan	**:**	**Sistem Informasi FTIf – ITS**
Dosen Pembimbing	**:**	**Radityo Prasetianto W., S.Kom, M.Kom**

ABSTRACT

Plagiarism is an action that copy other people's work without give their identity as a credit. Plagiarism happens frequently between college students, even when they made a program source code. Plagiarism that happens when students make a source code is copying or modifying source code from the other students. But in fact, mistakes in the detection of plagiarism happens often. That mistakes come when a source code that made by students was judged as a plagiarism because the source code is exactly same with the other students, however that students make the source code from beginning without copy from the other sources.

This research explains how to detect the similarity percentage of student's source code at Learning Management System use multiple provider. Multiple provider is using of levenshtien distance and rabin karp algorithm. Levenshtien distance algorithm included in the substring matching methods category where detection process done with matching two strings into two dimension matrix.However rabin karp algorithm is a word matching algorithm that find a substring pattern in the text using hashing process.

The results of this final project are Levenshtein Distance Algorithm can detect similarity of source code better than Rabin Karp and Rabin Karp algorithm can managed to produce the percentage of similarity larger than Levenshtein Distance algorithm.

Kata Kunci: Learning Management System, similarity, source code, Levenshtien Distance, Rabin Karp

KATA PENGANTAR

Alhamdulillahirobbil ‘alamiin, tiada yang mampu menolong selain Allah SWT sehingga penulis dapat menyelesaikan buku tugas akhir dengan judul **“PENGEMBANGAN SISTEM PLUGIN BERBASIS DEPENDENCY INJECTION PADA LEARNING MANAGEMENT SYSTEM BERBASIS WEB UNTUK MENDETEKSI KEMIRIPAN PADA SOURCE CODE”**yang merupakan salah satu syarat kelulusan pada Jurusan Sistem Informasi, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

Dalam pengerjaan tugas akhir yang berlangsung selama bulan Januari Tahun 2015- Juli 2015 penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada yang senantiasa terlibat secara langsung memberikan bantuan dan dukungan dalam pengerjaan tugas akhir ini :

- Allah SWT yang telah memberikan kesehatan, kemudahan dan kesempatan untuk bisa menyelesaikan tugas akhir ini.
- Kepada Bapak Febriliyan Samopa selaku ketua jurusan sistem informasi.
- Kepada Bapak Radityo Prasetianto Wibowo S.Kom., M.Kom., selaku dosen pembimbing. Terima kasih atas segala bimbingan dan waktu yang telah diberikan untuk membantu penulis dalam menyelesaikan tugas akhir ini.

Penulis menyadari bahwa masih terdapat banyak kekurangan dalam penulisan tugas akhir ini, oleh karena itu penulis menerima saran dan kritik yang membangun Namun penulis berharap bahwa tugas akhir ini dapat bermanfaat bagi semua pihak yang membutuhkan.

Surabaya, Juli 2015

Penulis

Halaman ini sengaja Dikosongkan

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

Halaman ini sengaja Dikosongkan

BAB I
PENDAHULUAN

Pada bagian ini akan dijelaskan mengenai latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian yang mendasari penelitian tugas akhir, serta keterkaitan dengan roadmap Laboratorium E-Bisnis.

1.1. Latar Belakang

Karya merupakan suatu hasil atau pencapaian yang berasal dari ide yang dituangkan seseorang melalui kerja kerasnya. Namun kebanyakan karya tersebut sering disalahgunakan oleh orang lain yang menyebabkan timbulnya tindak plagiarisme. Plagiarisme sendiri adalah menggunakan karya atau ide orang lain tanpa mencantumkan nama ataupun identitas dari penulis aslinya[1]. Plagiarisme dapat juga dikatakan sebagai tindak pencurian. Jika mencuri mobil ataupun barang mewah lainnya akan dijatuhi hukuman, maka mencuri ide ataupun tulisan orang lain akan mendapat hukuman pula[2].

Praktek plagiarisme dapat dilakukan dimana saja. Salah satunya yaitu di kalangan mahasiswa, tak terkecuali untuk mahasiswa Jurusan Sistem Informasi ITS yang merupakan mahasiswa yang berada di lingkungan yang berkaitan dengan teknologi informasi. Praktik plagiarisme biasanya dilakukan dengan cara menyalin dan memodifikasi *source code* atau kode program yang dilakukan ketika mahasiswa sedang melakukan tugas berupa kuis ataupun ujian yang diberikan oleh dosen melalui asisten praktikum[3]. Modifikasi yang dimaksud adalah mengubah tipe data, mengubah urutan *statement*, dan mengganti komentar atapun identitas penulis[4]. Praktek plagiarisme sendiri sangat susah untuk dideteksi, namun dengan mendeteksi kemiripan pada *source code* setidaknya tindak plagiarisme dapat diminimalisir. Namun pada kenyataannya, para asisten praktikum sering mengalami kesulitan dalam penilaian tugas. Hal ini dikarenakan para asisten praktikum belum menemukan solusi

untuk menentukan persentase tingkat kemiripan dalam tugas mahasiswa.

Untuk menghindari kejadian tersebut, perlu dilakukan usaha untuk meminimalisir tindak plagiarisme. Menurut Sugiyanto (2010) ada dua cara dalam meminimalisir tindak plagiarisme yaitu mencegah yang berarti menghalangi agar tindak plagiarisme tidak terjadi, dan mendeteksi yang berarti melakukan pendeteksian kemiripan terhadap tugas-tugas mahasiswa yang telah dikumpulkan. Salah satu contoh mencegah plagiarisme terjadi adalah dengan cara pemberian sanksi yang jelas apabila ada mahasiswa yang dengan sengaja menjiplak tugas dari mahasiswa lain[3]. Sedangkan contoh mendeteksi kemiripan adalahmemeriksa secara manual tugas-tugas mahasiswa yang sudah dikumpulkan antara satu dengan yang lainnya. Namun hal tersebut memiliki kelemahan yaitu waktu yang diperlukan untuk pemeriksaan, karena dengan memeriksa tugas dari puluhan mahasiswa memerlukan waktu sangat lama, apalagi dengan hanya menafsirkan dari gaya penulisannya saja[5]. Oleh karena itu, untuk menghindari kelemahan tersebut perlu adanya suatu sistem yang otomatis dapat mendeteksi plagiarisme secara efektif.

Tujuan dari tugas akhir ini adalah membangun sebuah *plugin* pada *Learning Management System* yang sudah ada untuk mengetahui persentase tingkat kemiripan antara *source code* satu dengan *source code* yang lainnya. *Source code* yang digunakan merupakan *source code* yang dibuat mahasiswa dalam kuis yang ada pada *Learning Management System* yang dibuat oleh Sipahutar (2013). Dalam sistem ini digunakan algoritma *Levenshtien Distance* untuk mengecek tingkat kemiripan *string* antar kedua *source code*. Algoritma *Levenshtien Distance* akan digunakan setelah *source code* dinormalisasi ke dalam bentuk yang lebih sederhana. Sebelum proses normalisasi, proses *scanner* akan mengubah *source code* yang telah diinputkan oleh pengguna menjadi *token*. Selain menggunakan *Levenshtien Distance*, penulis

menggunakan salah satu algoritma yang juga popular dalam mendeteksi kemiripan pada *source code* yaitu algoritma *Rabin Karp*.

1.2. Rumusan Masalah

Berdasarkan penjelasan latar belakang di atas, rumusan masalah yang menjadi fokus utama dan perlu diperhatikan adalah:

Bagaimana membangun sistem *plugin* pada *Learning Management System*?

Bagaimana mengintegrasikan *Multi Provider* pada *Learning Management System*?

1.3. Batasan Masalah

Dari permasalahan yang telah disebutkan diatas, batasan masalah dalam tugas akhir ini adalah sebagai berikut:

1. *Learning Management System* yang dipakai adalah aplikasi *E-learning* untuk belajar dasar pemrograman *Java* yang dibuat oleh Irfan Satria Sipahutar.
2. Sistem *plugin* berfungsi untuk mendeteksi kemiripan dan membuat report dari hasil pendeteksian tersebut.
3. *Design Pattern* utama yang dipakai untuk sistem *plugin* adalah *Dependency Injection*.
4. Pengujian kemiripan *source code* menggunakan *Multi Provider* yang algoritma *Levenshtien Distance* dan algoritma *Rabin-Karp*.

1.4. Tujuan Tugas Akhir

Berdasarkan rumusan masalah yang telah dijelaskan sebelumnya, maka tujuan yang akan dicapai dalam penelitian ini adalah:

1. Mengetahui cara membangun sistem *plugin* untuk mendeteksi kemiripan*source code* pada *Learning Management System*.
2. Mengetahui cara mengintegrasikan *Multi Provider* pada *Learning Management System*.

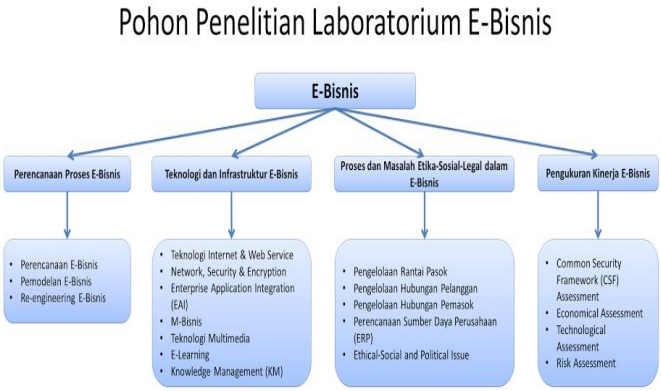
1.5. Manfaat Tugas Akhir

Tugas akhir ini diharapkan dapat memberikan manfaat bagi asisten praktikum dan dosen antara lain:

Dapat membantu asisten praktikum maupun dosen dalam menilai tugas mata kuliah dari mahasiswa.
Mendapatkan terobosan baru terkait cara mendeteksi tingkat kemiripan yang dilakukan mahasiswa.
Mampu mengoreksi setiap tugas dari mahasiswa secara efektif.

1.6. Relevansi

Pembuatan dari tugas akhir ini berkaitan dengan *Road Map* Laboratorium E-Bisnis. Hal ini berkaitan dengan perkembangan penelitian yang dilakukan laboratorium E-bisnis. Rencana penelitian laboratorium E-Bisnis digambarkan dalam pohon penelitian laboratorium E-Bisnis. Pohon penelitian ini memiliki empat bahasan utama, yang diantaranya adalah Perencanaan Proses Bisnis, Teknologi dan Infrastruktur E-Bisnis, Proses dan masalah etika sosial-legal E-bisnis, dan Pengukuran Kinerja E-bisnis. Pembahasan mengenai permasalahan dalam tugas akhir ini berada pada ruas pertama dalam Pohon penelitian laboratorium E-bisnis yaitu pada *Re-engineering* E-Bisnis. Diagram pohon penelitian laboratorium E-Bisnis yang sedang dikembangkan ditunjukkan oleh gambar 1.1 berikut ini.



Gambar 1.1 Pohon Penelitian E-Bisnis

1.7. Sistematika Penulisan

Dalam penulisan pada buku tugas akhir ini, terdapat beberapa sistematika penulisan yang diterapkan, yaitu terdiri dari beberapa bab yang berisi informasi yang berbeda-beda. Berikut penjelasan dari tiap-tiap bab yang terdapat pada buku tugas akhir ini:

- **BAB I PENDAHULUAN**
  Berisi uraian mengenai inisiasi penelitian yang akan dilakukan, yaitu latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, dan sistematika penulisan.

- **BAB II TINJAUAN PUSTAKA**
  Pada bab 2 berisi penjelasan mengenai pustaka dan definisi dari setiap hal yang berkaitan dengan penelitian yang dapat dijadikan sebagai sumber referensi dalam pembuatan tugas akhir ini. Teori yang dipaparkan dalam bab ini, antara lain mengenai bahasa pemrograman *Java*,*Plugin Design Pattern*, *Learning Management System*, Algoritma *Levenshtein Distance*, aplikasi yang memakai

algoritma *Rabin-Karp*, dan konsep-konsep lain yang berkaitan dengan pembuatan tugas akhir.

- **BAB III METODOLOGI**
  Berisi tentang tahapan-tahapan yang akan dilakukan dalam melakukan peneletian. Tahapan-tahapan ini diperlukan sebagai panduan secara terstruktur dan sistematis dalam pengerjaan tugas akhir

- **BAB IV ANALISIS DAN DESAIN SISTEM**
  Bab ini menjelaskan hal-hal terkait kebutuhan dan desain sistem pada aplikasi yang dibuat. Analisis kebutuhan sistem didasarkan pada aktivitas, user, fungsional, dan arsitektur. Desain sistem divisualisasikan dengan konsep Unified Modelling Language (UML).

- **BAB V IMPLEMENTASI DAN UJI COBA**
  Bab ini berisi hasil implementasi dari aplikasi yang telah dibangun beserta hasil pengujian aplikasi menggunakan User Acceptance Model.

- **BAB VI HASIL DAN PEMBAHASAN**
  Bab ini berisi tentang data yang didapatkan setelah dilakukan uji coba. Selain itu dijelaskan mengenai analisis yang dilakukan terhadap data yang telah didapatkan.

- **BAB VII KESIMPULAN DAN SARAN**
  Berisi tentang simpulan dari keseluruhan tugas akhir dan saran maupun rekomendasi terhadap penelitian tugas akhir selanjutnya yang memiliki kesamaan topik.

BAB II
TINJAUAN PUSTAKA

2.1. Studi Sebelumnya

Pada kasus-kasus sebelumnya, pengembangan aplikasi tentang pendeteksi kemiripan pada source sudah banyak dilakukan. Beberapa contoh studi kasus yang pernah dilakukan dalam mendeteksi kemiripan pada source code dijelaskan pada Tabel 2.1.

Tabel 2.1 Studi sebelumnya

Judul Penelitian	Tujuan Penelitian	Peneliti
Plagiarism Detection in Java Code	Mendeteksi plagiarisme *source code* menggunakan algoritma *Levenshtien Distance*	[2]
Plagiarism Detection System Design for Programming Assignment in Virtual Classroom Based on Moodle	Menambahkan fungsi pada *E-learning* untuk mendeteksi plagiarisme *source code* menggunakan algoritma *Rabin-Karp*	[6]

2.2. Bahasa Pemrograman *Java*

Java merupakan bahasa pemrograman yang bersifat umum/non-spesifik (*general purpose*), dan secara khusus didesain untuk memanfaatkan dependensi implementasi seminimal mungkin[7]. Bahasa pemrograman yang dirilis pada tahun 1995 ini dibuat oleh James Gosling saat masih bergabung di Sun Microsystems (yang saat ini merupakan bagian dari Oracle). Sintaksis yang ada di Java banyak mengadopsi dari bahasa pemrograman *C++* dan lebih diringkas ke dalam sintaksis yang lebih sederhana. Menurut Schildt (2007) program *Java* adalah kumpulan *whitespace*,

identifier, *literal*, *comment*, *separator*, *operator*, dan *Java Keyword*.

2.2.1. Whitespace

Java termasuk dalam kategori *free-form language* yang berarti kita tidak perlu terpaku pada aturan khusus. Kita dapat membuat suatu program dengan hanya mengetiknya pada satu baris saja atau dengan tata cara yang kita inginkan. Namun hal tersebut harus bisa terjadi asalkan ada satu karakter *whitespace* antara setiap *token* yang belum digambarkan oleh *operator* atau *separator*. Dalam bahasa pemrograman *Java*, *whitespace* terdiri dari spasi, tab, ataupun enter.

2.2.2. Identifier

Identifier digunakan untuk nama kelas, nama metode, dan nama variabel. *Identifier* dapat mendeskripsikan setiap urutan karakter huruf besar, huruf kecil, garis bawah, maupun karakter tanda dolar ($). Karena Java merupakan bahasa pemrograman yang bersifat sensitif, penulisan "KELAS" berbeda *identifier* dengan "kelas".

2.2.3. Literal

Sebuah nilai konstan yang di bahasa pemrograman *Java* dibuat dengan menggunakan representasi secara *literal*. Sebuah *literal* dapat digunakan pada nilai mana saja yang sesuai dengan jenis yang ada. Sebagai contoh 100 adalah sebuah *literal* yang menentukan bahwa karakter tersebut berupa *integer*.

2.2.4. Comment

Seperti bahasa pemrograman yang lainnya, *Java* dapat digunakan untuk memasukkan komentar ke dalam *source code* kita. Di dalam *Java* ada tiga tipe *comment* yang disediakan. Yang pertama yaitu *multiline comment*. Tipe dari *comment* ini dimulai dengan karakter /* dan diakhiri dengan karakter */. Yang kedua adalah *single-line comment*. Tipe dari *comment* ini dimulai dengan karakter // dan berakhir pada akhir baris.

Dan yang terakhir adalah *documentation comment*. Tipe ini digunakan untuk menghasilkan file HTML yang mendokumentasikan program kita. *Documentation comment* diawali dengan /** dan diakhiri dengan */.

2.2.5. Separator

Di dalam *Java*, karakter yang digunakan sebagai pemisah (*separator*) sangatlah sedikit. Karakter yang biasa digunakan di dalam *Java* sebagai separator adalah titik koma (;). Karakter ini digunakan untuk mengakhiri sebuah *statement*. Tabel 2.2 merupakan penjelasan setiap karakter yang termasuk ke dalam *separator* beserta fungsinya [8].

Tabel 2.2 Daftar Karakter Separator

Karakter	Nama	Fungsi
( )	Kurung	Digunakan untuk mengisikan parameter dalam pendefinisian dan pemanggilan metode. Selain itu juga digunakan untuk mendefinisikan prioritas dalam ekspresi.
{ }	Kurung kurawal	Digunakan untuk memuat nilai-nilai *array* yang diinisialisasi secara otomatis. Juga digunakan untuk menentukan blok kode, untuk kelas, metode, dan lingkup lokal.
[ ]	Kurung siku	Digunakan untuk menyatakan jenis *array*.
;	Titik koma	Digunakan untuk mengakhiri *statement*.
,	Koma	Memisahkan pengidentifikasi berurutan dalam deklarasi variabel. Juga digunakan untuk *chain statement* bersama-sama di dalam for *statement*.
.	Titik	Digunakan untuk memisahkan nama *package* dari *subpackages* dan *class*. Juga digunakan untuk memisahkan variabel atau metode dari variabel referensi.

2.2.6. Operator

Java menyediakan berbagai macam *operator*. Di dalam setiap *operator* terdapat *operand*, istilah *operand* digunakan pada sebuah objek yang ada pada operasi matematika yang dapat digunakan untuk melakukan operasi. Sebagian besar dari *operator* di bagi dalam beberapa grup. Grup tersebut adalah *Arithmetic Operators,Bitwise Operators, Relational Operators*, dan *Boolean Logical Operators*. Berikut penjelasan dari grup *operator* tersebut:

a. ***Arithmetic Operators*** digunakan dalam ekspresi matematika dengan cara yang sama saat *operator* tersebut digunakan dalam aljabar. *Operand* dari *operator* aritmatika harus dari tipe numerik. Anda tidak dapat menggunakannya pada jenis *boolean*, tetapi Anda dapat menggunakannya pada jenis *char*, karena tipe *char* di *Java*, pada dasarnya, bagian dari *integer*. Macam-macam *operator* yang termasuk *Arithmetic Operators* dijelaskan pada Tabel 2.3.

Tabel 2.3 Daftar Arithmetic Operators

Operator	Result
+	Addition
-	Substraction (also unary minus)
*	Multiplication
/	Division
%	Modulus
++	Increment
+=	Addition Assignment
-=	Substraction Assignment
*=	Multiplication Assignment
/=	Division Assignment
%=	Modulus Assignment
--	Decrement

b. ***Bitwise Operators***. *Java* mendefinisikan beberapa operator *bitwise* yang dapat diterapkan pada jenis *integer, long, int, short, char*, dan *byte*. *Operator* ini bertindak atas *bit* individual dari *operand* mereka.

Karena operator *bitwise* memanipulasi *bit* dalam *integer*, penting untuk memahami apa efek manipulasi tersebut mungkin memiliki sebuah nilai. Macam-macam *operator* yang termasuk *Bitwise Operators* dijelaskan pada Tabel 2.4.

Tabel 2.4 Daftar Bitwise Operators

Operator	Result
~	Bitwise unary NOT
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
>>	Shift right
>>>	Shift right zero fill
<<	Shift left
&=	Bitwise AND assignment
|=	Bitwise OR assignment
^=	Bitwise exclusive OR assignment
>>=	Shift right assignment
>>>=	Shift right zero fill assignment
<<=	Shift left

c. ***Relational Operators*** menentukan bahwa satu *operand* memiliki hubungan dengan satu atau lebih *operand* yang lainnya. Secara khusus, jenis *operator* ini menentukan kesamaan dan penataan *operand*. Hasil dari *relational operators* adalah berupa nilai *boolean*. Macam-macam *operator* yang termasuk *Relational Operators* dijelaskan pada Tabel 2.5.

Tabel 2.5 Daftar Relational Operators

Operator	Result
==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

d. ***Boolean Logical Operators*** yang ditampilkan di sini hanya beroperasi pada *operand boolean*. Semua *operator* logika biner menggabungkan dua nilai *boolean* untuk membentuk nilai *boolean* yang dihasilkan. Macam-macam *operator* yang termasuk *Boolean Logical Operators* dijelaskan pada Tabel 2.6.

Tabel 2.6 Daftar Boolean Logical Operators

Operator	Result
&	Logical AND
|	Logical OR
^	Logical XOR (exclusive OR)
||	Short-circuit OR
&&	Short-circuit AND
!	Logical unary NOT
&=	AND assignment
|=	OR assignment
^=	XOR assignment
==	Equal to
!=	Not equal to
?:	Ternary if-then-else

2.2.7. Java Keyword

Java Keyword dikombinasikan dengan sintaks operator dan pemisah, membentuk dasar dari bahasa pemrograman *Java*. Kata kunci tersebut tidak dapat digunakan sebagai nama untuk variabel, kelas, atau metode. Saat ini ada 50 *keyword* yang terdefiniskan dalam *Java*. *Keyword* tersebut dapat dilihat pada Tabel 2.7.

Tabel 2.7 Daftar Java Keywords

Abstract	Continue	For	New	Switch
Assert	Default	Goto	Package	Synchronized
Boolean	Do	If	Private	This
Break	Double	Implements	Protected	Throw
Byte	Else	Import	Public	Throws
Case	Enum	Instanceof	Return	Transient

Catch	Extends	Int	Short	Try
Char	Final	Interface	Static	Void
Class	Finally	Long	Strictfp	Volatile
Const	Float	Native	Super	While

2.3. Learning Management System

Learning Management System (LMS) adalah aplikasi perangkat lunak yang digunakan untuk kegiatan *online* program pembelajaran elektronik dan isi pelatihan. Menurut Ellis (2009) sebuah *LMS* yang kuat harus bisa melakukan hal-hal berikut:

- Menggunakan layanan "*self-guided*" dan "*self-service*".
- Mengumpulkan dan menyampaikan konten pembelajaran dengan cepat.
- Mengkonsolidasikan inisiatif pelatihan pada *platform* berbasis *web scalable*.
- Mendukung portabilitas dan standard.
- Personalisasi isi dan memungkinkan penggunaan kembali pengetahuan.

Pada penelitian Sipahutar (2013), *Learning Management System* digunakan untuk pembelajaran bahasa pemrograman *Java* pada mata kuliah Algoritma dan Pemrograman di Jurusan Sistem Informasi ITS.

LMS tersebut berisikan tutorial serta kuis yang terkait dengan pembuatan *source code* dalam *Java*. Di dalam *LMS* tersebut juga terdapat penilaian otomatis pada sistem untuk mengecek kesuksesan suatu program yang dibuat oleh mahasiswa. Penilaian tersebutlah yang nantinya akan dijadikan sebagai nilai kuis mahasiswa.

2.4. Plugin Design Patterns

Plug-In merupakan sebuah komponen perangkat lunak yang menambahkan fitur khusus atau fungsionalitas untuk aplikasi perangkat lunak atau *website* yang sudah ada**[8]**. Contoh

umum penggunaan *plug-in* pada *website* adalah penambahan fitur baru seperti *search-engines* ataupun *virus-scanners*. Dalam penelitian Tresnawati (2011), fungsi *SMS Gateway* digunakan sebagai bentuk laporan plagiarisme mahasiswa kepada dosen dalam aplikasi *e-learning* untuk *programming assignment* yang dibuat di *Moodle*. Masalah umum yang dihadapi pengembang aplikasi saat membangun aplikasi adalah bagaimana untuk memungkinkan aplikasi tersebut menjadi "*plug-able*" saat *runtime*. Artinya, untuk memungkinkan kode non-inti untuk memodifikasi cara aplikasi diproses pada saat *runtime*. Oleh karena itu dibutuhkan suatu pola rancangan agar suatu *plug-in* dapat berjalan pada aplikasi utama yang dinamakan *design patterns*.

Design Pattern adalah sebuah istilah dalam Rekayasa Perangkat Lunak (*Software Engineering*) yang mengacu kepada solusi umum yang dapat digunakan secara berulang kali untuk menyelesaikan masalah-masalah umum yang ditemukan dalam desain perangkat lunak**[9]**. *Design pattern* merupakan penjelasan atau *template* yang menunjukkan bagaimana cara menyelesaikan sebuah masalah yang kemudian dapat digunakan di berbagai situasi yang berbeda-beda. Ada banyak *Design Patterns* yang sudah diakui kemampuannya, diterima dan diaplikasikan oleh banyak praktisi. *Design Patterns* yang cukup populer adalah yang diperkenalkan *The Gang of Four (GoF)* - Erich Gamma, Richard Helm, Ralph Johnson dan John Vlissides**[10]**.

Sebuah arsitektur *plugin* yang baik haruslah memiliki *design pattern* yang berkemampuan untuk menuliskan kode *loosely coupled* yang sesuai dengan pemahaman dari *Law of Demeter*. *Loosely coupled* adalah sebuah konsep sistem untuk mengurangi ketergantungan (*interdependency*) dari suatu sistem**[11]**. Sedangkan *Law of Demeter* yang memiliki slogan "*Only talk to your immediate friends*" adalah sebuah metode yang efektif untuk memerangi *coupling***[12]**. Hukum ini

menyatakan bahwa setiap objek hanya boleh memanggil *method* yang dimiliki oleh:

- Dirinya sendiri.
- Parameter yang dikirimkan kepada *method*.
- Setiap objek yang dibentuk.
- Setiap yang memegang objek secara langsung.

Beberapa design pattern yang sesuai dengan *Law of Demeter* adalah *Command, Memento, Callback, Dependency Injection, Abstract Factory,* dan *Builder*. Karakteristik dari *design pattern* tersebut dapat diihat pada tabel 2.8. Pada tabel tersebut dapat dilihat bahwa *Abstract Factory* dan *Dependency Injection* sama-sama memiliki tujuan dan karakteristik yang hampir sama.

Tabel 2.8 Design Patterns

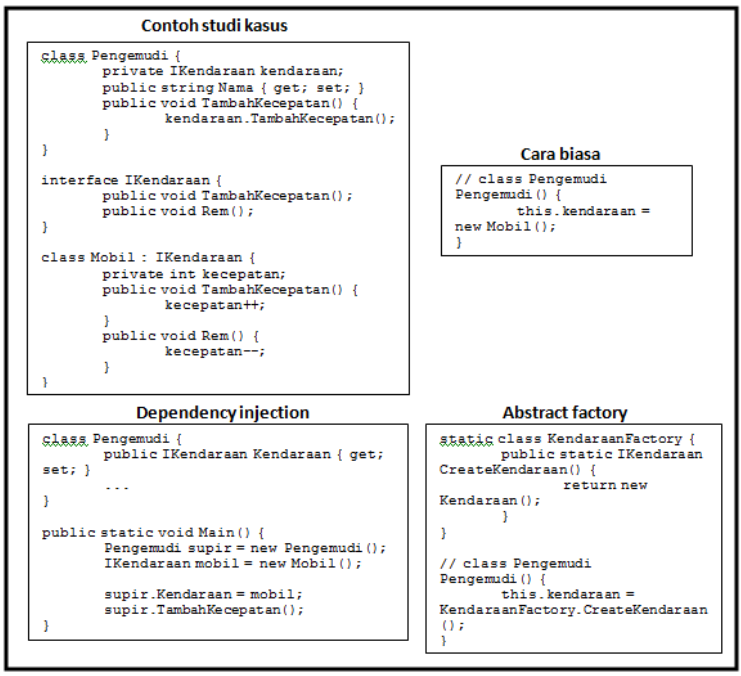
Design Pattern	Tujuan	Karakteristik
Command	Untuk mewakili maupun merangkum informasi yang diperlukan untuk memanggil metode di waktu yang berbeda	Perlu mengeluarkan permintaan untuk objek tanpa mengetahui apapun tentang operasi yang diminta atau penerima permintaan
Memento	Untuk merekam atau menampilkan kembali objek yang pernah ditampilkan sebelumnya	Perlu untuk mengembalikan objek ke keadaan sebelumnya (misalnya operasi "undo" atau "rollback")
Call-Back	Untuk mengeksekusi kembali (callback) argumen pada beberapa waktu yang sesuai	Perlu menggunakan sebuah privilege agar dapat kembali ke keadaan sebelumnya

Design Pattern	Tujuan	Karakteristik
Dependency Injection	Untuk mendapatkan objek lain yang dibutuhkan tanpa harus mencari dan memasukan objek tersebut secara eksplisit	Perlu adanya ketergantungan (dependency) antar objek satu dengan yang lain agar perubahan yang dilakukan tidak signifikan
Abstract Factory	Untuk menyediakan sebuah interface untuk menciptakan obyek yang terkait tanpa perlu menentukan class nyatanya	Digunakan saat ingin menghasilkan objek yang berasal dari salah satu dari beberapa kelas yang berkaitan, dimana objek tersebut dapat mengembalikan beberapa objek lain
Builder	Untuk memisahkan pembangunan object yang rumit dari representasinya sehingga dengan proses konstruksi yang sama dapat menghasilkan representasi yang berbeda	Proses konstruksi harus memungkinkan representasi yang berbeda untuk objek yang dibangun

Abstract Factory dan *Dependency Injection* sama-sama cocok digunakan pada pemrograman berorientasi objek (*OOP*) dimana kedua design pattern ini sama-sama mereferensi atau terhubung dengan program lainnya dan membuat program asli tergantung pada program yang direferensikan. Perbedaan *abstract factory* dan *dependency injection* dapat dilihat pada gambar 2.1. Pada gambar tersebut diketahui bahwa terdapat kelas Pengemudi dimana saat menjalankannya tergantung pada kelas Kendaraan. Untuk menghubungkan kelas Pengemudi dan kelas Mobil dibutuhkan suatu kelas contructor.

Jika dengan cara biasa, kelas Pengemudi menjadi bergantung dengan kelas Mobil yang merupakan *concrete class*.

Oleh karena itu dibutuhkan suatu perantara yaitu sebuah kelas *factory* untuk mengurangi ketergantungan kelas Pengemudi dengan kelas Mobil. Namun akibatnya kelas *factory* yang dibuat akan menjadi *tightly coupled* dengan kelas Kendaraan. Dengan menggunakan *dependency injection*, kelas Pengemudi bisa melakukan tugasnya tanpa harus mengetahui kelas Mobil maupun *factory*.



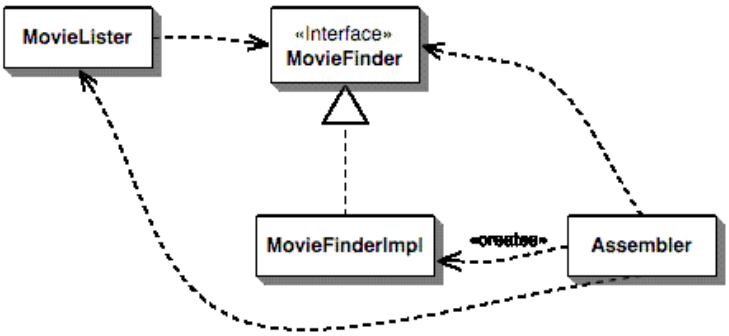
Gambar 2.1 Penerapan Dependency Injection

Dari penjelasan sebelumnya diketahui bahwa *dependency injection* merupakan *design pattern* yang cocok diterapkan pada pengembangan sistem *plugin*. Hal ini dikarenakan *dependency injection* bersifat *loosely coupled*, dimana

ketergantungan antar kelas menjadi longgar dan terkesan terpisah sehingga akan lebih mudah jika dilakukan *testing*.

2.5. Dependency Injection

Dependency Injection terjadi ketika *class* dependensinya diberikan melalui konstruktor mereka, *method*, atau langsung ke *field*. Ide dasar dari *Dependency Injection* memiliki objek yang terpisah, *assembler* yang bertugas mengisikan *field* di *lister class* dengan implementasi yang tepat untuk *finder interface*, dan menghasilkan diagram ketergantungan yang dapat dilihat pada Gambar 2.2[13]. Pada intinya *assembler* berguna untuk menginstansiasi objek-objek yang dibutuhkan oleh suatu aplikasi dan menginjeksi mereka ke dalam objek-objek yang membutuhkannya. Setidaknya ada tiga cara dalam membuat *dependency injection*, yaitu:



Gambar 2.2 Struktur Dependency Injection

2.5.1. Contructor Injection

Injection yang terjadi pada saat pembentukan *method constructor* melalui *passing parameter* dalam konstruktor tersebut. Cara ini banyak digunakan pada *PicoContainerFramework*. Gambar 2.3 menjelaskan bahwa metode ini memerlukan klien untuk memberikan sebuah parameter di konstruktor untuk ketergantungan.

```
// Constructor
Client(Service service) {
    // Save the reference to the passed-in service inside this client
    this.service = service;
}
```

Gambar 2.3 Constructor Injection

2.5.2. Setter Injection

Injection yang terjadi pada saat pengaturan atribut. Cara ini sering digunakan pada *Spring Framework*. Gambar 2.4 menjelaskan bahwa metode ini memerlukan klien untuk memberikan *setter method* untuk setiap ketergantungan.

```
// Setter method
public void setService(Service service) {
    // Save the reference to the passed-in service inside this client
    this.service = service;
}
```

Gambar 2.4 Setter Injection

2.5.3. Interface Injection

Injection yang terjadi di *interface*, cara ini banyak digunakan di *Avalon Framework*. Gambar 2.5 menjelaskan bahwa klien mempublikasikan peran *interface* untuk *setter method* dependensi klien. Hal ini dapat digunakan untuk menetapkan bagaimana *injector* harus berbicara dengan klien ketika menginjeksikan dependensi.

```
// Service setter interface.
public interface ServiceSetter {
    public void setService(Service service);
}

// Client class
public class Client implements ServiceSetter {
    // Internal reference to the service used by this client.
    private Service service;

    // Set the service that this client is to use.
    @Override
    public void setService(Service service) {
        this.service = service;
    }
}
```

Gambar 2.5 Interface Injection

2.6. Jenis-jenis Plagiarisme

Plagiarisme atau sering disebut plagiat adalah penjiplakan atau pengambilankarangan, pendapat, dan sebagainya dari orang lain dan menjadikannya seolah karangan dan pendapat sendiri[14]. Plagiarisme dalam literatur terjadi ketika seseorang mengaku atau memberi kesan bahwa ia adalah penulis asli suatu naskah yang ditulis orang lain, atau mengambil mentah-mentah dari tulisan atau karya orang lain atau karya sendiri (*swaplagiarisme*) secara keseluruhan atau sebagian, tanpa memberi sumber. Tindakan plagiarisme merupakan suatu bentuk pelanggaran hak cipta sehingga *plagiator* atau pelaku plagiarisme dapat dijatuhi hukuman. Hal tersebut dikarenakan secara tidak langsung tindakannya dikategorikan tindakan mencuri karya orang lain. Beberapa jenis teknik plagiarism yang dikenal selama ini, yaitu:

- *Word-for-word plagiarism*: yaitu tindakan plagiarisme dengan cara menyalin setiap kata secara langsung tanpa diubah sedikitpun.
- *Plagiarism of the form of a source*: yaitu tindakan plagiarism dengan cara menyalin dan menulis ulang

kode-kode program tanpa mengubah struktur dan jalannya program.

- *Plagiarism of authorship*: tindakan plagiarism dengan cara mengakui hasil karya orang lain sebagai hasil karya sendiri dengan cara mencantumkan nama sendiri menggantikan nama pengarang sebenarnya.

Ada banyak cara yang dapat diterapkan untuk memodifikasi suatu kode program. Menurut Sutanto (2008) modifikasi-modifikasi yang paling umum dilakukan antara lain:

- *Lexical Change*. Perubahan Leksikal atau *Lexical Change* adalah perubahan yang terjadi pada *source code*. Modifikasi yang dilakukan merupakan modifikasi yang tidak begitu memerlukan pemahaman bahasa pemrograman yang tinggi. Modifikasi yang sering dilakukan meliputi:
  1. Memodifikasi *comment*, dengan menambahkan atau menghapus *comment*.
  2. Memodifikasi *identifier*, dengan mengganti nama *identifier* dan menambahkan atau mengurangi detail judul program.
  3. Memodifikasi *variable*, dengan mendeklarasikan *variable* yang tidak terpakai di program dan memisahkan pendeklarasian *variable*.

- *Structural Change*. Perubahan Struktural atau *Structural Change* adalah perubahan yang terjadi pada struktur program. Modifikasi jenis ini merupakan modifikasi yang membutuhkan pemahaman lebih tentang bahasa pemrograman. Plagiarisme yang dimodifikasi seperti ini lebih sulit untuk dideteksi. Modifikasi yang sering dilakukan meliputi:
  1. Memodifikasi *procedure* dan *function*, dengan mengganti baris program menjadi *procedure* atau *function* dan menambahkan pendeklarasian

procedure atau *function* yang tidak terpakai di program.

2. Memodifikasi *statement*, dengan membuat *statement* yang tidak mempengaruhi hasil, seperti blok *begin-end, repeat-until, while-do, for-to-do, if-then-else, case-of*, dan *assignment*.
3. Memodifikasi *control logic*, dengan mengganti *for-to-do* menjadi *repeat-until*, *for-to-do* menjadi *while-do*, *while-do* menjadi *repeat-until*, *repeat-until* menjadi *while-do* dan *case-of* menjadi *if-then-else.*

2.7. Tokenization

Proses pindai atau yang biasa kita sebut *scanning* merupakan proses melihat dengan teliti dan seksama. Sedangkan *scanner* atau pemindai adalah alat yang dapat membaca data dengan teliti dan seksama. *Scanner* bertugas untuk melakukan *lexical analyzer*, yaitu mengidentifikasi semua besaran yang membangun suatu bahasa pada suatu *source code*[3]. *Scanner* akan menerima input berupa *source code* dan memecahnya menjadi sekumpulan *lexam* dan kemudian diubah menjadi *token*. Proses inilah yang biasa disebut *tokenization*. Yang dimaksud *token* adalah hasil atau *output*ketika karakter dari program *Java* dikelompokkan menjadi *symbol*[2]. *Token* dapat berupa *identifier, keywor*d, pemisah, *operator, literal* dan komentar.

2.8. Rabin-Karp

Sebagian besar dari sistem pendeteksi yang sudah ada, sistem untuk mendeteksi plagiarisme banyak yang menggunakan metode perbandingan antara dua *source code*[15]. Salah satunya adalah sistem yang menggunakan metode algoritma *Rabin Karp*. Algoritma *Rabin Karp* adalah algoritma pencarian kata yang mencari sebuah pola berupa *substring* dalam sebuah teks menggunakan *hashing*[16]. Algoritma ini sangat efektif untuk pencocokan kata dengan pola banyak.

Algoritma *Rabin Karp* banyak diimplementasikan pada sistem yang berguna untuk mendeteksi plagiarisme.

Kunci utama algoritma *Rabin Karp* adalah pada penghitungan hashing untuk menemukan sebuah *substring* dalam sebuah teks. *Hashing* adalah metode yangmenggunakan fungsi*hash* untuk mengubah suatu jenis data menjadi beberapa bilangan bulat sederhana[5].

Seperti yang sudah dijelaskan sebelumnya, algoritma *Rabin-Karp* menjadi *similarity method* yang populer. Dari sistem-sistem untuk mendeteksi plagiarism yang sudah ada, sebagian besar mengimplementasikan algoritma *Rabin-Karp* untuk mendeteksi plagiarisme. MOSS dan JPlag merupakan salah satu sistem yang menggunakan algoritma tersebut.

2.8.1. Measure Of Software Similarity (MOSS)

MOSS adalah sistem yang berjalan otomatis dalam menentukan kesamaan program. MOSS telah dikembangkan pada tahun 1994 oleh Alex Aiken yang tersedia dalam bahasa pemrograman C, C++, Java, atau Pascal serta dalam dua platform yaitu UNIX dan Windows [17]. MOSS melakukan perubahan pada source code dari file sebenarnya menjadi source code yang terpisah-pisah. Setelah melalui proses tersebut *source code* diubah menjadi token. Kemudian token akan dideteksi kesamaannya dengan file *source code* yang lain menggunakan algoritma pembanding yang ada pada sistem MOSS [6]. Sistem MOSS tersedia dalam aplikasi berbasis web.

2.8.2. JPlag

JPLag dibangun dan dikembangkan oleh Guido Malpohl dari Jurusan Teknik Informatika Universitas Karlsruhe. Sistem ini dapat mendeteksi kesamaan pada source code Java, C, C ++, dan Scheme. Jplag tersedia sebagai aplikasi berbasis web dan dapat digunakan secara gratis namun kita harus mempunyai akun yang ada di link http://jplag.ipd.kit.edu/ terlebih dahulu.

Menurut Tresnawati (2011) JPlag memiliki beberapa karakteristik sebagai berikut:

- JPlag tersedia dalam layanan web.
- JPlag memiliki user interface yang friendly sehingga user dapat memahami hasil dari sistem.
- JPlag merupakan sumber daya yang efisien.
- JPlag memiliki kinerja yang bagus dalam mendeteksi plagiarisme.

2.9. Levenshtien Distance

Menurut Stein (2006) metode pendeteksian plagiarisme dibagi menjadi tiga bagian, yaitu *Substring Matching*, *Keyword Similarity*, dan *Fingerprint Analysis*. Dari ketiga metode pendeteksian plagiarisme tersebut, metode *Substring Matching* merupakan metode yang paling efektif dalam mendeteksi plagiarisme. Metode *Substring Matching* diterapkan dengan membandingkan seluruh isi yang ada dalam dokumen. Walaupun proses tersebut membutuhkan waktu lama, metode ini sangat efektif karena membandingkan seluruh isi dokumen bukan sebagian.

Hal ini lebih diperkuat oleh Dewandono (2013) bahwa metode *Substring Matching* sangat efektif dibandingkan metode *Fingerprint Analysis*. Walaupun salah satu metode *Fingerprint Analysis* yaitu algoritma *Rabin-Karp* digunakan untuk mendeteksi keakuratan salinan antar dokumen, metode tersebut hanya membandingkan sebagian isi dokumen saja dan mengabaikan beberapa rincian seperti tanda baca.

Salah satu yang termasuk dalam *Substring Matching* adalah algoritma *Levenshtien Distance*. Algoritma ini bertujuan untuk mengukur jarak antara dua *string* yang ukurannya tidak sama dengan menghitung jumlah pengoperasian yang perlu dilakukan untuk mengubah *string* yang satu menjadi *string* yang kedua yang diperbandingkan. Pengoperasian yang dilakukan termasuk operasi penambahan, penghapusan, dan penggantian karakter. Metode *Levenshtien Distance* dapat

digunakan dalam berbagai macam hal yang berhubugan dengan *substring matching*[18], yaitu:

- *File revision*
- *Remote screen update problem*
- *Spelling correction*
- *Plagiarism detection*
- *Molecular biology*
- *Speech recognition*

Pertama-tama *Levenshtien Distance* mengubah dua program yang akan dibandingkan menjadi dua buah *string* panjang. Setelah itu dua buah *string* tersebut akan membentuk *matriks* dua dimensi. *String* yang lebih panjang akan diletakkan pada kolom *matriks* dan *string* yang lebih kecil diletakkan pada baris *matriks*. Langkah-langkah algoritma *Levenshtien Distance* adalah sebagai berikut:

1. Set n sebagai panjang string1.
   Set m sebagai panjang string 2.
2. Jika n=0, maka keluar dari program.
   Jika m=0, maka keluar dari program.
3. Bentuk matriks dengan jumlah baris m dan jumlah kolom n.
4. Inisialisasi baris pertama 0..n.
   Inisialisasi kolom pertama 0..m.
5. Cek setiap karakter string1 (dengan i dari 1 sampai n).
   Cek setiap karakter string2 (dengan j dari 1 sampai m).
6. Jika string1[i] sama dengan string2[j] maka cost = 0.
   Jika string1[i] tidak sama dengan string2[j] maka cost = 1.
7. Elemen lainnya (matriks[i,j]) akan diisi dengan nilai minimum dari:
   - Sel di atasnya ditambah 1: matriks[i-1,j] + 1
   - Sel di sebelah kirinya ditambah 1: matriks[i, j-1] + 1

- Sel pada diagonal atas dan kiri ditambah cost: matriks[i-1, j-1] + cost

8. Setelah semua sel terisi, maka jumlah perbedaan antara 2 string dapat dilihat sel kanan bawah pada matriks[n,m].

Berikut ini contoh dua buah string yang dibandingkan. Kita letakkan kata “KOLUSI” dan “KORUPSI” secara horizontal dan vertikal dalam baris dan kolom pada matriks. Gambar 2.6 menampilkan matriks kedua string tersebut.

		K	O	L	U	S	I
	0	1	2	3	4	5	6
K	1						
O	2						
R	3						
U	4						
P	5						
S	6						
I	7						

Gambar 2.6 Matriks Perbandingan

Setelah melakukan proses *Levenshtien Distance*, maka *matriks* sebelumnya akan menjadi seperti pada Gambar 2.7.

		K	O	L	U	S	I
	0	1	2	3	4	5	6
K	1	0	1	2	3	4	5
O	2	1	0	1	2	3	4
R	3	2	1	1	2	3	4
U	4	3	2	2	1	2	3
P	5	4	3	3	2	2	3
S	6	5	4	4	3	2	3
I	7	6	5	5	4	3	2

Gambar 2.7 Matriks Setelah Levenshtien Distance

Pada gambar tersebut, dapat kita ketahui bahwa perubahan yang dilakukan berjumlah 2 yaitu menghapus karakter “P” pada “KORUPSI” dan mengganti karakter “R” menjadi “L”. Untuk mengetahui kemiripan *source code*, algoritma *Levenshtien Distance* dapat dirumuskan dalam persamaan sebagai berikut:

$$Similarity = \left(1 - \frac{LD}{T}\right) \times 100\%$$

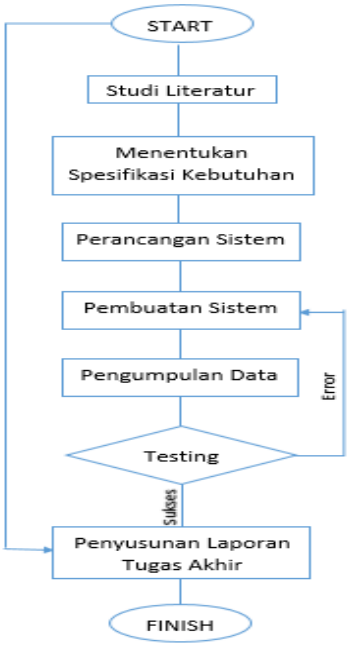
Dimana:
LD = Hasil dari Levenshtien Distance
T = Panjang karakter terpanjang dari kedua string
Similarity = Persentase Perbedaan

Dari contoh sebelumnya jika kita masukkan nilai Levenshtien Distance pada persamaan diatas maka hasil kemiripannya adalah 72%.

(Halaman ini sengaja dikosongkan)

BAB III
METODOLOGI PENELITIAN

Bagian ini menjelaskan metodologi yang digunakan dalam pengerjaan tugas akhir ini. Metodologi ini diperlukan sebagai panduan secara terstruktur dan sistematis dalam pengerjaan tugas akhir. Gambaran dari metodologi bisa dilihat pada Gambar 3.1. Berikut penjelasan dari beberapa tahap yang ada dalam metodologi penelitian:



Gambar 3.1 Metodologi Penelitian

3.1. Studi Literatur

Tahap ini adalah tahap melakukan studi literatur yang dibutuhkan dalam pengembangan aplikasi. Tahap ini bertujuan untuk mencari sumber-sumber pendukung untuk melakukan penelitian. Studi Literatur bertujan untuk mengetahui penelitian-penelitian apa saja yang pernah dilakukan dan apa

saja perbedaan antar penelitian yang pernah dilakukan sebelumnya.

3.2. Menentukan Spesifikasi Kebutuhan

Pada tahap ini ditentukan kebutuhan-kebutuhan fungsional yang menjadi fitur dari sistem. Analisa kebutuhan digunakan untuk mendapatkan informasi, model, spesifikasi tentang sistem yang akan dibuat.

3.3. Perancangan Sistem

Pada tahap ini sistem akan dirancang sedemikian rupa agar dapat digunakan sebagai acuan pembuatan sistem. Pada tahap ini ada beberapa proses antara lain:

- Membuat *GUI Storyboard*. Setelah analisis kebutuhan, akan dibuat desain *GUI* sebagai gambaran antarmuka dari sistem.
- Membuat *Domain Model*. Selanjutnya adalah tahap pembuatan *domain model* untuk menggambarkan permasalahan ke dalam bentuk *domain* sehingga mudah dipahami.
- Membuat *Use Case Model*. Pada tahap ini akan dijelaskan mengenai fungsi yang ada di dalam sistem, objek yang mempengaruhi sistem, dan hubungan antara sistem dan aktor.
- Membuat *Robustness Analysis*. Pada tahap ini deskirpsi dari *use case* akan dianalisa. Hal ini bertujuan untuk menentukan objek apa saja yang terlibat di setiap *use case*.
- Membuat *Sequence Model*. Tahap ini bertujuan untuk menggambarkan secara detail bagaimana implementasi dari *use case*.

3.4. Pembuatan Sistem

Tahap ini adalah tahap pembuatan sistem yang disesuaikan dengan desain yang telah dibuat sebelumnya. Setelah pembuatan sistem selesai, tahapan selanjutnya adalah

pengumpulan data dari mahasiswa yang nantinya akan digunakan pada tahap uji coba sistem.

3.5. Pengumpulan Data

Tahap ini adalah tahap pengumpulan data yang berupa *source code* dari mahasiswa. Data yang akan diuji akan dibagi menjadi dua kategori yaitu *source code* yang sengaja menyalin dari *source code* lain dan *source code* yang dikerjakan sendiri. Untuk *source code* yang sengaja menyalin dari *source code* lain dibagi menjadi 6 *source code* yang masing-masing telah dimodifikasi pada tingkat yang berbeda-beda. Modifikasi yang dilakukan yaitu modifikasi *comment*, modifikasi *identifier*, modifikasi *variable*, modifikasi *procedure* atau *function*, modifikasi *statement*, dan modifikasi *control logic*.

3.6. Testing

Tahap ini dilakukan setelah aplikasi selesai dibangun. Uji coba yang dilakukan berupa pengujian kemiripan terhadap dua *source code* dari dua mahasiswa yang berbeda. Dalam uji kemiripan ini akan diketahui pada saat modifikasi apa suatu *source code* yang tidak mirip dianggap mirip, suatu *source code* yang tidak mirip dianggap tidak mirip, suatu *source code* yang mirip dianggap tidak mirip, dan suatu *source code* yang mirip dianggap mirip. Apabila terjadi kegagalan uji kemiripan pada sistem, maka penulis perlu memperbaiki bagian yang menjadi permasalahan. Apabila pengujian aplikasi berjalan lancar, maka akan dilanjutkan dengan membuat kesimpulan dan saran penelitian laporan tugas akhir.

3.7. Penyusunan Laporan Tugas Akhir

Laporan tugas akhir ini berisikan dokumentasi dari semua tahapan yang telah dilakukan sebelunya beserta kesimpulan dari pengerjaan tugas akhir. Tujuan dari dokumentasi ini adalah memberikan informasi kepada pembaca dan juga meminta kritik dan saran sehingga penulis dapat meningkatkan kemampuan dalam hal pengembangan aplikasi kedepannya.

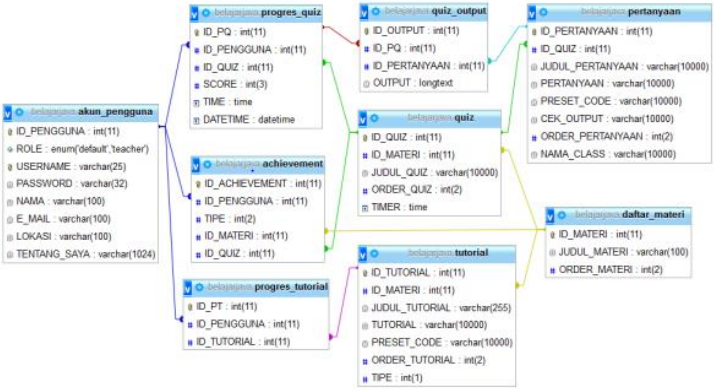
(Halaman ini sengaja dikosongkan)

BAB IV
ANALISIS KEBUTUHAN DAN DESAIN SISTEM

Pada bab ini akan dibahas mengenai hal-hal yang terkait dengan kebutuhan dan desain sistem. Perancangan sistem menggunakan *ICONIX Process* yang terdiri dari pembuatan *Graphical User Interface (GUI) Storyboard*, *Domain Model*, *Use Case Model*, *Robustness Diagram*, *Sequence Diagram*,*Class Diagram*, dan *Test Case*.

4.1 Gambaran Umum *E-learning* Belajar *Java*

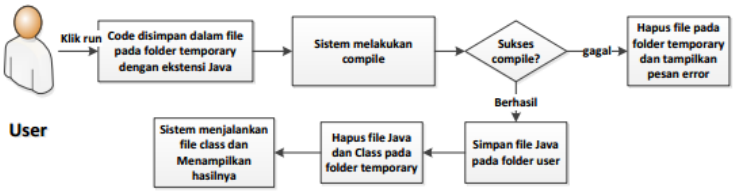
Tujuan utama dari *e-learning* Belajar *Java* adalah untuk membantu mahasiswa dalam mempelajari *Java*. Dimana aplikasi ini memiliki fitur tutorial dan *quiz* untuk mengetahui seberapa jauh kemampuan mahasiswa dalam memahami pemrograman *Java*. Gambar 4.1 merupakan desain database dari aplikasi Belajar *Java*.



Gambar 4.1 Desain Database Belajar Java

Dalam setiap quiz terdapat nilai dari mahasiswa yang menggambarkan bagaimana perkembangan belajar mahasiswa tersebut. Hasil pengerjaan dari mahasiswa akan disimpan ke dalam direktori sesuai dengan nomor id mahasiswa dan id pertanyaan. Hal ini bertujuan agar mahasiswa dapat mengambil kembali file pengerjaan jika dibutuhkan sewaktu-

waktu. Untuk mengetahui bagaimana alur sistem *e-learning* melakukan *compiling code* yang telah ditulis oleh mahasiswa dapat dilihat pada gambar 4.2.



Gambar 4.2 Alur Proses Compile

Namun dengan adanya keuntungan tersebut, tidak menutup kemungkinan jika sewaktu-waktu ada mahasiswa yang sengaja menduplikat hasil pengerjaan dari mahasiswa lain yang terdapat di file histori tersebut. Oleh karena itu dibutuhkan suatu fungsi tambahan agar sistem dapat mendeteksi tingkat kemiripan *source code* dari hasil quiz antar mahasiswa.

Agar fungsi tersebut bersifat *plug-able* dimana dapat diaktifkan ataupun dinonaktifkan ketika aplikasi berjalan, maka fungsi tersebut dibuat ke dalam bentuk *plugin*. Hal ini dikarenakan jika menambah fungsi langsung ke dalam aplikasi, maka diperlukan juga perancangan ulang terhadap aplikasi tersebut. Apabila fungsi dijadikan ke dalam bentuk *plugin*, aplikasi yang sudah ada tidak perlu diubah kembali desainnya. Yang diperlukan hanyalah membuat sebuah *method* pada *plugin* yang dijalankan ketika *plugin* diaktifkan ataupun berfungsi sebagai pemanggil entitas di aplikasi yang diperlukan oleh *plugin*.

4.2 Gambaran Umum *Wordpress*

Dalam membuat analisis kebutuhan untuk sistem *plugin*, dibutuhkan suatu gambaran sistem lain sebagai referensi dalam menentukan kebutuhan *plugin*. Pada bagian ini, digunakan referensi dari salah satu sistem yang banyak digunakan *developer* lain untuk mengembangkan sistem *plugin* yaitu *Wordpress*.

Pada *wordpress*, sebuah *plugin* dapat berjalan karena adanya menu *Plugin Manager*. Dimana menu tersebut digunakan untuk mengatur agar plugin bisa berjalan pada *Wordpress*. *Wordpress* memberikan tampilan menu *Plugin Manager* tersebut untuk *user* yang berperan sebagai *administrator*. Dengan adanya *Plugin Manager* tersebut, *administrator* dapat melakukan beberapa aktivitas antara lain:

1. Menginstall *plugin*,
2. Menghapus *plugin*,
3. Mengaktifkan *plugin*,
4. Menonaktifkan *plugin*,
5. Mengatur konfigurasi *plugin*.

Agar dapat terbaca oleh *Plugin Manager* dan terinstall dalam aplikasi, ada dua hal yang harus diperhatikan dalam membuat *plugin* pada *Wordpress*, yaitu:

1. Nama *file* php utama pada *plugin* harus sama dengan nama *plugin*. Misalnya nama *plugin* adalah "Abcde" maka nama *file* php utama pada *plugin* adalah abcde.php.
2. Harus terdapat *Standart Plugin Information* pada *header* dari *file* php utama *plugin* seperti yang terdapat pada gambar 4.3. Tanpa adanya informasi tersebut, maka *plugin* tidak akan terbaca dan tidak dapat terpasang pada aplikasi.

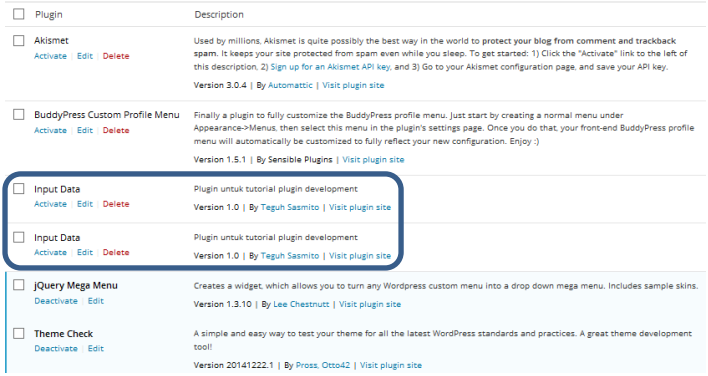
```
<?php
/*
Plugin Name: Magic Plugin
Plugin URI: http://example.com/magic-plugin
Description: Magic Plugin performs magic
Version: 2.3
Author: Mr. Magic
Author URI: http://example.com/
*/
```

Gambar 4.3 Standard Plugin Information

Walaupun setiap *plugin* memiliki karakteristik yang unik, namun *plugin manager* yang ada di *Wordpress* memiliki kelemahan dalam memanajemen *plugin*. Kelemahan tersebut adalah ketidakmampuan *Wordpress* dalam menyaring informasi *plugin*. Akibatnya jika ada *plugin* yang diinstal lebih

dari satu kali, maka di dalam *plugin manager* nantinya terdapat dua atau lebih *plugin* kembar yang terinstall. Untuk penjelasan lebih detail dapat dilihat pada gambar 4.4.

Untuk menghubungkan antara *plugin* dengan aplikasi, *wordpress* membuat *plugin manager* dengan menggunakan *design pattern Singleton*. *Singleton* hanya melibatkan satu kelas yang bertanggung jawab untuk menginstansiasi dirinya sendiri dan pada saat yang bersamaan menyediakan akses secara global terhadap *instance* tersebut. Pada *pattern singleton*, *instance* bisa diakses dari manapun tanpa harus memanggil *constructor* dari kelas *instance* tersebut. Di *wordpress*, *singleton* membuat *instance* untuk menyimpan data user yang sedang aktif atau tercatat sedang *login* dalam aplikasi. Dalam hal ini *user* yang bisa bisa mengakses *plugin manager* adalah *user* yang tercatat sedang *login* sebagai *administrator*.



Gambar 4.4 Plugin Manager Wordpress

Namun *design pattern Singleton* juga mempunyai kelemahan. Karena *singleton* selalu bekerja menggunakan metode *static*, hal ini menjadikan aplikasi yang dibuat akan susah untuk di-unit-testkan. Menurut Bergmann (2010), *design patternDependency Injection* merupakan cara alternatif mengurangi adanya penulisan metode *static* maupun *singleton*.

Untuk lebih memahami keunggulan *dependency injection* dari *singleton*, tabel 4.1 dibawah ini menjelaskan bagaimana penerapan *singleton* maupun *dependency injection* dalam kasus pembuatan Jadwal Kuliah.

Tabel 4.1 Perbedaan Singleton dan DI

Perbedaan	Singleton	Dependency Injection
Cara Kerja	Jika menggunakan Singleton,yang pertama dilakukan adalah membuat objek yang diperlukan dalam membuat Jadwal Kuliah (contoh: Dosen, Ruangan, Waktu).	Jika menggunakan dependency injection, yang perlu dilakukan hanyalah memanggil objek yang dibutuhkan oleh Jadwal Kuliah tanpa perlu membuatnya terlebih dahulu.
Karakteristik	Perlu mereferensikan global scope	Menghilangkan global scope dengan memberikan ketergantungan yang diperlukan melalui konstruktor/method

4.3 Gambaran Umum Sistem *Similarity Testing* lain

Salah satu sistem pendeteksi kemiripan pada *source code* adalah sistem yang dibuat oleh Tresnawati pada penelitian yang berjudul "*Plagiarism Detection System Design for Programming Assignment in Virtual Classroom Based on Moodle*". Sistem ini berjalan sebagai fungsi tambahan pada *E-learning*. Berikut beberapa fungsi yang ada pada sistem ini:

- Memungkinkan *teacher* untuk mendeteksi plagiarisme dan kecurangan pada tugas siswa yang telah dikumpulkan. Sistem membaca tugas yang telah dikumpulkan dan memasukannya ke algoritma untuk menemukan tingkat kesamaan antar *source code*.
- *Teacher* dapat menampilkan hasil laporan yang berisi semua tugas yang dikumpulkan dan persentase kemiripan setiap tugas dengan orang lain.
- Sistem mampu menampilkan perbandingan isi *file* yang memiliki kesamaan.

- Sistem secara otomatis dapat mengirim peringatan kepada siswa yang terdeteksi melakukan tindak plagiarisme dalam bentuk peringatan *SMS*.

4.4 Gambaran Umum Sistem*Plugin*

Sistem ini merupakan sebuah *plugin* yang akan dipasang pada aplikasi *e-learning* pembelajaran *Java*, dimana sistem ini berfungsi untuk mendeteksi kemiripan *source code* yang dibuat oleh pengguna aplikasi saat mengerjakan pertanyaan di menu *quiz*. Setelah mendeteksi kemiripan, sistem ini mengirimkan pesan pengingat ke *email teacher* berupa *link* hasil tes kemiripan.

4.5 Analisis Kebutuhan

Dalam melakukan desain terhadap sistem, dibutuhkan beberapa kebutuhan yang perlu dispesifikasikan berdasarkan hasil analisis kebutuhan *user*, kebutuhan fungsional, dan arsitektur sistem.

4.5.1 Analisis Kebutuhan *User*

Berdasarkan analisis yang dilakukan pada *e-learning*Belajar*Java* dan *plugin* yang ada di *Wordpress*, didapatkan beberapa aktivitas yang dibutuhkan pada sistem.Sistem yang dibuat menyediakan beberapa jenis pengguna yang masing-masing mempunyai aktivitas pada sistem. Berikut beberapa aktivitas berdasarkan jenis pengguna:

1. Administrator/Admin
   Pada sistem ini, Administrator merupakan pengguna yang dapat mengoperasikan menu *plugin manager*agar*plugin* dapat berjalan di aplikasi. Pembuatan user Admin didasarkan pada hasil analisis dari fitur yang ada di *Wordpress*. Berikut beberapa aktivitas yang dapat dilakukan oleh administrator:
   - Menambah atau menginstall *plugin*
   - Menghapus atau menguninstall *plugin*

- Mengaktifkan *plugin*
- Menonaktifkan *plugin*
- Melihat daftar *Plugin*
- Mengatur konfigurasi *plugin*

2. *Teacher*
   Pada sistem ini, *Teacher* merupakan pengguna yang dapat merasakan fungsi dari *plugin* yang terdapat pada aplikasi. Berikut beberapa aktivitas yang dapat dilakukan oleh *Teacher*:
   - Melihat *report* tes kemiripan per Pertanyaan
   - Melakukan tes ulang

4.5.2 Analisis Kebutuhan Fungsional

Berdasarkan hasil analisis kebutuhan user yang berasal dari hasil analisis kebutuhan dari *e-learning* Belajar Java dan fitur yang ada di *Wordpress* dan hasil analisis dari penelitian yang dilakukan oleh Tresnawati, didapatkan fungsi-fungsi yang perlu disediakan oleh aplikasi elearning maupun sistem *plugin*. Berikut beberapa daftar fungsi aplikasi yang telah didapat dari analisis kebutuhan *plugin* dan kebutuhan user:

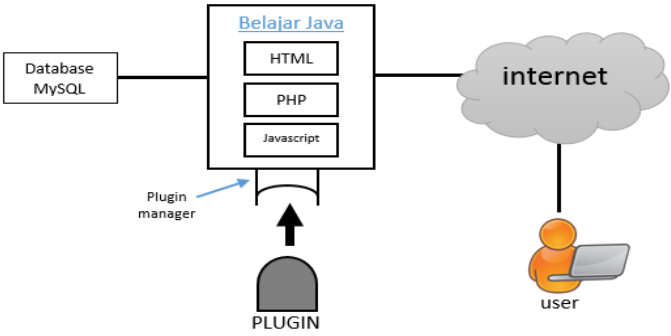
Tabel 4.2 *Functional Requirement*

KODE	Fungsi	Kategori	Sumber
F-1	Admin dapat menginstall plugin	Plugin Manager	[19]
F-2	Admin dapat meng-*uninstall* plugin	Plugin Manager	[19]
F-3	Admin dapat mengaktifkan plugin	Plugin Manager	[19]
F-4	Admin dapat menonaktifkan plugin	Plugin Manager	[19]
F-5	Admin dapat melihat daftar plugin	Plugin Manager	[19]
F-6	Admin dapat mengatur nilai konfigurasi sistem plugin	Plugin Manager	[19]
F-7	Sistem dapat melakukan tes kemiripan source code antar mahasiswa dari setiap	Plugin	[6]

KODE	Fungsi	Kategori	Sumber
	jawaban yang telah dikumpulkan		
F-8	*Teacher* dapat melihat hasil tes kemiripan per pertanyaan yang dikerjakan oleh pengguna berupa persentase kemiripan	Plugin	[2]
F-9	Sistem dapat mengirimkan pesan pengingat dari hasil tes kemiripan ke email yang sudah tercatat di akun *teacher*.	Plugin	[6]
F-10	*Teacher* dapat membuat tes ulang dari setiap pertanyaan yang dikerjakan oleh mahasiswa	Plugin	[2]

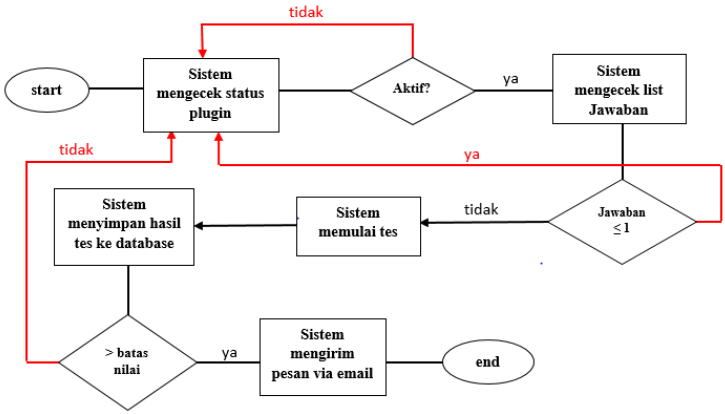
4.5.3 Arsitektur Sistem

Sistem *plugin* ini akan dirancang agar dapat berjalan pada aplikasi Belajar Java, dimana aplikasi tersebut menggunakan *PHP* sebagai *server scripting* dan menggunakan *MySQL* untuk penyimpanan data. Aplikasi dirancang berbasis web sehingga dapat diakses oleh pengguna melalui berbagai perangkat. Plugin manager dirancang agar dapat memanajemen plugin yang akan dipasang pada aplikasi.



Gambar 4.5 arsitektur sistem

Sistem plugin mulai berjalan dengan mengecek status keaktifan plugin. Setelah itu sistem mengecek daftar jawaban yang telah tersimpan di direktori jawaban. Kemudian sistem mengecek keterangan pada database apakah jawaban sudah pernah di tes dengan jawaban yang lain atau tidak. Jika masih belum, sistem akan mengecek kemiripan antar jawaban. Jika sudah selesai, hasil tes akan disimpan di database dan sistem akan kembali mengecek jawaban satu menit kemudian. Jika hasil tes ada yang melebihi batas nilai yang ditentukan maka sistem akan mengirimkan pesan ke email teacher. Flowchart proses jalannya sistem dapat dilihat pada gambar 4.6.



Gambar 4.6 *flowchart* sistem

4.6 Desain Sistem

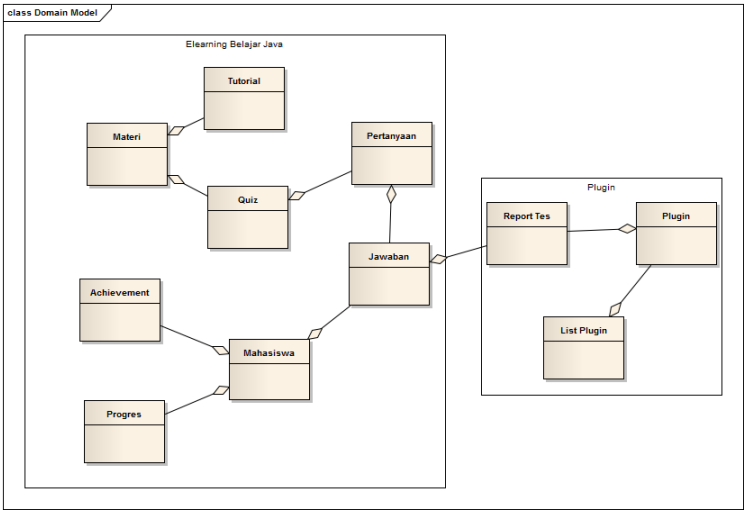
Perancangan sistem pada sistem *plugin* ini menggunakan *ICONIX Process* yang terdiri dari pembuatan *Graphical User Interface (GUI) Storyboard*, *Domain Model*, *Use Case Model*, *Robustness Diagram*, *Sequence Diagram*, *Class Diagram*, dan *Test Case*.

4.6.1 Pemanfaatan *GUI Storyboard*

Graphical User Interface (GUI) Storyboard digunakan sebagai gambaran antarmuka dari sistem. Tujuannya agar lebih mudah dalam penjabaran model dan pembuatan use case pada sistem. Pada Lampiran A terlampir *GUI* pada sistem yang terdiri dari halaman atau *view* yang berbeda-beda tergantung pada jenis pengguna yang mengakses sistem.

4.6.2 Pemanfaatan *Domain Model*

Domain Model merupakan sekumpulan dari *entity* di *database* yang saling berelasi. Salah satu fungsi *domain model* adalah menyamakan istilah yang akan pakai diproses selanjutnya. Berikut adalah *domain model* dari sistem yang dibangun.



Gambar 4.7 Domain Model

4.6.3 Pemanfaatan *Use Case Model*

Use case diagram merupakan model diagram *UML* yang digunakan untuk menggambarkan requirement fungsional yang diharapkan dari sebuah sistem.*Use case model* merupakan kumpulan diagram yang menggambarkan aktivitas yang dilakukan oleh aktor dalam sistem. Aktor yang berhubungan dengan sistem ini antara lain Admin dan Teacher. Berikut daftar *use case diagram* untuk setiap aktivitas dalam sistem. Detail *use case diagram* dan deskripsi dari tiap *use case* dijelaskan pada Lampiran B.

Tabel 4.3 Daftar Use Case

KODE	USE CASE
UC-01	Login Admin
UC-02	Logout Admin
UC-03	Install Plugin
UC-04	Melihat Daftar Plugin
UC-05	Uninstall Plugin
UC-06	Mengaktifkan Plugin
UC-07	Menon-aktifkan Plugin

KODE	USE CASE
UC-08	Mengatur Nilai Konfigurasi Sistem
UC-09	Melakukan Tes Kemiripan
UC-10	Melihat Report Tes per Quiz
UC-11	Melakukan Tes Ulang

4.6.4 Pemanfaatan *Robustness Analysis*

Robustness analysis menggambarkan setiap *use case* secara visual untuk menjelaskan secara detail proses yang terjadi di dalam *use case*. Lampiran C melampirkan *robustness diagram* untuk setiap *use case*.

4.6.5 Pemanfaatan *Sequence Model*

Sequence Diagram menggambarkan urutan proses pada tiap *use case* secara detail disertai dengan proses yang terjadi pada tiap bagian dalam diagram berdasarkan urutan yang telah ditentukan. *Sequence diagram* yang ada pada tiap *use case* dicantumkan pada Lampiran D.

4.6.6 Pemanfaatan *Class Model*

Class Diagram adalah diagram yang menunjukan kelas-kelas yang ada dari sebuah sistem beserta korelasinya. Tujuan dari *class diagram* adalah menggambarkan seluruh objek yang ada secara detail agar lebih dalam tahap pemrogramman aplikasi. Rancangan *class diagram* dapat dilihat pada Lampiran E.

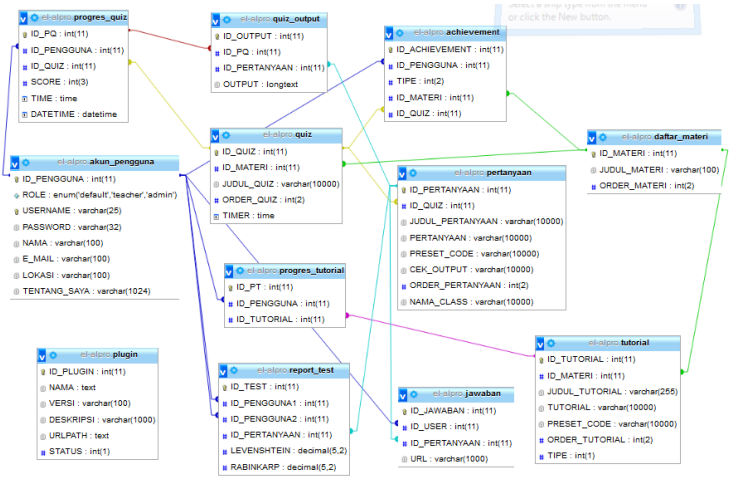
4.6.7 *Test Case*

Test Case merupakan tahap pengujian pada setiap *use case* yang telah dibuat. Test Case dibuat setelah aplikasi selesai dibuat. Tujuan dari pembuatan *test case* adalah untuk mengetahui apakah sistem sudah berjalan sesuai dengan desain yang telah dibuat.Test case dibuat dalam format tabel seperti yang digambarkan pada Tabel 4.4. Test Case dari sistem ini dapat dilihat pada Lampiran F.

Tabel 4.4 contoh tabel *test case*

No	Aksi Test	Data Test	Hasil yang Diharapkan	Hasil Aktual	Sukses / Gagal
1	User mengisikan username dan password kemudian mengklik tombol login	Username = admin Password = admin123	Sistem menampilkan halaman admin	User masuk ke halaman admin	Sukses

4.6.8 Desain *Database*



Gambar 4.8 Desain database sistem

Database pada sistem menggunakan *database MYSQL*. Database yang dirancang menyesuaikan pada desain database yang ada di aplikasi “Belajar Java”. Dari penyesuaian tersebut didapat 2 macam tabel tambahan dengan penjelasannya adalah sebagai berikut:

Plugin
Berisikan informasi data plugin yang ada di aplikasi seperti nama plugin, versi, deskripsi, dan url direktori plugin.
Jawaban
Berisikan informasi mengenai jawaban mahasiswa seperti judul pertanyaan, nama pengguna, dan direktori jawaban.
Report_tes
Berisikan data-data yang berhubungan dengan plugin pendeteksi kemiripan pada source code. Data-data tersebut antara lain nama pengguna, judul pertanyaan, dan persentase kemiripan.

BAB V
IMPLEMENTASI DAN UJI COBA

Pada bab ini dijelaskan proses tahap implementasi dan uji coba pada sistem *plugin*. Adapun beberapa pembahasan antara lain lingkungan implementasi, struktur direktori, penulisan kode program, dan uji coba.

5.1. Lingkungan Implementasi

Sistem plugin ini dikembangkan menggunakan sebuah perangkat keras yang spesifikasinya dapat dilihat pada tabel 5.1. Sedangkan untuk perangkat lunak, sistem plugin dikembangkan menggunakan beberapa teknologi terkait yang spesifikasinya dapat dilihat pada tabel 5.2.

Tabel 5.1 spesifikasi *hardware* untuk implementasi

Aplikasi	Perangkat Keras	Spesifikasi
Plugin	NoteBook	Prosesor : Intel Pentium dual-core T2130(1.86GHz)
		Memori : 1 GB
		Sistem Operasi : Windows 8

Tabel 5.2 spesifikasi *software* untuk implementasi

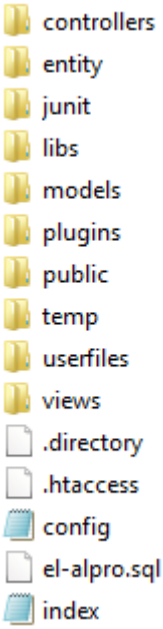
Aplikasi	Perangkat Lunak	Spesifikasi
Plugin	Web Server	Apache 2.4.3
	Database	MySQL 5.0.10
	Bahasa Pemrograman	PHP 5.4.7
	Editor	Notepad++

5.2. Struktur Direktori

Dalam pengembangan sistem plugin ini, ada beberapa folder yang ditambah antara lain:

a. Entity: Folder terdiri dari file yang berisikan objek yang merepresentasikan tabel.
b. Plugins: Folder berisikan file-file yang berhubungan dengan plugin.

Struktur direktori dari aplikasi “Belajar Java” yang telah ditambahkan dua folder diatas dapat dilihat pada gambar 5.1.



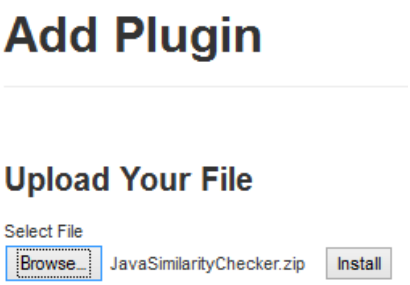
Gambar 5.1 direktori Belajar Java

5.3. Implementasi Fungsi

Pada pengembangan sistem plugin ini terdapat beberapa fungsi-fungsi utama yang diimplementasikan. Implementasi fungsi tersebut dijelaskan pada subbab-subbab berikut.

5.3.1. Penginstallan Plugin

Untuk menambahkan plugin, sistem terlebih dahulu menampilkan halaman tambah plugin. Setelah itu Admin mengupload file .zip kemudian mengklik tombol Install seperti pada gambar 5.2.



Gambar 5.2 halaman *add plugin*

Ketika Admin mengklik tombol Install, sistem mengecek apakah file merupakan file dengan ekstensi .zip atau tidak. Setelah itu sistem akan membuat direktori dengan nama folder yang sesuai dengan nama file. zip. Jika ada nama folder yang sama, sebelumnya sistem akan menghapus direktori yang lama dan menggantinya dengan direktori yang baru dengan nama folder yang sama. Gambar 5.3 merupakan potongan *code* dimana sistem mengecek *file extension*.

```
$allowedExts=array('zip');
$errors=array();
$ufile=$_FILES['ufile'];
$file_name=$_FILES['ufile']['name'];
$fileext=pathinfo($file_name,PATHINFO_EXTENSION);
$basename= preg_replace('/.[^.]*$/','',$file_name);

if(in_array($fileext,$allowedExts)===TRUE){
  $dirname='plugins/'.$basename;
if(!file_exists($dirname)){
            mkdir($dirname,0777,true);
}else{
            $this->delete_directory($dirname);
            mkdir($dirname,0777,true);
}
```

Gambar 5.3 potongan *code* pengecekan *file extension*

Setelah itu sistem akan mengecek apakah di dalam file.zip berisikan file atau tidak. Jika terdapat file, sistem akan mengekstrak file ke direktori dan mengecek apakah file yang sudah diekstrak berisikan file .txt dengan nama yang sama dan file "index.php". Gambar 5.4 merupakan potongan *code* dimana sistem mengekstraksi file .zip dan mengecek isi di dalam file tersebut.

```
if($ufile!=null){

  if(copy($_FILES['ufile']['tmp_name'],$file_name))
  {
   $this->view->render('admin/addplugin');
$zip=new ZipArchive;
$res=$zip->open($file_name);
   if($res===TRUE){
    $zip->extractTo('plugins/'.$basename.'/');
$zip->close();
    unlink($file_name);
            if(file_exists('plugins/'.$basename.'/'.$basename.'.txt')){
            if(file_exists('plugins/'.$basename.'/index.php')){
```

Gambar 5.4 potongan *code* ekstrasi file .zip

Setelah itu sistem mengecek isi dari file.txt dimana isi dari file tersebut akan disimpan ke tabel plugin yang ada di database. Isi dari file tersebut merupakan keterangan dari plugin yang sedang diinstall antara lain nama plugin, versi plugin, deskripsi plugin, url plugin ketika disimpan di direktori, dan status plugin. Gambar 5.5 merupakan potongan code ketika sistem membaca isi file .txt dan menyimpan isi tersebut ke database.

```
$file=fopen('plugins/'.$basename.'/'.$basename.'.txt',"
r");
$plugin_name=array();
$i=0;
while(!feof($file)){
$lines= fgets($file);
$line=explode('\n',$lines);
$plugin_name[$i]=$line[0];
$i++;
}
fclose($file);
$nama=$plugin_name[0];
$versi=$plugin_name[1];
$deskripsi=$plugin_name[2];
$url=$plugin_name[3];
$status=$plugin_name[4];
$nama2=rtrim($nama);
$this->model->deleteExist($nama2);

//ambil dari entitas
require"entity/plugin_entity.php";
$plugin=new PluginEntity();
$plugin->set_NAMA($nama);
$plugin->set_VERSI($versi);
$plugin->set_DESKRIPSI($deskripsi);
$plugin->set_URLPATH($url);
$plugin->set_STATUS($status);
$this->model->createPlugin($plugin);
echo'<center>File berhasil diekstrak</center>';
```

Gambar 5.5 potongan *code* menyimpan *file*

Jika berhasil, maka akan muncul keterangan plugin pada menu “Plugin Manager” seperti pada gambar 5.6. Dan plugin akan disimpan di folder yang sesuai dengan nama plugin tersebut.

Gambar 5.6 halaman plugin manager

5.3.2. Pengaktifan Plugin

Untuk mengaktifkan plugin, Admin perlu mengklik tombol "Activate" yang ada pada kolom plugin di halaman "Plugin Manager". Saat Admin mengklik tombol "Activate", sistem mengecek nilai status plugin di database. Kemudian sistem akan mengubah status plugin di database dari 0 ke 1. Dan tombol "Activate" yang ada di halaman "Plugin Manager" akan berubah menjadi tombol "Deactivate". Hal itu menandakan plugin sedang aktif. Gambar 5.7 merupakan potongan code dimana sistem menjalankan fungsi mengaktifkan dan menonaktifkan plugin.

```
$data=$this->model->cekStatus($id);
if($data['STATUS']==0){
$this->model->activatePlugin($id);
$this->executePlugin($id);
$this->copyFiles($id);
header('location: '. URL1 .'pluginmanager');
}else{
$this->model->deactivatePlugin($id);
$this->deleteFiles($id);
header('location: '. URL1 .'pluginmanager');
```

Gambar 5.7 potongan *code* pengaktifan plugin

Ketika ingin menonaktifkan plugin, Admin perlu mengklik tombol "Deactivate" yang ada pada kolom plugin di halaman "Plugin Manager". Sistem akan melakukan hal yang sama seperti ketika admin mengaktifkan plugin, yang berbeda adalah sistem mengubah status plugin di database dari 1 ke 0. Gambar 5.8 merupakan potongan code dimana sistem mengubah nilai status plugin di database saat plugin diaktifkan dan dinonaktifkan.

```
function activatePlugin($id){
$sth=$this->db->prepare('UPDATE plugin SET STATUS= "1"
WHERE ID_PLUGIN = '.$id.'');
$sth->execute();
}

function deactivatePlugin($id){
$sth=$this->db->prepare('UPDATE plugin SET STATUS= "0"
WHERE ID_PLUGIN = '.$id.'');
$sth->execute();
}
```

Gambar 5.8 potongan *code* ubah nilai status plugin

Ketika Admin mengaktifkan plugin, sistem juga menjalankan fungsi executePlugin dimana sistem menjalankan file index.php yang ada di direktori plugin. Gambar 5.9 merupakan potongan code ketika sistem menjalankan plugin.

```
$status=$this->model->getPath($id);
$url=$status['URLPATH'];
$string= preg_replace('/\s+/','',$url);
require$string;
$index=new Index();
$eksekusi=$index->plugin();
return;
```

Gambar 5.9 potongan *code* menjalankan plugin

5.3.3. Penguninstallan Plugin

Untuk menguninstall atau menghapus plugin, Admin mengklik tombol “delete” yang ada pada menu “Plugin Manager”. Jika plugin masih aktif, sistem akan memunculkan pesan “Plugin masih aktif”. Jika plugin tidak aktif, sistem akan menjalankan fungsi deletePlugin seperti potongan code yang ada pada gambar 5.10.

```
$inputSplit=explode("-",$get_value);
$id=$inputSplit[0];
$nama=$inputSplit[1];
$nama=isset($inputSplit[1])?$inputSplit[1]:0;
$data=$this->model->cekStatus($id);
if($data['STATUS']==0){
$dirname='plugins/'.$nama;
$this->deleteTablePlugin($id);
$this->delete_directory($dirname);
$this->model->deletePlugin($id);
header('location: '. URL1 .'pluginmanager');
}else{
echo"<script>alert('Plugin masih aktif');</script>";
$this->view->data =$this->model->showPlugin();
$this->view->render('admin/pluginmanager');
}
```

Gambar 5.10 potongan *code* delete plugin

Ketika fungsi deletePlugin berjalan, sistem akan menghapus plugin dari direktori aplikasi. Kemudian jika plugin tersebut memerlukan adanya suatu tabel dalam pengeksekusiannya, maka sistem akan menghapus tabel yang dibuat dari plugin tersebut yang ada di database. Gambar 5.11 merupakan potongan code sistem menjalankan fungsi menghapus tabel yang dibuat plugin.

```
$status=$this->model->deletePath($id);
$url=$status['URLPATH'];
$string= preg_replace('/\s+/','',$url);
require$string;
$index=new Index();
$eksekusi=$index->delete();
return;
```

Gambar 5.11 potongan *code* hapus tabel

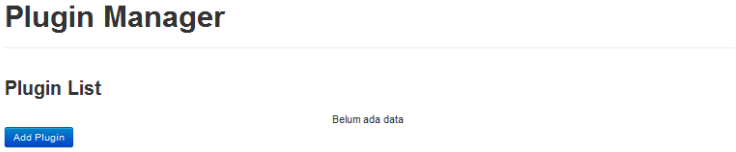
Selanjutnya sistem akan menghapus keterangan plugin pada tabel plugin yang ada di database. Gambar 5.12 merupakan

potongan code ketika sistem menghapus plugin dari tabel plugin.

```
function deletePlugin($id){
$sth=$this->db->prepare('DELETE FROM plugin WHERE
ID_PLUGIN = '.$id.'');
$sth->execute();
}
```

Gambar 5.12 potongan *code* hapus data dari tabel plugin

Ketika plugin selesai dihapus, maka keterangan plugin yang ada di menu "Plugin Manager" akan hilang seperti pada gambar 5.13.



Gambar 5.13 plugin manager saat plugin terhapus

5.3.4. Pengaturan Nilai Konfigurasi Plugin

Untuk mengatur konfigurasi plugin, admin harus mengklik tombol "Edit" yang ada pada menu "Plugin Manager". Sistem akan menampilkan halaman konfigurasi sesuai masing-masing plugin. Untuk plugin "Java Similarity Checker" ini, halaman konfigurasi yang ditampilkan dapat dilihat pada gambar 5.14.

Gambar 5.14 halaman konfigurasi plugin

Untuk memanggil tampilan pengaturan konfigurasi tiap plugin, sistem memanggil file "konfigurasi.php" yang ada pada direkroti plugin. Gambar 5.15 merupakan potongan code dimana sistem memanggil halaman konfigurasi tiap plugin.

```
$status=$this->model->getNamaPlugin($id);
$this->view->id =$id;
$nama=$status['NAMA'];
$string= preg_replace('/\s+/','',$nama);
$this->view->pluginrender($string.'/konfigurasi');
```

Gambar 5.15 potongan *code* halaman konfigurasi plugin

Ketika admin mengklik tombol untuk menyimpan konfigurasi plugin, sistem memanggil file "save.php" yang ada di plugin. File "save.php" berguna untuk menyimpan nilai konfigurasi yang telah diubah. Gambar 5.16 merupakan potongan code untuk menyimpan nilai konfigurasi yang telah diubah di halaman konfigurasi plugin.

```
if(isset($_POST['komenn'])&&isset($_POST['spasi'])&&isset($_POST['batas'
])){
$data=$_POST['komenn']."\r\n".$_POST['spasi']."\r\n".$_POST['batas']."\r\n
";
//echo $data;
unlink('plugins/JavaSimilarityChecker/konfigurasi.txt');
fopen('plugins/JavaSimilarityChecker/konfigurasi.txt',"w");
$ret=
file_put_contents('plugins/JavaSimilarityChecker/konfigurasi.txt',$data,
```

Gambar 5.16 potongan *code* simpan nilai konfigurasi

5.3.5. Pengimplementasian Tes Kemiripan

Dalam menjalankan fungsi melakukan tes kemiripan, sistem secara otomatis mengecek daftar jawaban yang ada di direktori. Sebelum melakukan tes kemiripan, sistem terlebih dahulu mengecek ketersediaan jawaban yang ada di direktori jawaban. Kemudian sistem menambahkan data jawaban ke dalam tabel jawaban yang ada di database. Setelah menyimpan data jawaban, sistem memulai menjalankan uji kemiripan berdasarkan id pertanyaan pada jawaban. Gambar 5.17 merupakan potongan code yang menjelaskan cara sistem melakukan tes kemiripan.

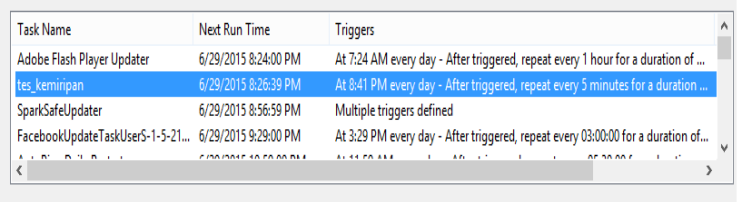
Untuk menjalankan fungsi sistem secara otomatis, diperlukan adanya scheduling atau penjadwalan agar program atau sistem yang dibuat dapat berjalan secara teratur atau sesuai dengan waktu yang ditentukan. Untuk fungsi melakukan uji kemiripan, sistem plugin diatur dapat berjalan secara otomatis setiap 5 menit.

```
foreach(new Combinations($jawaban,2)as$substring){
$split1=explode("/",$substring[0]);
$split2=explode("/",$substring[1]);
$iduser1=$split1[6];
$idpertanyaan1=$split1[7];
$iduser2=$split2[6];
$file1= file_get_contents($substring[0]);
$file2= file_get_contents($substring[1]);
$lev=$testing->levensh($file1,$file2);
$rab=$testing->rabin($file1,$file2);
$dupsql="SELECT * FROM report_test where (ID_PENGGUNA1 = $iduser1
AND ID_PENGGUNA2 = $iduser2 AND ID_PERTANYAAN =
$idpertanyaan1)";
$dupraw=mysql_query($dupsql)ordie(mysql_error());
if(mysql_num_rows($dupraw)>0){
mysql_query("DELETE FROM report_test WHERE (ID_PENGGUNA1 =
$iduser1 AND ID_PENGGUNA2 = $iduser2 AND ID_PERTANYAAN =
$idpertanyaan1)")ordie(mysql_error());
}

mysql_query("INSERT INTO report_test (ID_PENGGUNA1, ID_PENGGUNA2,
ID_PERTANYAAN, LEVENSHTEIN, RABINKARP) VALUES ($iduser1,
$iduser2, $idpertanyaan1, $lev, $rab)")ordie(mysql_error());
}
```

Gambar 5.17 Potongan *Code* Melakukan Tes Kemiripan

Untuk proses scheduling pada sistem plugin sendiri, digunakan fitur task scheduler pada windows untuk mengatur fungsi plugin agar dapat berjalan sesuai waktu yang ditentukan. Gambar 5.18 merupakan task yang sudah diatur pada task scheduler untuk menjalankan uji kemiripan secara otomatis dan terjadwal.



Gambar 5.18 task yang sudah diatur pada task scheduler

5.3.6. Pengimplementasian Tes Ulang

Untuk menjalankan fungsi tes ulang yang ada pada desain sistem, teacher mengklik tombol tes ulang yang ada pada halaman "report test". Kemudian sistem akan menampilkan halaman konfigurasi untuk tes ulang. Gambar 5.19 merupakan potongan code untuk memanggil halaman konfigurasi untuk tes ulang.

```
function tesUlang($id){
$this->view->list=$this->model->quizList();
$this->view->id =$id;
$this->view->render('JavaSimilarityChecker/tesulang');
}
```

Gambar 5.19 potongan *code* halaman konfigurasi tes ulang

Setelah berada pada halaman konfigurasi, teacher mengatur nilai konfigurasi yang ingin dipilih. Sistem akan membaca konfigurasi yang telah diatur oleh teacher dan mulai melakukan tes ulang dengan konfigurasi yang sudah ditentukan oleh teacher. Setelah melakukan tes ulang, sistem akan menampilkan halaman "Report Tes Ulang" yang berisikan hasil tes ulang berdasarkan quiz yang telah dipilih. Gambar 5.20 merupakan potongan code untuk menampilkan halaman report tes ulang berdasarkan quiz yang telah dipilih.

```
include"plugins/JavaSimilarityChecker/save.php";
$this->view->data =$this->model->reportQuizList($idnya);
$this->view->render('JavaSimilarityChecker/reportTesUlang');
```

Gambar 5.20 Potongan code menampilkan report tes

5.4. Uji Coba

Bagian ini membahas tentang uji coba yang dilakukan terhadap sistem. Adapun uji coba yang dilakukan adalah uji coba fungsional dan uji coba validitas yang digunakan untuk mengetahui keberhasil fungsi pada plugin dan kebenaran algoritma yang dipakai. Uji coba akurasi juga diperlukan untuk mengetahui kemampuan algoritma dalam mendeteksi kemiripan source code.

5.4.1. Uji Coba Fungsional

Uji coba fungsional merupakan uji coba yang dilakukan untuk mengetahui apakah fitur-fitur yang ada pada sistem sudah berjalan lancar. Uji coba fungsional yang dilakukan pada sistem ini berdasarkan pada format test case yang sudah dijelaskan pada tahap desain sistem. Tabel 5.3 merupakan daftar uji coba fungsional yang dilakukan. Untuk detail test case yang dilakukan dapat dilihat pada Lampiran F.

Tabel 5.3 daftar uji coba fungsional

Kode	Test Case
TC-01	Login Admin
TC-02	Logout Admin
TC-03	Install Plugin
TC-04	Melihat Daftar Plugin
TC-05	Uninstall Plugin
TC-06	Mengaktifkan Plugin
TC-07	Menon-aktifkan Plugin
TC-08	Mengatur Nilai Konfigurasi Sistem
TC-09	Melakukan Tes Kemiripan
TC-10	Melihat Report Tes per Quiz
TC-11	Melakukan Tes Ulang

5.4.2. Uji Coba Validitas

Uji coba algoritma digunakan untuk menguji kebenaran cara penghitungan kemiripan oleh algoritma yang dipakai. Dalam plugin “Java Similarity Checker” digunakan dua algoritma untuk mengecek kemiripan source code yaitu Levenshtein Distance dan Rabin Karp.

Uji coba yang dilakukan adalah mencocokkan kesamaan hasil yang didapat dari perhitungan yang dilakukan oleh algoritma di plugin dan algoritma yang ada di aplikasi pendeteksi kemiripan yang sudah ada.

Untuk algoritma Levenshtein Distance, digunakan algoritma dari aplikasi Online Calculator (http://planetcalc.com/1721/)

untuk menentukan ketepatan algoritma Levenshtein Distance pada plugin. Sedangkan untuk algoritma Rabin Karp, digunakan aplikasi buatan dari Arfian Hidayat (http://program.arfianhidayat.com/rabinkarp.html) untuk menentukan ketepatan algoritma Rabin Karp yang ada pada plugin.

5.4.3. Uji Coba Akurasi

Uji coba akurasi merupakan uji coba yang dilakukan untuk mengetahui seberapa akurat algoritma yang dipakai dalam melakukan uji kemiripan. Dalam uji coba ini digunakan dua perhitungan yang banyak dipakai untuk mengukur kinerja pada sistem yaitu recall dan precision. Precision adalah tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem. Sedangkan recall adalah tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi. Dalam menentukan nilai recall dan precision dalam agoritma yang dipakai dalam sistem ini, terlebih dahulu ditentukan kriteria pada source code. Terdapat beberapa kriteria yang ditentukan dalam menentukan keakuratan algoritma dalam uji coba ini, yaitu:

- True Positive, apabila dua source code dilihat secara kasat mata adalah mirip dan persentase kemiripan dua source code tersebut melebihi batas yang ditentukan pada konfigurasi plugin yaitu 80%.
- True Negative, apabila dua source code dilihat secara kasat mata adalah tidak mirip dan persentase kemiripan dua source code tersebut tidak melebihi batas yang ditentukan pada konfigurasi plugin yaitu 80%.
- False Positive, apabila dua source code dilihat secara kasat mata adalah tidak mirip dan persentase kemiripan dua source code tersebut melebihi batas yang ditentukan pada konfigurasi plugin yaitu 80%.
- False Negative, apabila dua source code dilihat secara kasat mata adalah mirip dan persentase kemiripan dua

source code tersebut tidak melebihi batas yang ditentukan pada konfigurasi plugin yaitu 80%.

Dari kriteria yang telah dibuat, dibuat rumus perhitungan recall dan precision dari sistem berdasarkan tabel 5.4 yang dapat dilihat pada gambar 5.21.

Tabel 5.4 penentuan kriteria

		Source code mirip	
		Mirip	Tidak Mirip
Source code melebihi batas	Melebihi	TP (True Positive)	FP (False Positive)
	Tidak Melebihi	FN (False Negative)	TN (True Negative)

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

Gambar 5.21 rumus *precision* dan *recall*

Uji coba yang dilakukan ada dua tipe yaitu uji coba dengan menentukan kriteria modifikasi dan uji coba tanpa menentukan kriteria modifikasi.

1. **Uji coba dengan menentukan kriteria modifikasi**

   Uji coba dilakukan pada kuis "Sort Test" yang ada dalam materi "Sorting" yang ada pada aplikasi "Belajar Java". Dalam kuis "Sort Test", mahasiswa diwajibkan untuk mengurutkan angka dalam array dari yang terbesar ke yang terkecil.

   Pada tabel 5.5 uji coba dilakukan menggunakan delapan akun mahasiswa yang berbeda dimana akun pertama yaitu User1 mengerjakan kuis tanpa melakukan kemiripan dari pengerjaan mahasiswa lain. Untuk contoh pengerjaan kuis yang dilakukan oleh User1 dapat dilihat pada gambar 5.22.

Untuk akun kedua sampai dengan kedelapan, pengerjaan kuis dilakukan dengan melakukan perubahan atau modifikasi dari pengerjaan kuis oleh akun pertama. Bagian yang dimodifikasi adalah isi array, komentar, dan variabel. Modifikasi yang yang dilakukan antara lain mengubah isi array, menambah komentar, mengubah variabel, menambah komentar dan mengubah isi array, mengubah isi array dan variabel, menambah komentar dan mengubah variabel, serta modifikasi ketiga bagian. Untuk perbandingan source code antar user dapat dilihat pada Lampiran G.

Tabel 5.5 uji coba skenario 1

Modifikasi / Akun	Mengubah Isi Array	Menambah Comment	Mengubah Variabel
User 1	-	-	-
User 2	✓	-	-
User 3	-	✓	-
User 4	-	-	✓
User 5	✓	✓	-
User 6	✓	-	✓
User 7	-	✓	✓
User 8	✓	✓	✓

```
public class Belajar {
        public static void main(String args[]){
        int a[] = {9, 20, 15, 18, 1, 3, 2, 5, 6, 8, 10, 11, 14, 13, 4, 7, 12, 17, 19, 16};
        for ( int kiri = 0; kiri < a.length - 1; kiri++ ){
                for ( int kanan = kiri + 1; kanan < a.length; kanan++ ){
                        if ( a[ kanan ] > a[ kiri ] ){
                                int temporary = a[ kiri ];
                                a[ kiri ] = a[ kanan ];
                                a[ kanan ] = temporary;
                        }
                }
        }
        System.out.println(a[0]+" "+a[1]+" "+a[2]+" "+a[3]+" "+a[4]+" "+a[5]+" "+a[6]+"
"+a[7]+" "+a[8]+" "+a[9]+" "+a[10]+" "+a[11]+" "+a[12]+" "+a[13]+" "+a[14]+" "+a[15]+"
"+a[16]+" "+a[17]+" "+a[18]+" "+a[19]);
        }
}
```

Gambar 5.22 contoh pengerjaan kuis oleh User1

Dari beberapa modifikasi yang sudah dijelaskan, dilakukan beberapa skenario untuk mengetahui persentase

tingkat kemiripan antara User1 dan User2 sampai dengan User8. Skenario yang dilakukan antara lain:

- Uji kemiripan User 1 dengan User 2
  Mencocokkan kemiripan antara source code yang dikerjakan oleh User 1 dan source code yang dikerjakan oleh User 2 dengan melakukan modifikasi pada source code User 1 yaitu mengubah isi array. Untuk contoh pengerjaan kuis yang dilakukan oleh User2 dapat dilihat pada gambar 5.23.

```
public class Belajar {
    public static void main(String args[]){

     int a[] = {9, 2, 11, 1, 10, 3, 20, 5, 6, 8, 18, 15, 4, 13, 14, 7, 12, 17, 19, 16};


    for ( int kiri = 0; kiri < a.length - 1; kiri++ ){

        for ( int kanan = kiri + 1; kanan < a.length; kanan++ ){
            if ( a[ kanan ] > a[ kiri ] ){

                int temporary = a[ kiri ];
                a[ kiri ] = a[ kanan ];
                a[ kanan ] = temporary;

            }
        }
    }

    System.out.println(a[0]+" "+a[1]+" "+a[2]+" "+a[3]+" "+a[4]+" "+a[5]+" "+a[6]+" "+a
[7]+" "+a[8]+" "+a[9]+" "+a[10]+" "+a[11]+" "+a[12]+" "+a[13]+" "+a[14]+" "+a[15]+" "+a
[16]+" "+a[17]+" "+a[18]+" "+a[19]);

    }
}
```

Gambar 5.23 contoh pengerjaan kuis oleh user2

- Uji kemiripan User 1 dengan User 3
  Mencocokkan kemiripan antara source code yang dikerjakan oleh User 1 dan source code yang dikerjakan oleh User 3 dengan melakukan modifikasi pada source code User 1 yaitu menambah comment. Untuk contoh pengerjaan kuis yang dilakukan oleh User3 dapat dilihat pada gambar 5.24.

```
public class Belajar {
    public static void main(String args[]){

    int a[] = {9, 20, 15, 18, 1, 3, 2, 5, 6, 8, 10, 11, 14, 13, 4, 7, 12, 17, 19, 16};

    //loop index kiri yang dibandingkan
    for ( int kiri = 0; kiri < a.length - 1; kiri++ ){
        //loop index kanan sebagai pembanding
        for ( int kanan = kiri + 1; kanan < a.length; kanan++ ){
            if ( a[ kanan ] > a[ kiri ] ){
                //swap nilai index kiri dengan index kanan
                int temporary = a[ kiri ]; //simpan nilai index kiri sementara di temporary
                a[ kiri ] = a[ kanan ]; //ubah nilai index kiri dengan index kanan
                a[ kanan ] = temporary; //ubah nilai index kanan dengan temporary
            }
        }
    }

    System.out.println(a[0]+" "+a[1]+" "+a[2]+" "+a[3]+" "+a[4]+" "+a[5]+" "+a[6]+" "+a
[7]+" "+a[8]+" "+a[9]+" "+a[10]+" "+a[11]+" "+a[12]+" "+a[13]+" "+a[14]+" "+a[15]+" "+a
[16]+" "+a[17]+" "+a[18]+" "+a[19]);

    }
}
```

Gambar 5.24 contoh pengerjaan kuis oleh User3

- Uji kemiripan User 1 dengan User 4
  Mencocokkan kemiripan antara source code yang dikerjakan oleh User 1 dan source code yang dikerjakan oleh User 4 dengan melakukan modifikasi pada source code User 1 yaitu mengubah nama variabel. Untuk contoh pengerjaan kuis yang dilakukan oleh User4 dapat dilihat pada gambar 5.25.

```
public class Belajar {
    public static void main(String args[]){

    int a[] = {9, 20, 15, 18, 1, 3, 2, 5, 6, 8, 10, 11, 14, 13, 4, 7, 12, 17, 19, 16};

    for ( int indexKiri = 0; indexKiri < a.length - 1; indexKiri++ ){
        for ( int indexKanan = indexKiri + 1; indexKanan < a.length; indexKanan++ ){
            if ( a[ indexKanan ] > a[ indexKiri ] ){
                int temporary = a[ indexKiri ];
                a[ indexKiri ] = a[ indexKanan ];
                a[ indexKanan ] = temporary;
            }
        }
    }

    System.out.println(a[0]+" "+a[1]+" "+a[2]+" "+a[3]+" "+a[4]+" "+a[5]+" "+a[6]+" "+a
[7]+" "+a[8]+" "+a[9]+" "+a[10]+" "+a[11]+" "+a[12]+" "+a[13]+" "+a[14]+" "+a[15]+" "+a
[16]+" "+a[17]+" "+a[18]+" "+a[19]);

    }
}
```

Gambar 5.25 contoh pengerjaan kuis oleh user4

- Uji kemiripan User 1 dengan User 5
  Mencocokkan kemiripan antara source code yang dikerjakan oleh User 1 dan source code yang dikerjakan oleh User 5 dengan melakukan modifikasi pada source code User 1 yaitu mengubah isi array dan menambah comment. Untuk contoh pengerjaan kuis yang dilakukan oleh User5 dapat dilihat pada gambar 5.26.

```
public class Belajar {
    public static void main(String args[]){

    int a[] = {9, 2, 11, 1, 10, 3, 20, 5, 6, 8, 18, 15, 4, 13, 14, 7, 12, 17, 19, 16};

    //loop index kiri yang dibandingkan
    for ( int kiri = 0; kiri < a.length - 1; kiri++ ){
        //loop index kanan sebagai pembanding
        for ( int kanan = kiri + 1; kanan < a.length; kanan++ ){
            if ( a[ kanan ] > a[ kiri ] ){
                //swap nilai index kiri dengan index kanan
                int temporary = a[ kiri ]; //simpan nilai index kiri sementara di temporary
                a[ kiri ] = a[ kanan ]; //ubah nilai index kiri dengan index kanan
                a[ kanan ] = temporary; //ubah nilai index kanan dengan temporary
            }
        }
    }

    System.out.println(a[0]+" "+a[1]+" "+a[2]+" "+a[3]+" "+a[4]+" "+a[5]+" "+a[6]+" "+a
[7]+" "+a[8]+" "+a[9]+" "+a[10]+" "+a[11]+" "+a[12]+" "+a[13]+" "+a[14]+" "+a[15]+" "+a
[16]+" "+a[17]+" "+a[18]+" "+a[19]);

    }
}
```

Gambar 5.26 contoh pengerjaan oleh user5

- Uji kemiripan User 1 dengan User 6
  Mencocokkan kemiripan antara source code yang dikerjakan oleh User 1 dan source code yang dikerjakan oleh User 6 dengan melakukan modifikasi pada source code User 1 yaitu mengubah isi array dan mengubah nama variabel. Untuk contoh pengerjaan kuis yang dilakukan oleh User6 dapat dilihat pada gambar 5.27.

```
public class Belajar {
    public static void main(String args[]){

    int a[] = {9, 2, 11, 1, 10, 3, 20, 5, 6, 8, 18, 15, 4, 13, 14, 7, 12, 17, 19, 16};

    for ( int indexKiri = 0; indexKiri < a.length - 1; indexKiri++ ){
        for ( int indexKanan = indexKiri + 1; indexKanan < a.length; indexKanan++ ){
            if ( a[ indexKanan ] > a[ indexKiri ] ){
                int temporary = a[ indexKiri ];
                a[ indexKiri ] = a[ indexKanan ];
                a[ indexKanan ] = temporary;
            }
        }
    }

    System.out.println(a[0]+" "+a[1]+" "+a[2]+" "+a[3]+" "+a[4]+" "+a[5]+" "+a[6]+" "+a
[7]+" "+a[8]+" "+a[9]+" "+a[10]+" "+a[11]+" "+a[12]+" "+a[13]+" "+a[14]+" "+a[15]+" "+a
[16]+" "+a[17]+" "+a[18]+" "+a[19]);

    }
}
```

Gambar 5.27 contoh pengerjaan kuis oleh user6

- Uji kemiripan User 1 dengan User 7
  Mencocokkan kemiripan antara source code yang dikerjakan oleh User 1 dan source code yang dikerjakan oleh User 7 dengan melakukan modifikasi pada source code User 1 yaitu menambah comment dan mengubah nama variabel. Untuk contoh pengerjaan kuis yang dilakukan oleh User7 dapat dilihat pada gambar 5.28.

```
public class Belajar {
    public static void main(String args[]){

    int a[] = {9, 20, 15, 18, 1, 3, 2, 5, 6, 8, 10, 11, 14, 13, 4, 7, 12, 17, 19, 16};

    //loop index kiri yang dibandingkan
    for ( int indexKiri = 0; indexKiri < a.length - 1; indexKiri++ ){
        //loop index kanan sebagai pembanding
        for ( int indexKanan = indexKiri + 1; indexKanan < a.length; indexKanan++ ){
            if ( a[ indexKanan ] > a[ indexKiri ] ){
                //swap nilai index kiri dengan index kanan
                int temporary = a[ indexKiri ]; //simpan nilai index kiri sementara di temporary
                a[ indexKiri ] = a[ indexKanan ]; //ubah nilai index kiri dengan index kanan
                a[ indexKanan ] = temporary; //ubah nilai index kanan dengan temporary
            }
        }
    }

    System.out.println(a[0]+" "+a[1]+" "+a[2]+" "+a[3]+" "+a[4]+" "+a[5]+" "+a[6]+" "+a[7]+" "+a
[8]+" "+a[9]+" "+a[10]+" "+a[11]+" "+a[12]+" "+a[13]+" "+a[14]+" "+a[15]+" "+a[16]+" "+a[17]+"
"+a[18]+" "+a[19]);

    }
}
```

Gambar 5.28 contoh pengerjaan kuis oleh user7

- Uji kemiripan User 1 dengan User 8
  Mencocokkan kemiripan antara source code yang dikerjakan oleh User 1 dan source code yang dikerjakan oleh User 8 dengan melakukan modifikasi pada source code User 1 yaitu mengubah isi array, menambah comment, dan mengubah nama variabel. Untuk contoh pengerjaan kuis yang dilakukan oleh User8 dapat dilihat pada gambar 5.29.

```
public class Belajar {
    public static void main(String args[]){

    int a[] = {9, 2, 11, 1, 10, 3, 20, 5, 6, 8, 18, 15, 4, 13, 14, 7, 12, 17, 19, 16};

    //loop index kiri yang dibandingkan
    for ( int indexKiri = 0; indexKiri < a.length - 1; indexKiri++ ){
        //loop index kanan sebagai pembanding
        for ( int indexKanan = indexKiri + 1; indexKanan < a.length; indexKanan++ ){
            if ( a[ indexKanan ] > a[ indexKiri ] ){
                //swap nilai index kiri dengan index kanan
                int temporary = a[ indexKiri ]; //simpan nilai index kiri sementara di temporary
                a[ indexKiri ] = a[ indexKanan ]; //ubah nilai index kiri dengan index kanan
                a[ indexKanan ] = temporary; //ubah nilai index kanan dengan temporary
            }
        }
    }

    System.out.println(a[0]+" "+a[1]+" "+a[2]+" "+a[3]+" "+a[4]+" "+a[5]+" "+a[6]+" "+a[7]+" "+a
[8]+" "+a[9]+" "+a[10]+" "+a[11]+" "+a[12]+" "+a[13]+" "+a[14]+" "+a[15]+" "+a[16]+" "+a[17]+"
"+a[18]+" "+a[19]);

    }
}
```

Gambar 5.29 contoh pengerjaan oleh user8

2. Uji coba tanpa menentukan kriteria modifikasi

Uji coba dilakukan pada kuis “Searching Test” yang ada dalam materi “Search” yang ada pada aplikasi “Belajar Java”. Uji coba dilakukan pada kondisi nyata yaitu dengan menggunakan jawaban dari 10 sampel mahasiswa.Setiap jawaban yang telah dikerjakan oleh kesepuluh mahasiswa akan saling diuji coba untuk diketahui persentase kemiripannya satu dengan yang lain.

BAB VI
HASIL DAN PEMBAHASAN

Pada bab ini akan dijelaskan hasil dari setiap uji coba yang telah dijelaskan pada bab sebelumnya dan membahas hasil uji coba yang telah didapat.

6.1. Hasil Uji Coba

Bagian ini membahas tentang uji coba yang dilakukan terhadap sistem. Hasil uji coba meliputi hasil uji coba fungsional yang ada pada test case, hasil uji coba validitas algoritma yang digunakan, dan hasil uji coba akurasi berupa hasil keakuratan dari algoritma yang digunakan pada uji kemiripan.

6.1.1. Hasil Uji Coba Fungsional

Seperti yang telah dijelaskan pada bab sebelumnya, uji coba fungsional dilakukan untuk memastikan fitur-fitur pada sistem berjalan dengan baik. Dari uji coba yang telah dilakukan, didapatkan hasil dari setiap test case yang dapat dilihat pada tabel 6.1. Untuk detail hasil coba dari setiap test case dapat dilihat pada Lampiran F.

Tabel 6.1 hasil seluruh test case

Kode	Test Case	Status
TC-01	Login Admin	Terpenuhi
TC-02	Logout Admin	Terpenuhi
TC-03	Install Plugin	Terpenuhi
TC-04	Melihat Daftar Plugin	Terpenuhi
TC-05	Uninstall Plugin	Terpenuhi
TC-06	Mengaktifkan Plugin	Terpenuhi
TC-07	Menon-aktifkan Plugin	Terpenuhi
TC-08	Mengatur Nilai Konfigurasi Sistem	Terpenuhi
TC-09	Melakukan Tes Kemiripan	Terpenuhi
TC-10	Melihat Report Tes per Quiz	Terpenuhi
TC-11	Melakukan Tes Ulang	Terpenuhi

6.1.2. Hasil Uji Coba Validitas

Seperti yang telah dijelaskan pada subbab sebelumnya, uji coba dilakukan untuk menguji ketepatan cara penghitungan kemiripan oleh algoritma yang dipakai plugin. Uji coba yang dilakukan adalah mencocokkan kesamaan hasil yang didapat dari perhitungan yang dilakukan oleh algoritma di plugin dan algoritma yang ada di aplikasi pendeteksi kemiripan yang sudah ada.

Dari uji coba yang dilakukan, didapatkan beberapa hasil dari pencocokkan dua algoritma pada plugin dengan pengimplementasian algoritma pada aplikasi lain.Tabel 6.2 berisikan daftar-daftar kalimat yang dibuat untuk uji coba. Hasil uji coba dapat dilihat pada Tabel 6.3

Tabel 6.2 daftar kalimat untuk uji coba

No.	Kode	Kalimat 1	Kalimat 2
1	S-1	int a=0;	int b=5;
2	S-2	Puasa Ramadhan	Bulan ramadhan
3	S-3	Halo-halo bandung	Halo apa kabar?
4	S-4	for(x=0;x<5;x++){	for(y=0;y<letak;y++){
5	S-5	Ini bapak budi	Ini ibu budi

Tabel 6.3 hasil uji coba algoritma

No.	Kode	Levenshtein Distance		Rabin Karp		Status
		plugin	aplikasi lain	plugin	aplikasi lain	
1	S-1	75%	75%	40%	40%	Cocok
2	S-2	64.29%	64.29%	58.33%	58.33%	Cocok
3	S-3	41.18%	41.18%	25%	25%	Cocok
4	S-4	61.90%	61.90%	25%	25%	Cocok
5	S-5	64.29%	64.29%	45.45%	45.45%	Cocok

6.1.3. Hasil Uji Coba Akurasi

Seperti yang telah dijelaskan pada subbab sebelumnya, uji coba dilakukanuntuk mengetahui seberapa akurat algoritma yang dipakai dalam melakukan uji kemiripan. Uji coba yang dilakukan ada dua tipe yaitu uji coba dengan menentukan kriteria modifikasi dan uji coba tanpa menentukan kriteria modifikasi.

1. **Hasil Skenario 1**

   Dari uji coba yang dilakukan pada skenario 1, didapatkan hasil uji coba berupa persentase kemiripan dari source code yang diuji melalui algoritma Levenshtein Distance dan Rabin-Karp. Hasil uji kemiripan dapat dilihat pada tabel 6.4.

Tabel 6.4 hasil uji coba kemiripan skenario 1

No.	Source Code yang Diuji	Levenshtein Distance	Rabin-Karp
1	User 1 – User 2	98.07%	94.12%
2	User 1 – User 3	68.53%	90.27%
3	User 1 – User 4	85.31%	95.15%
4	User 1 – User 5	67.21%	84.78%
5	User 1 – User 6	83.62%	89.18%
6	User 1 – User 7	60.81%	87.24%
7	User 1 – User 8	59.60%	81.68%

Keterangan status kemiripan source code dari percobaan skenario 1 dapat dilihat pada tabel 6.5. Tabel 6.6 merupakan tabel yang berisi jumlah kriteria source yang berhasil dideteksi oleh algoritma Levenshtein Distance dan Rabin Karp.

Tabel 6.5 status kemiripan source code pada skenario 1

No.	Source Code yang Diuji	Mirip/Tidak Mirip
1	User 1 – User 2	Mirip
2	User 1 – User 3	Mirip
3	User 1 – User 4	Mirip

No.	Source Code yang Diuji	Mirip/Tidak Mirip
4	User 1 – User 5	Mirip
5	User 1 – User 6	Mirip
6	User 1 – User 7	Mirip
7	User 1 – User 8	Mirip

Tabel 6.6 kriteria source code skenario 1

kriteria source code	Levenshtein Distance	Rabin Karp
TP	4	7
FP	0	0
TN	0	0
FN	3	0

Dari kriteria source yang ada pada tabel 6.6, ditentukan nilai recall dan precision dari algoritma Levenshtein Distance dan Rabin Karp. Hasil perhitungan recall dan precision pada skenario1 dapat dilihat pada tabel 6.7.

Tabel 6.7 nilai recall dan precision skenario1

Jenis Perhitungan	Levenshtein Distance	Rabin Karp
Recall	0.58	1
Precison	1	1

2. Hasil Skenario 2

Dari uji coba yang dilakukan pada skenario 2, didapatkan hasil uji coba berupa persentase kemiripan dari source code yang diuji melalui algoritma Levenshtein Distance dan Rabin-Karp. Hasil uji kemiripan dapat dilihat pada tabel 6.8.

Tabel 6.8 hasil uji kemiripan skenario 2

No.	Mahasiswa 1	Mahasiswa 2	Levenshtein Distance	Rabin-Karp
1	UserA	UserB	66.18%	83.83%
2	UserA	UserC	61.09%	80.46%
3	UserA	UserD	70.31%	84.85%
4	UserB	UserC	59.47%	79.55%
5	UserB	UserD	74.59%	87.50%
6	UserC	UserD	65.85%	83.33%
7	UserA	UserE	67.84%	83.83%
8	UserA	UserF	68.88%	84.34%
9	UserB	UserE	50.75%	73.30%
10	UserB	UserF	84.16%	86.29%
11	UserC	UserE	92.65%	91.49%
12	UserC	UserF	79.74%	85.43%
13	UserD	UserE	85.29%	90.71%
14	UserD	UserF	74.15%	82.72%
15	UserE	UserF	83.33%	88.42%
16	UserA	UserG	84.80%	87.83%
17	UserA	UserH	69.17%	80.37%
18	UserA	UserJ	85.07%	90.26%
19	UserA	UserI	80.62%	83.50%
20	UserB	UserG	78.28%	86.32%
21	UserB	UserH	69.23%	78.79%
22	UserB	UserJ	82.81%	86.29%
23	UserB	UserI	81.00%	88.78%
24	UserC	UserG	71.43%	81.45%
25	UserC	UserH	82.38%	87.31%
26	UserC	UserJ	89.58%	91.71%
27	UserC	UserI	76.59%	82.54%
28	UserD	UserG	87.94%	92.55%
29	UserD	UserH	87.76%	90.91%
30	UserD	UserJ	69.17%	80.19%
31	UserD	UserI	75.77%	85.42%

No.	Mahasiswa 1	Mahasiswa 2	Levenshtein Distance	Rabin-Karp
32	UserE	UserG	66.08%	83.00%
33	UserE	UserH	80.18%	85.43%
34	UserE	UserJ	80.18%	88.89%
35	UserE	UserI	75.19%	83.41%
36	UserF	UserG	85.85%	88.04%
37	UserF	UserH	79.40%	88.52%
38	UserF	UserJ	91.33%	93.41%
39	UserF	UserI	66.92%	80.19%
40	UserG	UserH	71.22%	83.77%
41	UserG	UserJ	80.98%	85.26%
42	UserG	UserI	65.04%	76.28%
43	UserH	UserJ	77.39%	87.83%
44	UserH	UserI	69.17%	78.50%
45	UserJ	UserI	66.54%	83.57%

Keterangan status kemiripan source code dari percobaan skenario 2 dapat dilihat pada tabel 6.9. Tabel 6.10 merupakan tabel yang berisi jumlah kriteria source yang berhasil dideteksi oleh algoritma Levenshtein Distance dan Rabin Karp.

Tabel 6.9 status kemiripan source code pada skenario 2

No.	Mahasiswa 1	Mahasiswa 2	Mirip / Tidak Mirip
1	UserA	UserB	Tidak Mirip
2	UserA	UserC	Tidak Mirip
3	UserA	UserD	Tidak Mirip
4	UserB	UserC	Tidak Mirip
5	UserB	UserD	Mirip
6	UserC	UserD	Tidak Mirip
7	UserA	UserE	Tidak Mirip
8	UserA	UserF	Tidak Mirip
9	UserB	UserE	Mirip

No.	Mahasiswa 1	Mahasiswa 2	Mirip / Tidak Mirip
10	UserB	UserF	Tidak Mirip
11	UserC	UserE	Tidak Mirip
12	UserC	UserF	Tidak Mirip
13	UserD	UserE	Mirip
14	UserD	UserF	Tidak Mirip
15	UserE	UserF	Tidak Mirip
16	UserA	UserG	Tidak Mirip
17	UserA	UserH	Tidak Mirip
18	UserA	UserJ	Tidak Mirip
19	UserA	UserI	Tidak Mirip
20	UserB	UserG	Tidak Mirip
21	UserB	UserH	Tidak Mirip
22	UserB	UserJ	Tidak Mirip
23	UserB	UserI	Tidak Mirip
24	UserC	UserG	Tidak Mirip
25	UserC	UserH	Mirip
26	UserC	UserJ	Mirip
27	UserC	UserI	Tidak Mirip
28	UserD	UserG	Tidak Mirip
29	UserD	UserH	Tidak Mirip
30	UserD	UserJ	Tidak Mirip
31	UserD	UserI	Tidak Mirip
32	UserE	UserG	Tidak Mirip
33	UserE	UserH	Tidak Mirip
34	UserE	UserJ	Tidak Mirip
35	UserE	UserI	Tidak Mirip
36	UserF	UserG	Mirip
37	UserF	UserH	Mirip
38	UserF	UserJ	Tidak Mirip
39	UserF	UserI	Tidak Mirip
40	UserG	UserH	Tidak Mirip
41	UserG	UserJ	Tidak Mirip

No.	Mahasiswa 1	Mahasiswa 2	Mirip / Tidak Mirip
42	UserG	UserI	Tidak Mirip
43	UserH	UserJ	Tidak Mirip
44	UserH	UserI	Tidak Mirip
45	UserJ	UserI	Tidak Mirip

Tabel 6.10 kriteria *source code* skenario2

kriteria source code	Levenshtein Distance	Rabin Karp
TP	4	6
FP	14	34
TN	25	5
FN	2	0

Dari kriteria source yang ada pada tabel 6.10, ditentukan nilai recall dan precision dari algoritma Levenshtein Distance dan Rabin Karp. Hasil perhitungan recall dan precision pada skenario1 dapat dilihat pada tabel 6.11.

Tabel 6.11 nilai recall dan precision skenario2

Jenis Perhitungan	Levenshtein Distance	Rabin Karp
Recall	0.67	1
Precison	0.22	0.15

6.2. Pembahasan

Pada subbab ini diuraikan mengenai pembahasan hasil dari setiap uji coba yang telah dijelaskan pada subbab sebelumnya. Pembahasan yang dicantumkan antara lain pembahasan hasil uji coba fungsional, hasil uji coba algoritma, dan hasil uji coba akurasi.

6.2.1. Pembahasan Hasil Uji Coba Fungsional

Berdasarkan hasil uji coba fungsional yang ada pada tabel 6.1, dapat diketahui bahwa semua test case yang dilakukan sudah

terpenuhi. Test Case yang dilakukan antara lain test case Login Admin (TC-01), Logout Admin (TC-02), Install Plugin (TC-03), Melihat Daftar Plugin (TC-04), Menguninstall Plugin (TC-05), Mengaktifkan Plugin (TC-06), Menonaktifkan Plugin (TC-07), Mengatur Nilai Konfigurasi Sistem (TC-08), Melakukan Tes Kemiripan (TC-09), Melihat Report Tes per Quiz (TC-10), dan Melakukan Tes Ulang (TC-11). Oleh karena itu dapat disimpulkan bahwa sistem plugin Java Similarity Checker secara fungsional berjalan dengan baik.

6.2.2. Pembahasan Hasil Uji Coba Validitas

Dari hasil yang didapat pada uji coba validitas, dapat diketahui bahwa algoritma Levenshtein Distance dan Rabin Karp yang terpasang pada plugin "Java Similarity Checker" sesuai dengan algoritma Levenshtein Distance dan Rabin Karp yang ada pada sistem pencocokkan string yang sudah ada.

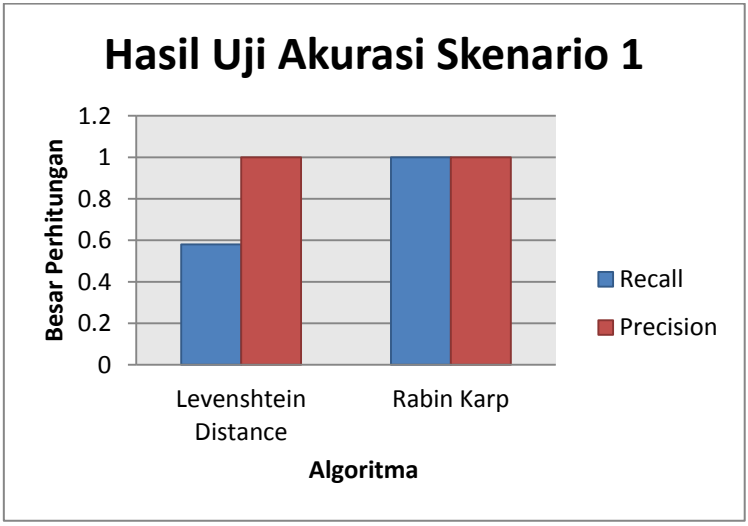
Pada tabel 6.3 diketahui bahwa untuk algoritma Levenshtein Distance dan Rabin Karp yang terpasang di plugin dan di aplikasi lain sama-sama menghasilkan persentase kesamaan terhadap dua kalimat. Secara keseluruhan pada hasil yang didapat, algoritma yang dibuat pada sistem plugin terbukti kebenarannya dan sudah sesuai dengan cara yang ada di aplikasi sebelumnya ataupun penghitungan secara manual.

6.2.3. Pembahasan Hasil Uji Coba Akurasi

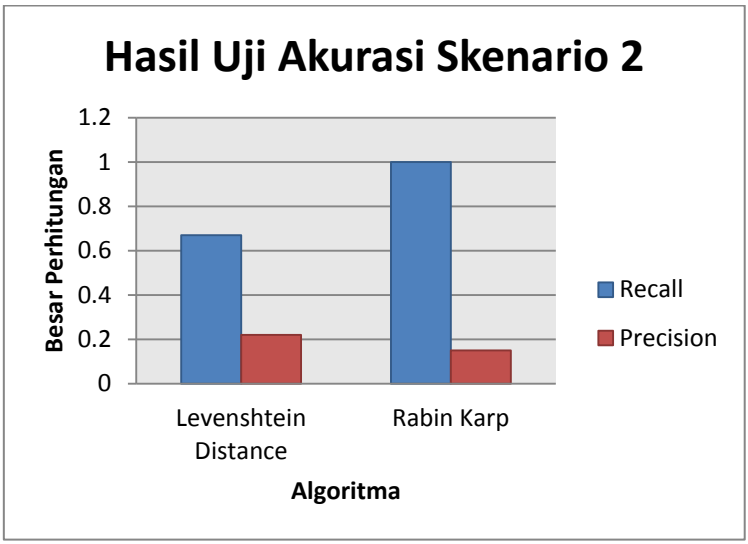
Dari diagram yang ada pada gambar 6.1, dapat diketahui bahwa pada skenario 1 algoritma Rabin-Karp dan Levenshtein Distance sama-sama memiliki nilai precision sebesar 1. Namun untuk nilai recall, algoritma Rabin Karp memiliki nilai lebih besar dari Levenshtein Distance. Algoritma Levenshtein Distance hanya mampu menghasilkan nilai recal sebesar 0.58.

Sedangkan pada uji coba pada gambar 6.2, diketahui bahwa pada skenario 2 algoritma Levenshtein Distance memiliki nilai recall lebih rendah daripada Rabin Karp. Nilai recall pada

Rabin Karp sebesar 1 sedangkan Levenshtein Distance hanya memperoleh nilai recall sebesar 0.67. Namun untuk nilai precision, algoritma Levenshtein Distance memperoleh nilai lebih tinggi daripada Rabin Karp. Algoritma Levenshtein Distance memperoleh nilai precision sebesar 0.22, sedangkan Rabin Karp hanya memperoleh nilai sebesar 0.15.



Gambar 6.1 hasil uji akurasi skenario1



Gambar 6.2 hasil uji akurasi skenario2

Dari kedua skenario yang sudah dilakukan, maka dapat ditarik kesimpulan bahwa algoritma Levenshtein Distance terbukti lebih tepat dalam menganalisa kemiripan pada source code yang terbukti mirip. Hal ini ditunjukkan dengan nilai precision dari algoritma Rabin Karp yang lebih rendah daripada Levenshtein Distance. Namun algoritma Rabin Karp terbukti berhasil dalam menghasilkan persentase kemiripan antar source code yang lebih besar daripada Levenshtein Distance. Hal ini dibuktikan dengan nilai recall dari algoritma Rabin Karp yang konsisten pada skenario1 dan skenario2.

(Halaman ini sengaja dikosongkan)

BAB VII
KESIMPULAN DAN SARAN

Bab ini berisikan kesimpulan dan saran dari seluruh proses pengerjaan tugas akhir. Kesimpulan dan saran diharapkan berguna untuk proses pengembangan sistem selanjutnya.

7.1. Kesimpulan

Berdasarkan dari proses pengerjaan tugas akhir ini, didapatkan beberapa kesimpulan sebagai berikut:

1. Sistem *plugin* untuk mendeteksi kemiripan pada *source code* atau yang dinamakan Java Similarity Checker berhasil terpasang dan berjalan pada elearning untuk pembelajaran Java yaitu Belajar Java. Sistem plugin dapat berjalan dengan menerapkan pola Dependency Injection. Sistem plugin yang dipasang dapat berjalan karena adanya Plugin Manager yang telah dibuat pada aplikasi yang berguna untuk memanajemen plugin yang ada pada aplikasi. Pembuatan sistem plugin ini melalui beberapa tahap yaitu tahap analisis, desain, pembuatan, dan pengujian aplikasi. Berdasarkan uji coba fungsional yang telah dilakukan, semua fungsi yang ada pada sistem berjalan dengan baik.
2. Dari hasil yang didapat pada uji coba validitas, dapat diketahui bahwa algoritma Levenshtein Distance dan Rabin Karp yang terpasang pada plugin “Java Similarity Checker” sesuai dengan algoritma Levenshtein Distance dan Rabin Karp yang ada pada sistem pencocokkan string yang sudah ada.
3. Berdasarkan sebagian besar hasil uji coba akurasi, dapat diketahui bahwa algoritma Levenshtein Distance terbukti lebih tepat dalam menganalisa kemiripan pada source code yang terbukti mirip. Hal ini ditunjukkan dengan nilai precision dari algoritma Rabin Karp yang lebih tinggi daripada Levenshtein Distance. Namun algoritma Rabin Karp terbukti berhasil dalam menghasilkan persentase

kemiripan antar source code yang lebih besar daripada Levenshtein Distance.

7.2. Saran

Karena masih terdapat beberapa kekurangan pada sistem plugin yang perlu diperbaiki dan dikembangkan lagi, maka terdapat beberapa saran yang diperlukan untuk penelitian selanjutnya yaitu:

1. Karena sistem plugin ini dapat berjalan di aplikasi Belajar Java, maka untuk kedepannya perlu dikembangkan sistem plugin yang sama namun dengan sistem elearning yang lain.
2. Perlu adanya pengembangan sistem plugin lain yang dapat berjalan pada plugin manager yang telah dibuat pada aplikasi Belajar Java.

DAFTAR PUSTAKA

[1] C. Park, ***Rebels Without a Clause: Towards an Institutional Framework for Dealing***,*Journal of Further and Higher Education Vol. 28, No. 3*, 2004.

[2] A. G. Liaqat and A. Ahmad, ***Plagiarism Detection in Java Code***, 2011.

[3] F. Rakhman, A. T. Wibowo and E. Darwiyanto, ***Analisa dan Implementasi Algoritma Edit Distance Sebagai Alat Bantu Pendeteksi Plagiarisme Source Code***,*Repository Telkom University*, 2012.

[4] M. Mozgovoy, ***Desktop Tools for Offline Plagiarism Detection in Computer Programs***,*Department of Computer Science*, University of Joensuu, 2006.

[5] H. B. Firdaus, **Deteksi Plagiat Dokumen Menggunakan Algoritma Rabin-Karp**,*Jurnal Ilmu Komputer Dan Teknologi Informasi*, 2003.

[6] D. Tresnawati, A. R. Syaichu and Kuspriyanto, ***Plagiarism Detection System Design for Programming Assignment in Virtual Classroom Based on Moodle***,*Procedia - Social and Behavioral Sciences 67*, pp. 114-122, 2011.

[7] J. Gosling, B. Joy, G. Steele, G. Bracha and A. Buckley, The Java® Language Specification, 2014.

[8] M. Fowler, "Plugin," [Online]. Available: http://martinfowler.com/eaaCatalog/plugin.html. [Accessed 2014].

[9] E. Gamma and e. al, Design Patterns, Addison-Wesley, 1994.

[10] A. Osmani, ***Learning Java Script Design Patterns***, *O'Reilly Media*, 2012.

[11] D. Kaye, ***Loosely Coupled: The Missing Pieces of Web Services***, 2003.

[12] D. Bock, **The Paperboy, The Wallet, and The Law Of Demeter**,*College of Computer and Information Science, Northeastern University*, 2012.

[13] M. Fowler, "Inversion of Control Containers and the Dependency Injection pattern," [Online]. Available: http://martinfowler.com/articles/injection.html. [Accessed 2014].

[14] S. Fish, "Plagiarism Is Not a Big Moral Deal," 2010. [Online]. Available: http://opinionator.blogs.nytimes.com/2010/08/09/plagiarism-is-not-a-big-moral-deal. [Accessed 2014].

[15] J. Jeong-Hoon, W. Gyun and C. Hwan-Gue, ***A Plagiarism Detection Technique for Java Program Using Bytecode Analysis***,*Third 2008 International Conference on Convergence and Hybrid Information Technology*, pp. 1092-1098, 2008.

[16] R. D. Dewandono, F. A. Saputra and S. Rochimah,***Clone Detection Using Rabin-karp Parallel Algorithm***, in *The Proceedings of The 7th ICTS*, Bali, 2013.

[17] A. Jadalla and A. Elnagar,***PDE4Java: Plagiarism Detection Engine For Java***, in *Proceedings of iiWAS2007*, 2007.

[18] F. Sutanto, ***Perancangan Program Aplikasi Pendeteksi Plagiarisme Source Code Dengan Menggunakan Metode Edit Distance (Studi Kasus: UPTPL Universitas Bina Nusantara),****Universitas Bina Nusantara*, 2008.

[19] WordPress, "WordPress.org," [Online]. Available: https://wordpress.org/. [Accessed 2015].

[20] F. Sugiyanto, "Plagiarisme Musuh Bersama," Suara Merdeka, 2010.

[21] B. Stein and S. M. z. Eissen, ***Near Similarity Search and Plagiarism Analysis***, in *the 29th Annual Conference of the German Classification Society (GfKl)*, Magdeburg, 2006.

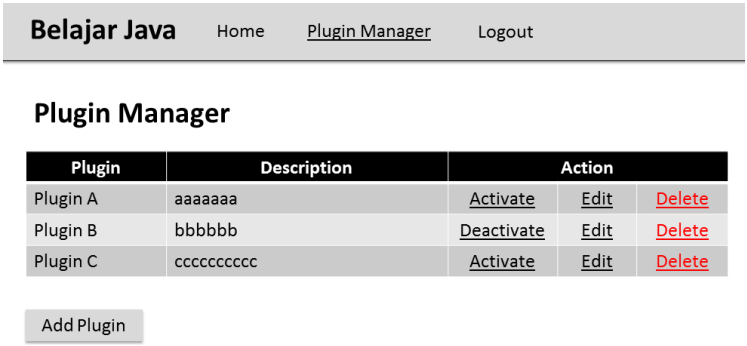
[22] I. S. Sipahutar, ***Pengembangan Aplikasi Berbasis Web Interaktif untuk Belajar Dasar Pemrograman Java (Studi Kasus Jurusan Sistem Informasi ITS Surabaya)***,*Digital Library ITS*, 2013.

[23] H. Schildt, ***Java: The Complete Reference, Seventh Edition***, New York City: The McGraw-Hill Companies, 2007.

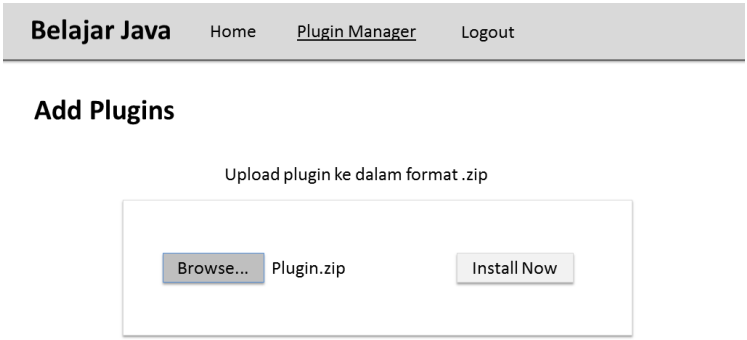
[24] R. K. Ellis, ***Field Guide to Learning Management Systems***, the American Society for Training & Development (ASTD), 2009.

Halaman ini Sengaja Dikosongkan

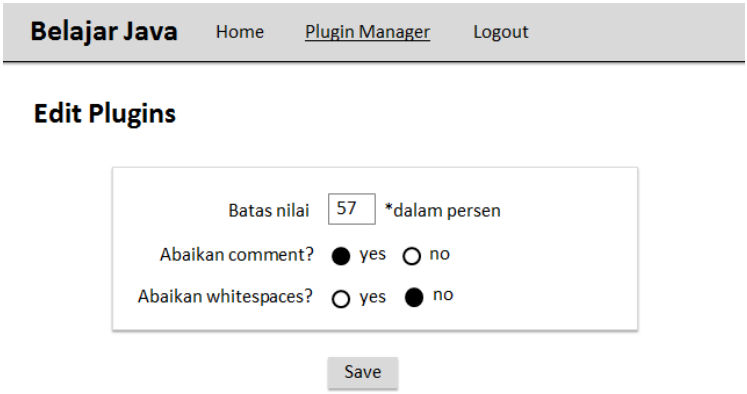
Lampiran A



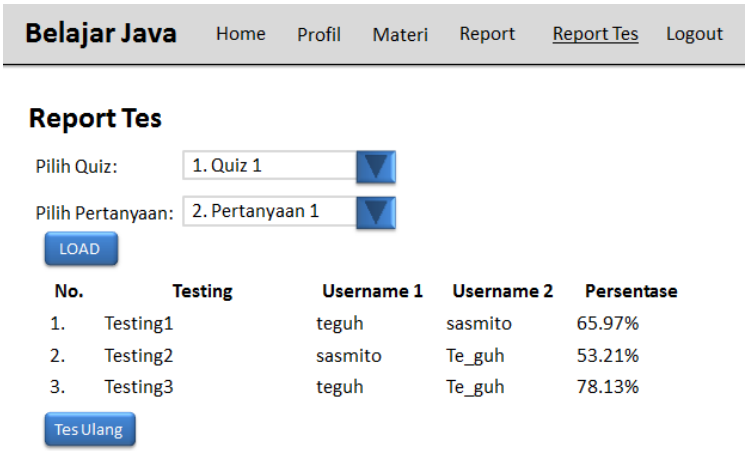
Gambar A.1 halaman plugin manager



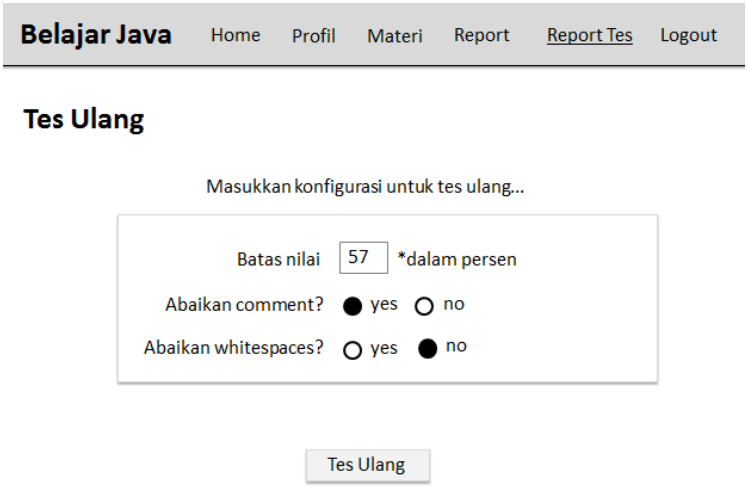
Gambar A.2 halaman tambah plugin



Gambar A.3 halaman edit plugin



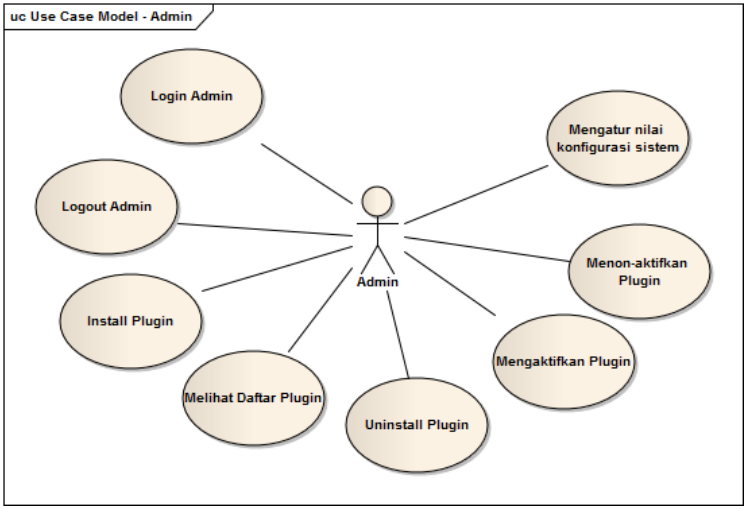
Gambar A.4 halaman report tes



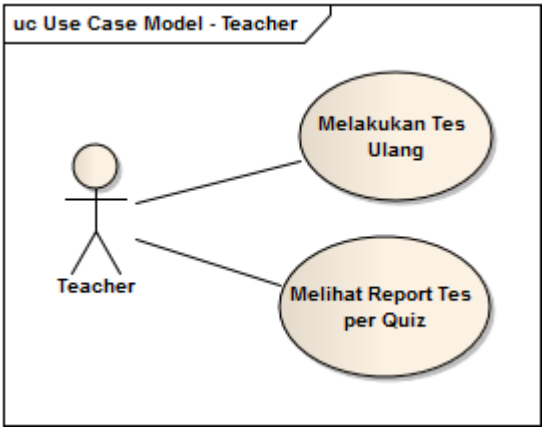
Gambar A.5 halaman tes ulang

Halaman ini Sengaja Dikosongkan

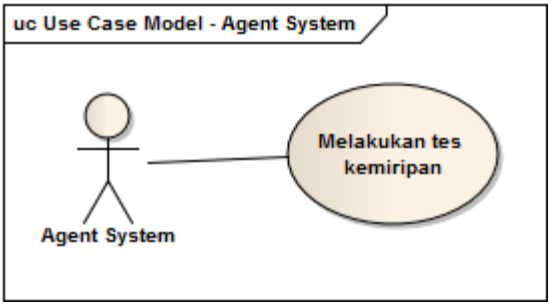
Lampiran B



Gambar B.1 Use Case Model - Admin



Gambar B.2 Use Case Model - Teacher



Gambar B.3 Use Case Model – System

Tabel B.1 Skenario UC-01

Use Case Code	UC-01
Use Case Name	Login Admin
Use Case User	Admin
Skenario	
Basic	Admin berada pada halaman login. Admin mengisi username dan password pada field yang telah disediakan. Kemudian Admin mengklik login. Sistem melakukan validasi data yang dimasukkan oleh admin dengan *role* yang ada di dalam database. Sistem menampilkan halaman admin.
Username atau Password tidak sesuai database	Sistem menampilkan pesan username atau password salah

Tabel B.2 Skenario UC-02

Use Case Code	UC-02
Use Case Name	Logout Admin
Use Case User	Admin
Skenario	
Basic	Admin mengklik tombol logout yang terdapat disetiap halaman web. Sistem menghapus informasi login admin.

	Sistem kemudian menampilkan halaman login.

Tabel B.3 Skenario UC-03

Use Case Code	UC-03
Use Case Name	Install Plugin
Use Case User	Admin
Skenario	
Basic	Pada halaman admin, admin mengklik menu plugin manager. Sistem menampilkan halaman plugin manager. Admin mengklik tombol add, sistem akan menampilkan halaman tambah plugin. Admin memilih tombol "Browse", sistem menampilkan halaman windows explorer. Admin memilih file, sistem menampilkan keterangan file di halaman tambah plugin. Admin mengklik tombol “Install”, sistem mengecek jenis file. Sistem mengekstraksi file dan mengecek apakah ada file .txt yang berisi keterangan plugin di dalam file ekstraksi. Sistem menyimpan deskripsi plugin ke dalam database. Sistem menyimpan plugin ke dalam direktori.
File bukan file .zip	Sistem gagal menginstall plugin. Sistem menampilkan pesan gagal menginstall.
File ekstraksi tidak berisi file .txt	Sistem mengahapus file plugin yang ada di direktori. Sistem menampilkan pesan gagal menginstall.
File ekstraksi tidak berisi file index.php	Sistem mengahapus file plugin yang ada di direktori. Sistem menampilkan pesan gagal menginstall.

Tabel B.4 Skenario UC-04

Use Case Code	UC-04
Use Case Name	Melihat Daftar Plugin
Use Case User	Admin
Skenario	
Basic	Admin mengklik menu Plugin Manager pada halaman Admin. Sistem mengecek data plugin yang ada di database. Sistem menampilkan daftar plugin pada halaman Plugin Manager.
Belum ada data plugin	Sistem menampilkan pesan belum ada data

Tabel B.5 Skenario UC-05

Use Case Code	UC-05
Use Case Name	Uninstall Plugin
Use Case User	Admin
Skenario	
Basic	Pada halaman admin, admin mengklik menu plugin manager. Sistem menampilkan halaman plugin manager. Admin mengklik tombol delete pada daftar plugin yang ingin di-uninstall. Sistem menampilkan pesan konfirmasi. Admin mengklik tombol "oke". Sistem kemudian mengecek status keaktifan dari plugin. Sistem menghapus plugin dari direktori dan database. Sistem mengembalikan tampilan ke halaman plugin manager.
Plugin masih aktif	Sistem menampilkan pesan bahwa plugin masih aktif

Tabel B.6 Skenario UC-06

Use Case Code	UC-06
Use Case Name	Mengaktifkan Plugin
Use Case User	Admin
Skenario	
Basic	Pada halaman admin, admin mengklik menu plugin manager. Sistem menampilkan halaman plugin manager. Admin mengklik tombol activate pada daftar plugin yang ingin diaktifkan. Sistem mengaktifkan status plugin.

Tabel B.7 Skenario UC-07

Use Case Code	UC-07
Use Case Name	Menon-aktifkan Plugin
Use Case User	Admin
Skenario	
Basic	Pada halaman admin, admin mengklik menu plugin manager. Sistem menampilkan halaman plugin manager. Admin mengklik tombol deactivate pada daftar plugin yang ingin dinon-aktifkan. Sistem menon-aktifkan status plugin.

Tabel B.8 Skenario UC-08

Use Case Code	UC-08
Use Case Name	Mengatur Nilai Konfigurasi Sistem
Use Case User	Admin
Skenario	
Basic	Pada halaman plugin manager, admin mengklik tombol edit pada plugin. Sistem menampilkan halaman edit pada plugin. Admin mengubah konfigurasi yang terdiri dari pengisian threshold, menghilangkan comment

	dan identifier. Admin mengklik tombol "Save". Sistem memvalidasi pengaturan konfigurasi yang diisi oleh admin. Sistem menyimpan konfigurasi di dalam direktori. Sistem menampilkan pesan konfigurasi berhasil disimpan. Sistem kembali ke halaman plugin manager.
Threshold yang diinputkan bukan berupa angka	Sistem menampilkan pesan input threshold harus berupa angka

Tabel B.9 Skenario UC-09

Use Case Code	UC-09
Use Case Name	Melakukan Tes Kemiripan
Use Case User	Agent System
Skenario	
Basic	Sistem mengecek ketersediaan file jawaban quiz yang ada di direktori. Sistem mengecek kemiripan jawaban sesuai konfigurasi awal dari sistem. Sistem menyimpan hasil tes kemiripan ke dalam database.
Setelah satu menit proses uji kemiripan	Sistem mengecek ulang file jawaban di direktori. Sistem mengecek kemiripan antar dua jawaban yang belum diuji.
File di direktori < 2 file	Sistem mengecek kemiripan satu menit kemudian.
Tes kemiripan melebihi threshold yang telah ditentukan	Sistem mengirimkan pesan pengingat ke email teacher yang tertera di akun teacher

Tabel B.10 Skenario UC-10

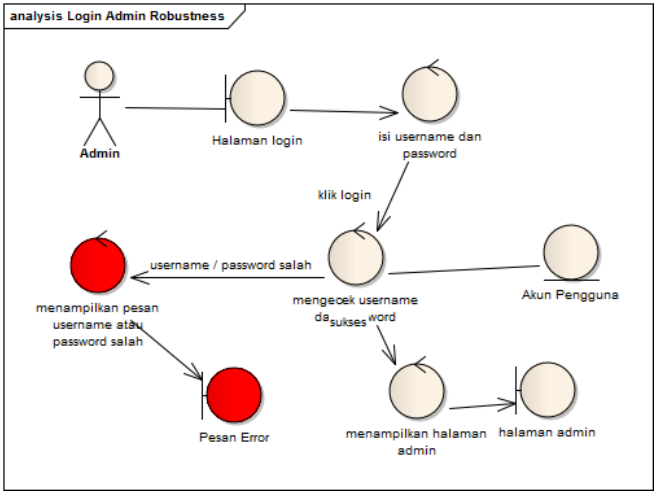
Use Case Code	UC-10
Use Case Name	Melihat Report Tes per Quiz
Use Case User	Teacher
Skenario	
Basic	Teacher mengklik tombol report yang ada pada menu utama web. Sistem kemudian menampilkan halaman report tes. Teacher memilih quiz yang ingin dilihat. Sistem menampilkan hasil tes kemiripan sesuai quiz yang dipilih.
Teacher belum memilih pertanyaan	Sistem menampilkan hasil tes kemiripan dari pilihan pertama pada field pertanyaan
Belum ada pengguna yang mengerjakan quiz	Sistem menampilkan pesan tidak ada data
Masih satu pengguna yang selesai mengerjakan quiz	Sistem menampilkan pesan tidak ada data

Tabel B.11 Skenario UC-11

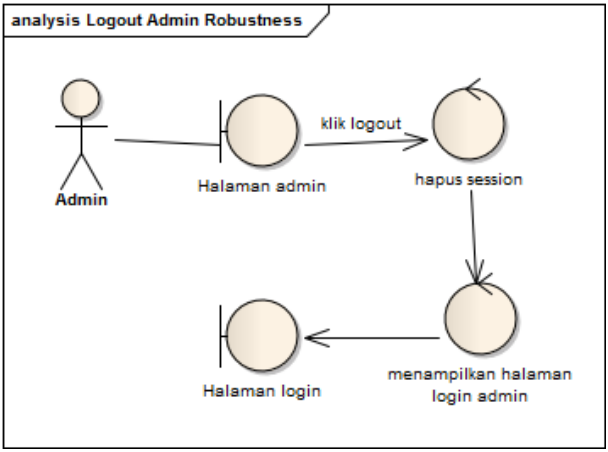
Use Case Code	UC-11
Use Case Name	Melakukan Tes Ulang
Use Case User	Teacher
Skenario	
Basic	Teacher berada pada halaman report tes salah satu pertanyaan, kemudian teacher mengklik tombol tes ulang. Sistem menampilkan halaman tes ulang. Teacher mengatur konfigurasi berupa pengisian batas nilai token,

	menghilangkan comment dan whitespaces untuk pengujian ulang. Sistem mengecek jawaban sesuai konfigurasi yang diatur oleh teacher. Sistem menampilkan hasil tes ulang pada halaman report tes kemiripan.
Teacher belum mengisikan konfigurasi	Sistem menampilkan hasil tes kemiripan sesuai default konfigurasi dari sistem.
Threshold yang diinputkan bukan berupa angka	Sistem menampilkan pesan input threshold harus berupa angka

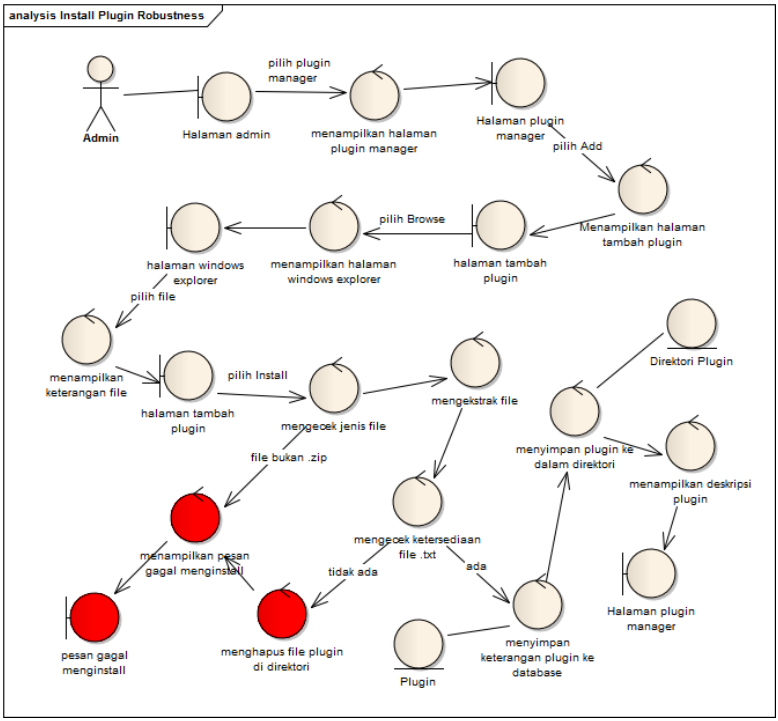
Lampiran C



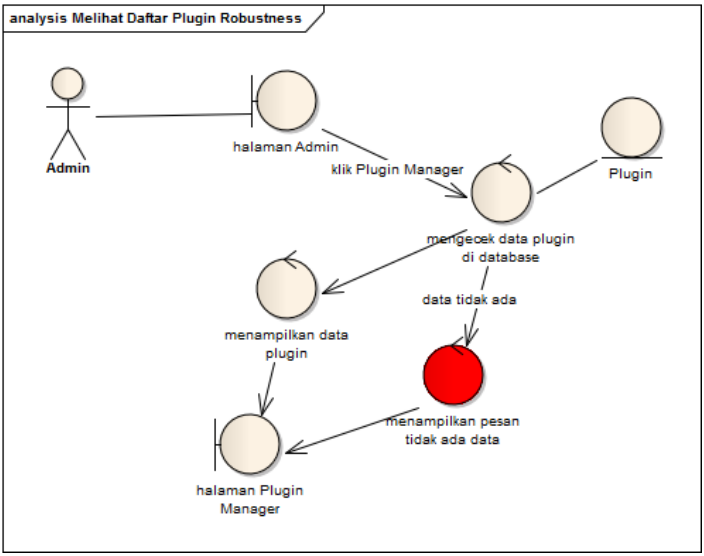
Gambar C.1 Robustness Diagram UC-01



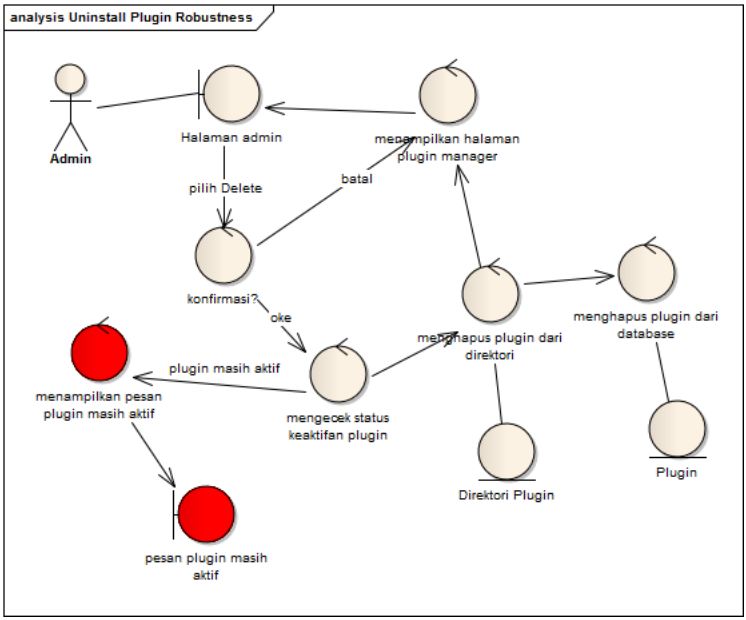
Gambar C.2 Robustness Diagram UC-02



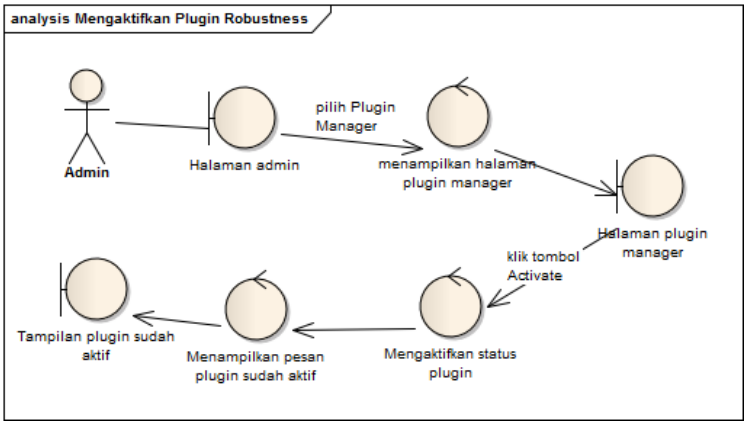
Gambar C.3 Robustness Diagram UC-03



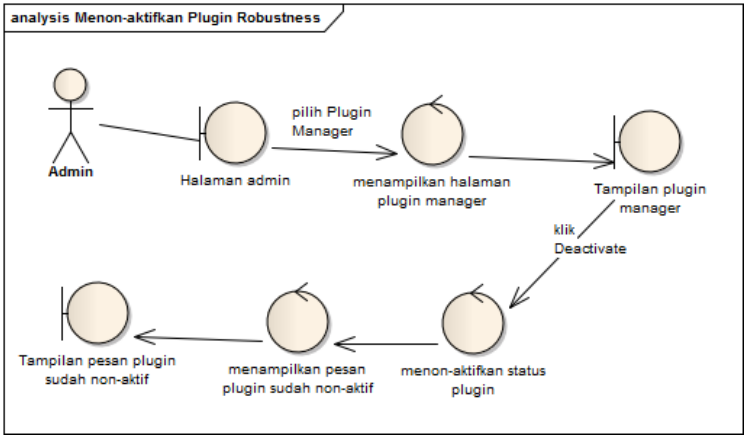
Gambar C.4 Robustness Diagram UC-04



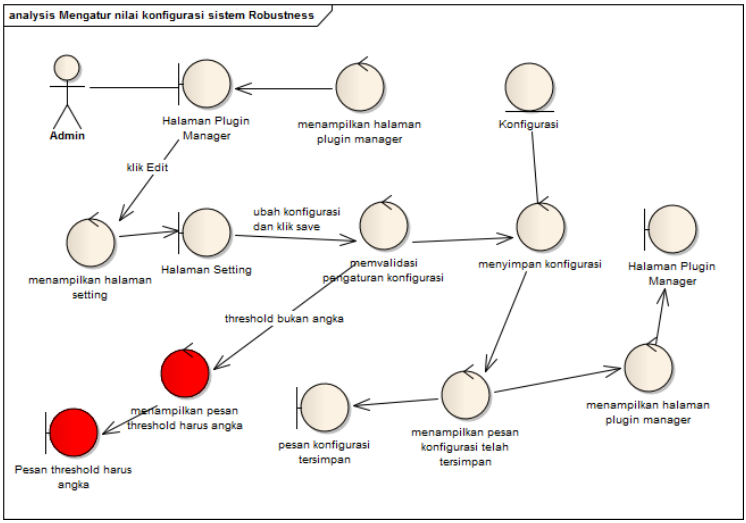
Gambar C.5 Robustness Diagram UC-05



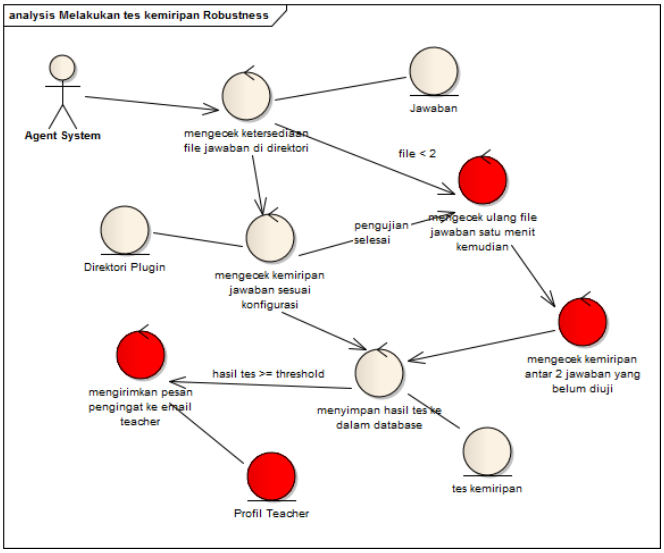
Gambar C.6 Robustness Diagram UC-06



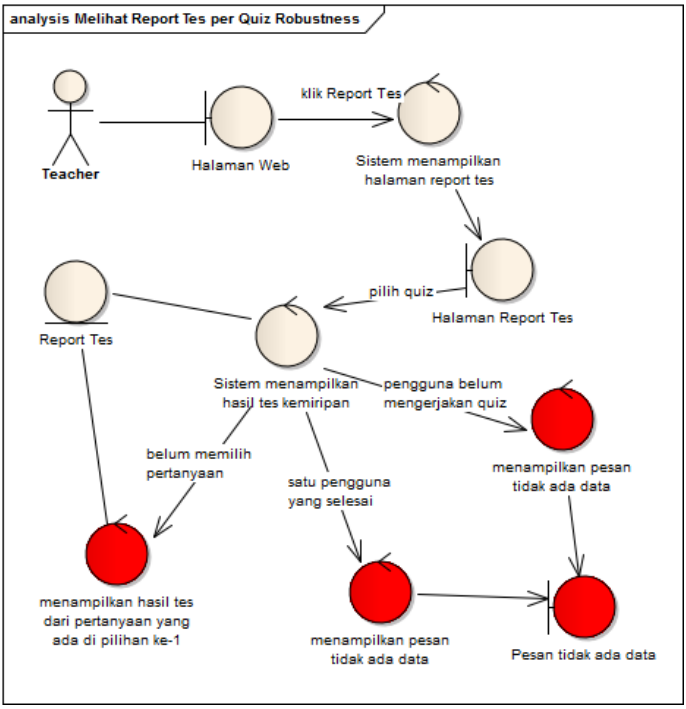
Gambar C.7 Robustness Diagram UC-07



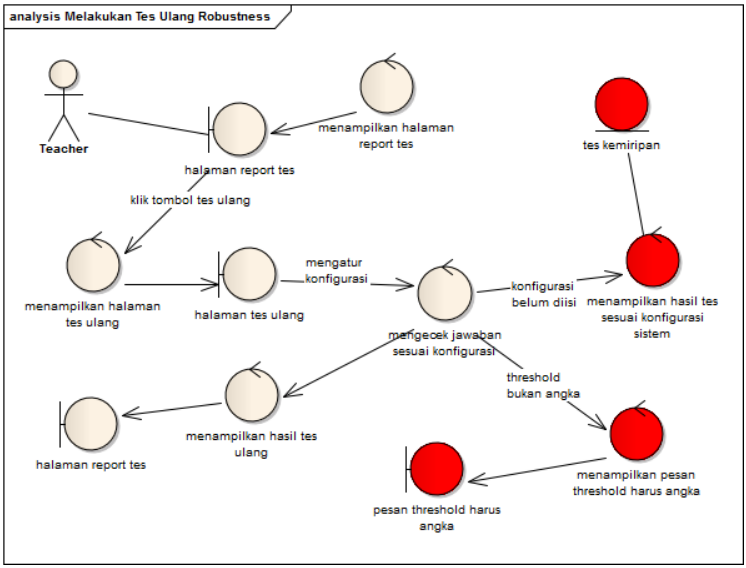
Gambar C.8 Robustness Diagram UC-08



Gambar C.9 Robustness Diagram UC-09



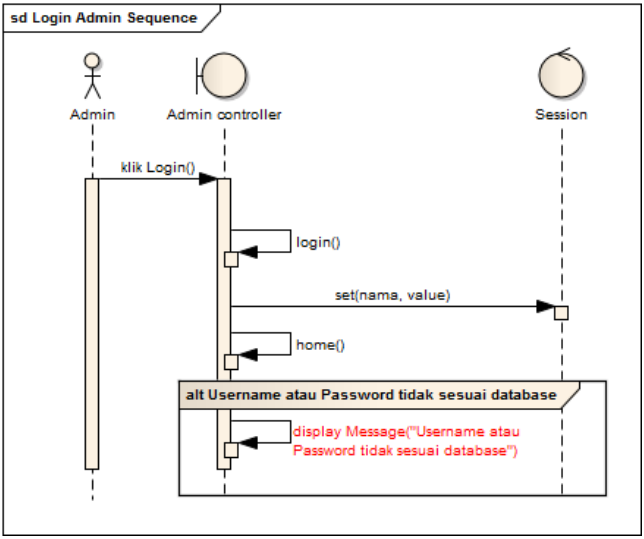
Gambar C.10 Robustness Diagram UC-10



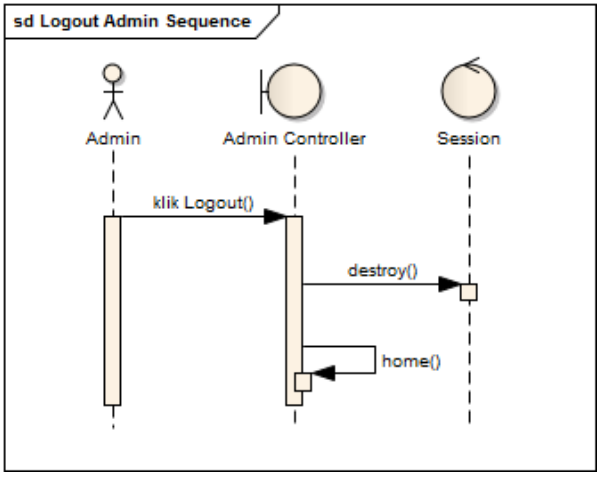
Gambar C.11 Robustness Diagram UC-11

Halaman ini Sengaja Dikosongkan

Lampiran D

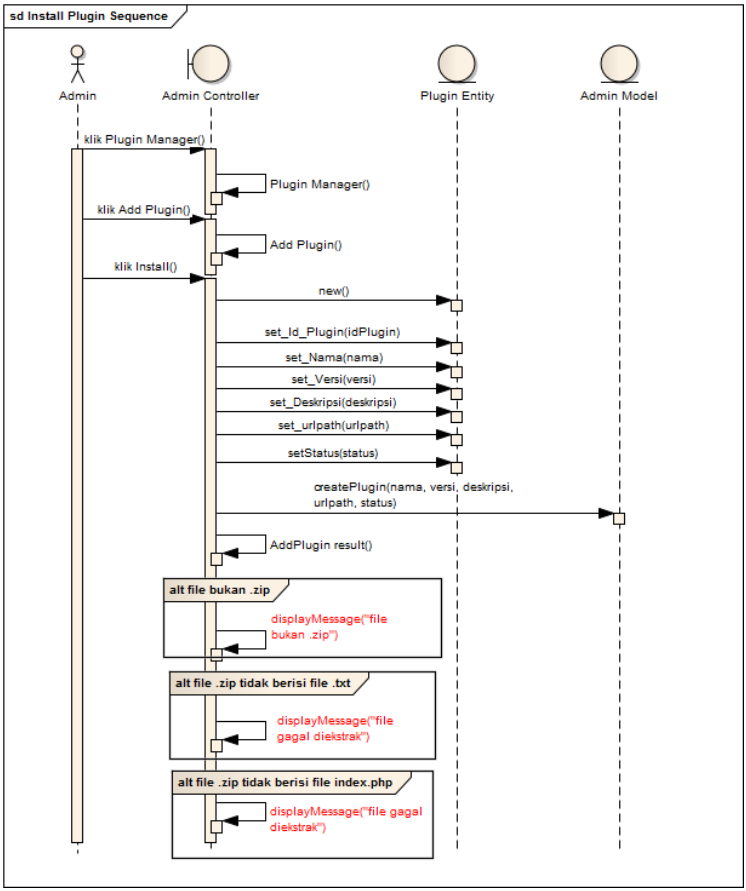


Gambar D.1 Sequence Diagram UC-01

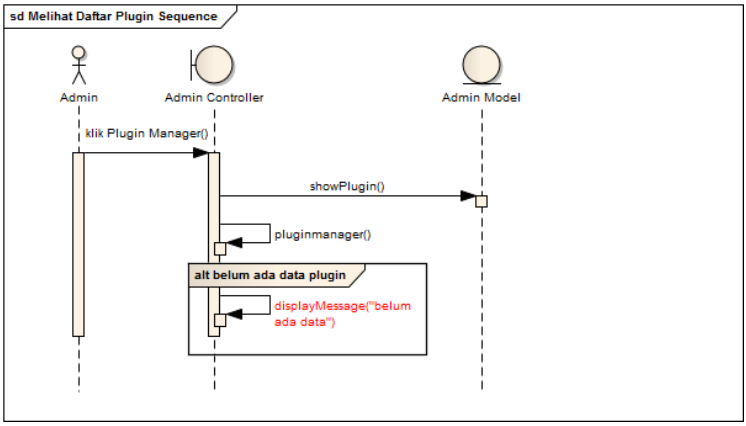


Gambar D.2 Sequence Diagram UC-02

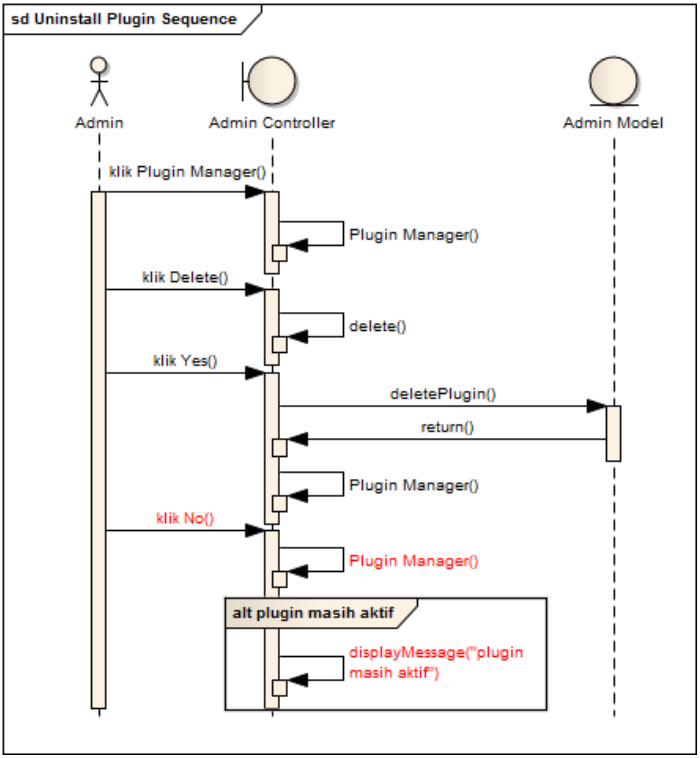
D-2



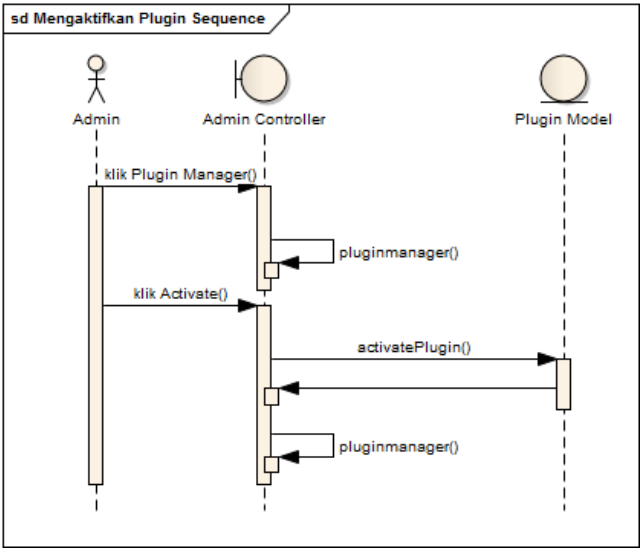
Gambar D.3 Sequence Diagram UC-03



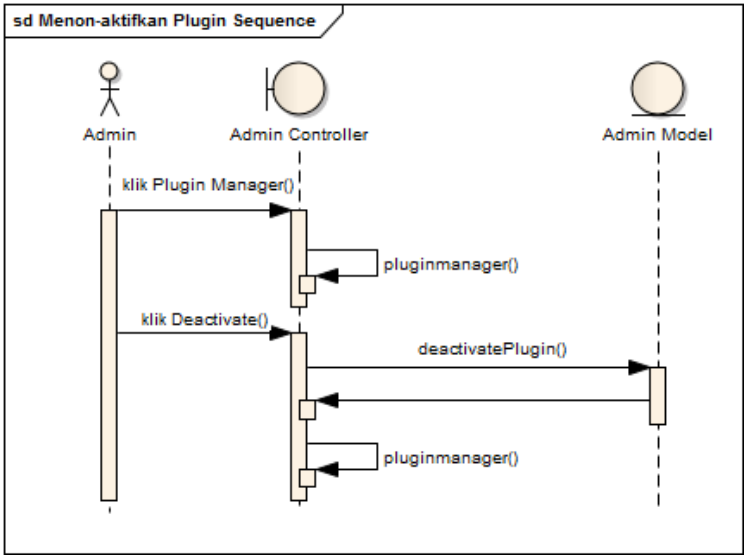
Gambar D.4 Sequence Diagram UC-04



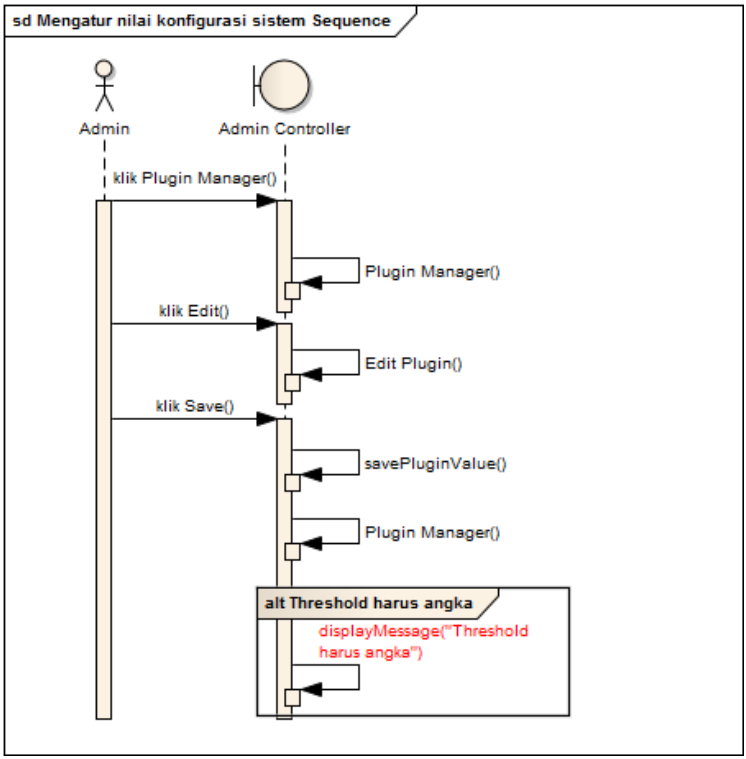
Gambar D.5 Sequence Diagram UC-05



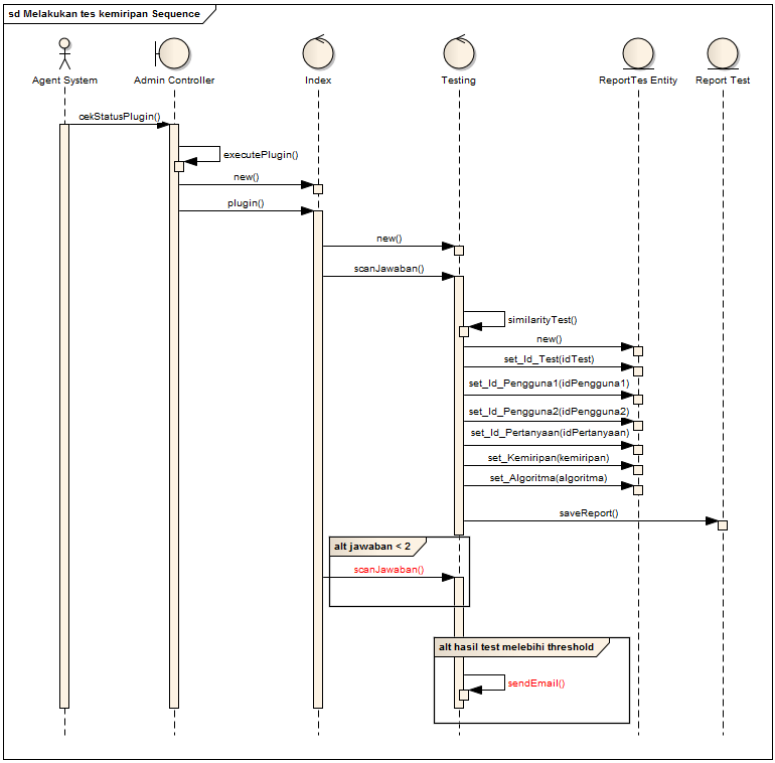
Gambar D.6 Sequence Diagram UC-06



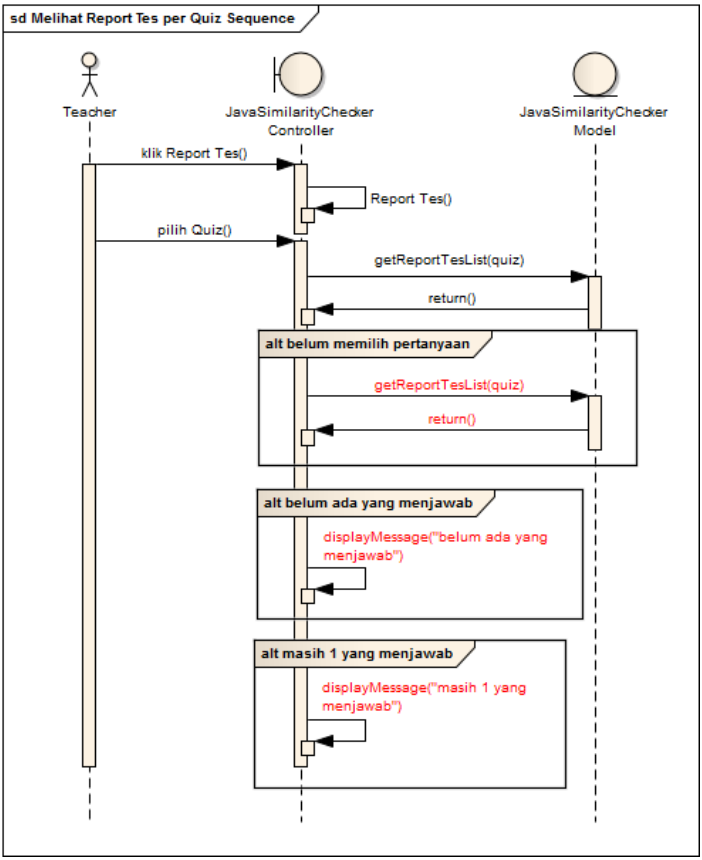
Gambar D.7 Sequence Diagram UC-07



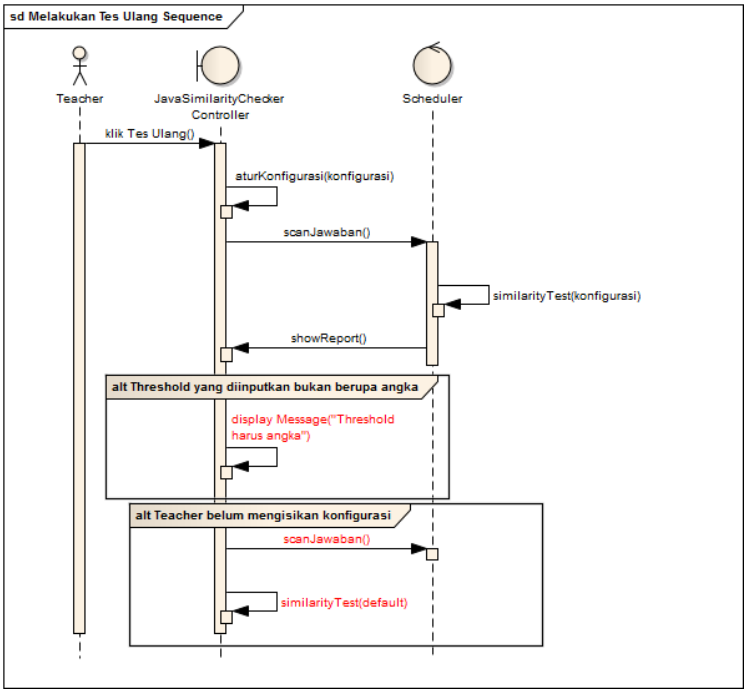
Gambar D.8 Sequence Diagram UC-08



Gambar D.9 Sequence Diagram UC-09

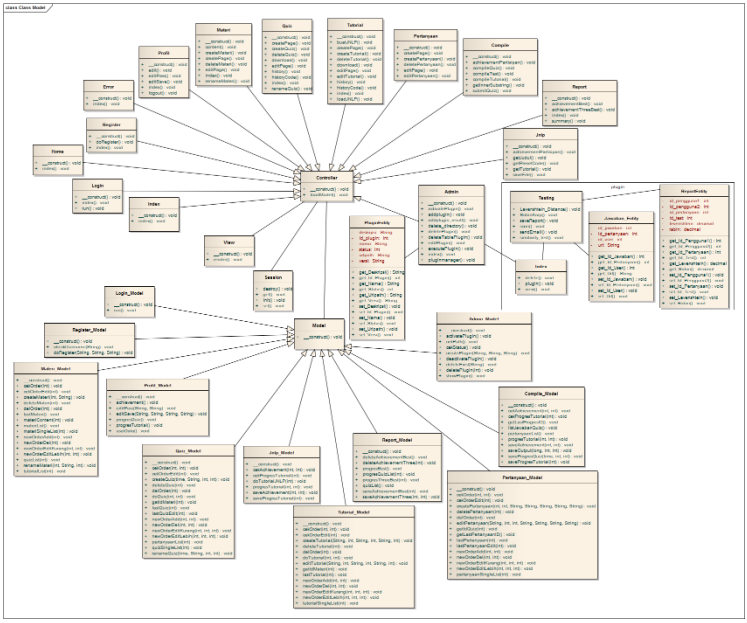


Gambar D.10 Sequence Diagram UC-10



Gambar D.11 Sequence Diagram UC-11

Lampiran E



Gambar E.1 Class Diagram

Halaman ini sengaja Dikosongkan

Lampiran F

Tabel F.1 Test Case - 01

Test Case Code		TC-01			
Test Case Name		Login Admin			
No.	Aksi Test	Data Test	Hasil yang Diharapkan	Hasil Aktual	Sukses/ Gagal
1	Login dengan username dan password yang sudah terdaftar	Username = “administrator" Password = “administrator”	User masuk ke halaman Admin	Sistem memasukkan user ke dalam aplikasi, user masuk ke halaman home Admin	Sukses
2	Login dengan username dan password yang belum terdaftar	Username = “mimin” Password = “mimin”	User tidak masuk ke aplikasi	Sistem menampilkan pesan username atau password salah	Sukses

Tabel F.2 Test Case - 02

Test Case Code		TC-02			
Test Case Name		Logout Admin			
No.	Aksi Test	Data Test	Hasil yang Diharapkan	Hasil Aktual	Sukses/ Gagal
1	Logout dengan akun yang sedang login	Login user = “administrator"	User keluar dari aplikasi	Sistem mengeluarkan user dari aplikasi, user kembali ke halaman login	Sukses

Tabel F.3 Test Case - 03

Test Case Code		TC-03			
Test Case Name		Install Plugin			
No.	Aksi Test	Data Test	Hasil yang Diharapkan	Hasil Aktual	Sukses/ Gagal
1	Menginstall file .zip yang berisi file .txt dan index.php di dalamnya	Login user = “administrator” File = “xxx.zip”	Plugin berhasil diinstall	Sistem memasukkan info plugin ke database dan membuat direktori untuk menyimpan file plugin. Sistem	Sukses

				menampilkan pesan plugin berhasil diinstall	
2	Menginstall file bukan .zip	Login user = “administrator”	Plugin gagal dinstall	Sistem menampilkan pesan plugin gagal dinstall	Sukses
3	Menginstall file.zip yang tidak berisi file .txt dan index.php	Login user = “administrator”	Plugin gagal dinstall	Sistem menampilkan pesan plugin gagal dinstall	Sukses

Tabel F.4 Test Case - 04

Test Case Code		TC-04			
Test Case Name		Melihat Daftar Plugin			
No.	Aksi Test	Data Test	Hasil yang Diharapkan	Hasil Aktual	Sukses/ Gagal
1	Mengakses daftar plugin	Login user = “administrator"	User berada di halaman plugin manager	Sistem menampilkan daftar plugin	Sukses

Tabel F.5 Test Case - 05

Test Case Code		TC-05			
Test Case Name		Uninstall Plugin			
No.	Aksi Test	Data Test	Hasil yang Diharapkan	Hasil Aktual	Sukses/ Gagal
1	Menguninstall plugin	Login user = “administrator" Plugin name = “abc”	Sistem menghapus plugin dari daftar plugin	Sistem menghapus info plugin di database dan menghapus direktori plugin	Sukses
2	Menguninstall plugin yang sedang aktif	Login user = “administrator" Plugin name = “abc”	Sistem menampilkan pesan plugin masih aktif	Sistem menampilkan pesan plugin masih aktif	Sukses

Tabel F.6 Test Case - 06

Test Case Code		TC-06			
Test Case Name		Mengaktifkan plugin			
No.	Aksi Test	Data Test	Hasil yang Diharapkan	Hasil Aktual	Sukses/ Gagal
1	Mengaktifkan plugin	Login user = “administrator"	Sistem mengubah status plugin	Sistem mengubah status plugin dari non-	Sukses

		Plugin name = “abc”		aktif ke aktif	

Tabel F.7 Test Case - 07

Test Case Code		TC-07			
Test Case Name		Menonaktifkan Plugin			
No.	Aksi Test	Data Test	Hasil yang Diharapkan	Hasil Aktual	Sukses/ Gagal
1	Menonaktifkan plugin	Login user = “administrator" Plugin name = “abc”	Sistem mengubah status plugin	Sistem mengubah status plugin dari aktif ke non-aktif	Sukses

Tabel F.8 Test Case - 08

Test Case Code		TC-08			
Test Case Name		Mengatur Nilai Konfigurasi Sistem			
No.	Aksi Test	Data Test	Hasil yang Diharapkan	Hasil Aktual	Sukses/ Gagal
1	Menampilkan nilai konfigurasi plugin sebelumnya	Login user = “administrator" Plugin name =	Sistem menampilkan nilai konfigurasi sebelumnya	Sistem menampilkan nilai konfigurasi sebelumnya yang ada	Sukses

		“abc” Nilai konfigurasi: - Hapus komentar: “NO” - Hapus spasi: “YES” - Batas nilai: “65”		di file”konfigurasi.txt”	
2	Menyimpan nilai konfigurasi plugin yang telah diubah	Login user = “administrator" Plugin name = “abc” Nilai konfigurasi: - Hapus komentar: “YES” - Hapus spasi: “YES” - Batas nilai: “75”	Sistem menyimpan nilai konfigurasi yang telah diubah	Sistem menyimpan nilai konfigurasi yang telah diubah pada file “konfigurasi.txt”	Sukses

Tabel F.9 Test Case - 09

Test Case Code		TC-09			
Test Case Name		Melakukan Tes Kemiripan			
No.	Aksi Test	Data Test	Hasil yang	Hasil Aktual	Sukses/

			Diharapkan		Gagal
1	Melakukan tes kemiripan pada jawaban yang ada di direktori	Id_Jawaban = {1,2,3,4,5}	Sistem menyimpan hasil tes ke database	Sistem melakukan tes kemiripan pada jawaban yang ada dan menyimpan hasil tes ke database	Sukses
2	Melakukan tes kemiripan kembali pada jawaban yang ada di direktori setelah jeda 5 menit	Id_Jawaban = {1,2,3,4,5}	Sistem menyimpan hasil tes ke database	Sistem melakukan tes kemiripan pada jawaban yang ada dan menyimpan hasil tes ke database	Sukses
3	Mengirim pesan via email	Persentase kemiripan = “85%” Nilai konfigurasi: - Batas nilai: “75”	Sistem mengirimkan pesan kepada teacher via email jika nilai kemiripan melebihi batas nilai yang telah ditentukan	Sistem dapatmengirimkan pesan via email saat ada nilai kemiripan > batas nilai	Sukses

Tabel F.10 Test Case - 10

Test Case Code		TC-10			
Test Case Name		Melihat Report Tes per Quiz			
No.	Aksi Test	Data Test	Hasil yang Diharapkan	Hasil Aktual	Sukses/ Gagal
1	Melihat detail report test	Login user = “teacher" ID Quiz = “5”	User masuk ke halaman detail report test, sistem menampilkan detail report test	Sistem memasukkan user ke halaman detail report test dan menampilkan hasil report test secara detail	Sukses

Tabel F.11 Test Case - 11

Test Case Code		TC-11			
Test Case Name		Melakukan Tes Ulang			
No.	Aksi Test	Data Test	Hasil yang Diharapkan	Hasil Aktual	Sukses/ Gagal
1	Melakukan tes ulang	Login user = “teacher" Nilai konfigurasi: - Hapus komentar: “YES”	Sistem menampilkan hasil tes ulang sesuai konfigurasi yang telah diatur berdasarkan quiz	Sistem melakukan tes kemiripan ulang sesuai nilai konfigurasi yang diatur user, sistem menampilkan hasil tes	Sukses

		- Hapus spasi: “YES”	yang telah dipilih	ulang sesuai konfigurasi yang telah diatur berdasarkan quiz yang telah dipilih	

Halaman ini Sengaja Dikosongkan

Lampiran G

```
public class Belajar {
	public static void main(String args[]){
	int a[] = {9, 20, 15, 18, 1, 3, 2, 5, 6, 8, 10, 11, 14, 13, 4, 7, 12, 17, 19, 16};
	for ( int kiri = 0; kiri < a.length - 1; kiri++ ){
		for ( int kanan = kiri + 1; kanan < a.length; kanan++ ){
			if ( a[ kanan ] > a[ kiri ] ){
				int temporary = a[ kiri ];
				a[ kiri ] = a[ kanan ];
				a[ kanan ] = temporary;
			}
		}
	}
	System.out.println(a[0]+" "+a[1]+" "+a[2]+" "+a[3]+" "+a[4]+" "+a[5]+" "+a[6]+"
"+a[7]+" "+a[8]+" "+a[9]+" "+a[10]+" "+a[11]+" "+a[12]+" "+a[13]+" "+a[14]+" "+a[15]+"
"+a[16]+" "+a[17]+" "+a[18]+" "+a[19]);
	}
}
```

```
public class Belajar {
    public static void main(String args[]){

     int a[] = {9, 2, 11, 1, 10, 3, 20, 5, 6, 8, 18, 15, 4, 13, 14, 7, 12, 17, 19, 16};


    for ( int kiri = 0; kiri < a.length - 1; kiri++ ){

        for ( int kanan = kiri + 1; kanan < a.length; kanan++ ){
            if ( a[ kanan ] > a[ kiri ] ){

                int temporary = a[ kiri ];
                a[ kiri ] = a[ kanan ];
                a[ kanan ] = temporary;


            }
        }
    }

    System.out.println(a[0]+" "+a[1]+" "+a[2]+" "+a[3]+" "+a[4]+" "+a[5]+" "+a[6]+" "+a
[7]+" "+a[8]+" "+a[9]+" "+a[10]+" "+a[11]+" "+a[12]+" "+a[13]+" "+a[14]+" "+a[15]+" "+a
[16]+" "+a[17]+" "+a[18]+" "+a[19]);

    }
}
```

Gambar G.1 Source Code User1 dan User2

G-2

```
public class Belajar {
	public static void main(String args[]){
	int a[] = {9, 20, 15, 18, 1, 3, 2, 5, 6, 8, 10, 11, 14, 13, 4, 7, 12, 17, 19, 16};
	for ( int kiri = 0; kiri < a.length - 1; kiri++ ){
		for ( int kanan = kiri + 1; kanan < a.length; kanan++ ){
			if ( a[ kanan ] > a[ kiri ] ){
				int temporary = a[ kiri ];
				a[ kiri ] = a[ kanan ];
				a[ kanan ] = temporary;
			}
		}
	}
	System.out.println(a[0]+" "+a[1]+" "+a[2]+" "+a[3]+" "+a[4]+" "+a[5]+" "+a[6]+"
"+a[7]+" "+a[8]+" "+a[9]+" "+a[10]+" "+a[11]+" "+a[12]+" "+a[13]+" "+a[14]+" "+a[15]+"
"+a[16]+" "+a[17]+" "+a[18]+" "+a[19]);
	}
}
```

```
public class Belajar {
    public static void main(String args[]){

    int a[] = {9, 20, 15, 18, 1, 3, 2, 5, 6, 8, 10, 11, 14, 13, 4, 7, 12, 17, 19, 16};

    //loop index kiri yang dibandingkan
    for ( int kiri = 0; kiri < a.length - 1; kiri++ ){
        //loop index kanan sebagai pembanding
        for ( int kanan = kiri + 1; kanan < a.length; kanan++ ){
            if ( a[ kanan ] > a[ kiri ] ){
                //swap nilai index kiri dengan index kanan
                int temporary = a[ kiri ]; //simpan nilai index kiri sementara di temporary
                a[ kiri ] = a[ kanan ]; //ubah nilai index kiri dengan index kanan
                a[ kanan ] = temporary; //ubah nilai index kanan dengan temporary
            }
        }
    }

    System.out.println(a[0]+" "+a[1]+" "+a[2]+" "+a[3]+" "+a[4]+" "+a[5]+" "+a[6]+" "+a
[7]+" "+a[8]+" "+a[9]+" "+a[10]+" "+a[11]+" "+a[12]+" "+a[13]+" "+a[14]+" "+a[15]+" "+a
[16]+" "+a[17]+" "+a[18]+" "+a[19]);

    }
}
```

Gambar G.2 Source Code User1 dan User3

```
public class Belajar {
	public static void main(String args[]){
	int a[] = {9, 20, 15, 18, 1, 3, 2, 5, 6, 8, 10, 11, 14, 13, 4, 7, 12, 17, 19, 16};
	for ( int kiri = 0; kiri < a.length - 1; kiri++ ){
		for ( int kanan = kiri + 1; kanan < a.length; kanan++ ){
			if ( a[ kanan ] > a[ kiri ] ){
				int temporary = a[ kiri ];
				a[ kiri ] = a[ kanan ];
				a[ kanan ] = temporary;
			}
		}
	}
	System.out.println(a[0]+" "+a[1]+" "+a[2]+" "+a[3]+" "+a[4]+" "+a[5]+" "+a[6]+"
"+a[7]+" "+a[8]+" "+a[9]+" "+a[10]+" "+a[11]+" "+a[12]+" "+a[13]+" "+a[14]+" "+a[15]+"
"+a[16]+" "+a[17]+" "+a[18]+" "+a[19]);
	}
}
```

```
public class Belajar {
    public static void main(String args[]){

    int a[] = {9, 20, 15, 18, 1, 3, 2, 5, 6, 8, 10, 11, 14, 13, 4, 7, 12, 17, 19, 16};

    for ( int indexKiri = 0; indexKiri < a.length - 1; indexKiri++ ){
        for ( int indexKanan = indexKiri + 1; indexKanan < a.length; indexKanan++ ){
            if ( a[ indexKanan ] > a[ indexKiri ] ){
                int temporary = a[ indexKiri ];
                a[ indexKiri ] = a[ indexKanan ];
                a[ indexKanan ] = temporary;
            }
        }
    }

    System.out.println(a[0]+" "+a[1]+" "+a[2]+" "+a[3]+" "+a[4]+" "+a[5]+" "+a[6]+" "+a
[7]+" "+a[8]+" "+a[9]+" "+a[10]+" "+a[11]+" "+a[12]+" "+a[13]+" "+a[14]+" "+a[15]+" "+a
[16]+" "+a[17]+" "+a[18]+" "+a[19]);

    }
}
```

Gambar G.3 Source Code User1 dan User4

G-4

```
public class Belajar {
	public static void main(String args[]){
	int a[] = {9, 20, 15, 18, 1, 3, 2, 5, 6, 8, 10, 11, 14, 13, 4, 7, 12, 17, 19, 16};
	for ( int kiri = 0; kiri < a.length - 1; kiri++ ){
		for ( int kanan = kiri + 1; kanan < a.length; kanan++ ){
			if ( a[ kanan ] > a[ kiri ] ){
				int temporary = a[ kiri ];
				a[ kiri ] = a[ kanan ];
				a[ kanan ] = temporary;
			}
		}
	}
	System.out.println(a[0]+" "+a[1]+" "+a[2]+" "+a[3]+" "+a[4]+" "+a[5]+" "+a[6]+"
"+a[7]+" "+a[8]+" "+a[9]+" "+a[10]+" "+a[11]+" "+a[12]+" "+a[13]+" "+a[14]+" "+a[15]+"
"+a[16]+" "+a[17]+" "+a[18]+" "+a[19]);
	}
}
```

```
public class Belajar {
    public static void main(String args[]){

    int a[] = {9, 2, 11, 1, 10, 3, 20, 5, 6, 8, 18, 15, 4, 13, 14, 7, 12, 17, 19, 16};

    //loop index kiri yang dibandingkan
    for ( int kiri = 0; kiri < a.length - 1; kiri++ ){
        //loop index kanan sebagai pembanding
        for ( int kanan = kiri + 1; kanan < a.length; kanan++ ){
            if ( a[ kanan ] > a[ kiri ] ){
                //swap nilai index kiri dengan index kanan
                int temporary = a[ kiri ]; //simpan nilai index kiri sementara di temporary
                a[ kiri ] = a[ kanan ]; //ubah nilai index kiri dengan index kanan
                a[ kanan ] = temporary; //ubah nilai index kanan dengan temporary
            }
        }
    }

    System.out.println(a[0]+" "+a[1]+" "+a[2]+" "+a[3]+" "+a[4]+" "+a[5]+" "+a[6]+" "+a
[7]+" "+a[8]+" "+a[9]+" "+a[10]+" "+a[11]+" "+a[12]+" "+a[13]+" "+a[14]+" "+a[15]+" "+a
[16]+" "+a[17]+" "+a[18]+" "+a[19]);

    }
}
```

Gambar G.4 Source Code User1 dan User5

```
public class Belajar {
	public static void main(String args[]){
	int a[] = {9, 20, 15, 18, 1, 3, 2, 5, 6, 8, 10, 11, 14, 13, 4, 7, 12, 17, 19, 16};
	for ( int kiri = 0; kiri < a.length - 1; kiri++ ){
		for ( int kanan = kiri + 1; kanan < a.length; kanan++ ){
			if ( a[ kanan ] > a[ kiri ] ){
				int temporary = a[ kiri ];
				a[ kiri ] = a[ kanan ];
				a[ kanan ] = temporary;
			}
		}
	}
	System.out.println(a[0]+" "+a[1]+" "+a[2]+" "+a[3]+" "+a[4]+" "+a[5]+" "+a[6]+"
"+a[7]+" "+a[8]+" "+a[9]+" "+a[10]+" "+a[11]+" "+a[12]+" "+a[13]+" "+a[14]+" "+a[15]+"
"+a[16]+" "+a[17]+" "+a[18]+" "+a[19]);
	}
}
```

```
public class Belajar {
    public static void main(String args[]){

    int a[] = {9, 2, 11, 1, 10, 3, 20, 5, 6, 8, 18, 15, 4, 13, 14, 7, 12, 17, 19, 16};

    for ( int indexKiri = 0; indexKiri < a.length - 1; indexKiri++ ){
        for ( int indexKanan = indexKiri + 1; indexKanan < a.length; indexKanan++ ){
            if ( a[ indexKanan ] > a[ indexKiri ] ){
                int temporary = a[ indexKiri ];
                a[ indexKiri ] = a[ indexKanan ];
                a[ indexKanan ] = temporary;
            }
        }
    }

    System.out.println(a[0]+" "+a[1]+" "+a[2]+" "+a[3]+" "+a[4]+" "+a[5]+" "+a[6]+" "+a
[7]+" "+a[8]+" "+a[9]+" "+a[10]+" "+a[11]+" "+a[12]+" "+a[13]+" "+a[14]+" "+a[15]+" "+a
[16]+" "+a[17]+" "+a[18]+" "+a[19]);

    }
}
```

Gambar G.5 Source Code User1 dan User6

G-6

```
public class Belajar {
        public static void main(String args[]){
        int a[] = {9, 20, 15, 18, 1, 3, 2, 5, 6, 8, 10, 11, 14, 13, 4, 7, 12, 17, 19, 16};
        for ( int kiri = 0; kiri < a.length - 1; kiri++ ){
                for ( int kanan = kiri + 1; kanan < a.length; kanan++ ){
                        if ( a[ kanan ] > a[ kiri ] ){
                                int temporary = a[ kiri ];
                                a[ kiri ] = a[ kanan ];
                                a[ kanan ] = temporary;
                        }
                }
        }
        System.out.println(a[0]+" "+a[1]+" "+a[2]+" "+a[3]+" "+a[4]+" "+a[5]+" "+a[6]+"
"+a[7]+" "+a[8]+" "+a[9]+" "+a[10]+" "+a[11]+" "+a[12]+" "+a[13]+" "+a[14]+" "+a[15]+"
"+a[16]+" "+a[17]+" "+a[18]+" "+a[19]);
        }
}
```

```
public class Belajar {
    public static void main(String args[]){

    int a[] = {9, 20, 15, 18, 1, 3, 2, 5, 6, 8, 10, 11, 14, 13, 4, 7, 12, 17, 19, 16};

    //loop index kiri yang dibandingkan
    for ( int indexKiri = 0; indexKiri < a.length - 1; indexKiri++ ){
        //loop index kanan sebagai pembanding
        for ( int indexKanan = indexKiri + 1; indexKanan < a.length; indexKanan++ ){
            if ( a[ indexKanan ] > a[ indexKiri ] ){
                //swap nilai index kiri dengan index kanan
                int temporary = a[ indexKiri ]; //simpan nilai index kiri sementara di temporary
                a[ indexKiri ] = a[ indexKanan ]; //ubah nilai index kiri dengan index kanan
                a[ indexKanan ] = temporary; //ubah nilai index kanan dengan temporary
            }
        }
    }

    System.out.println(a[0]+" "+a[1]+" "+a[2]+" "+a[3]+" "+a[4]+" "+a[5]+" "+a[6]+" "+a[7]+" "+a
[8]+" "+a[9]+" "+a[10]+" "+a[11]+" "+a[12]+" "+a[13]+" "+a[14]+" "+a[15]+" "+a[16]+" "+a[17]+"
"+a[18]+" "+a[19]);

    }
}
```

Gambar G.6 Source Code User1 dan User7

```
public class Belajar {
	public static void main(String args[]){
	int a[] = {9, 20, 15, 18, 1, 3, 2, 5, 6, 8, 10, 11, 14, 13, 4, 7, 12, 17, 19, 16};
	for ( int kiri = 0; kiri < a.length - 1; kiri++ ){
		for ( int kanan = kiri + 1; kanan < a.length; kanan++ ){
			if ( a[ kanan ] > a[ kiri ] ){
				int temporary = a[ kiri ];
				a[ kiri ] = a[ kanan ];
				a[ kanan ] = temporary;
			}
		}
	}
	System.out.println(a[0]+" "+a[1]+" "+a[2]+" "+a[3]+" "+a[4]+" "+a[5]+" "+a[6]+"
"+a[7]+" "+a[8]+" "+a[9]+" "+a[10]+" "+a[11]+" "+a[12]+" "+a[13]+" "+a[14]+" "+a[15]+"
"+a[16]+" "+a[17]+" "+a[18]+" "+a[19]);
	}
}
```

```
public class Belajar {
    public static void main(String args[]){

    int a[] = {9, 2, 11, 1, 10, 3, 20, 5, 6, 8, 18, 15, 4, 13, 14, 7, 12, 17, 19, 16};

    //loop index kiri yang dibandingkan
    for ( int indexKiri = 0; indexKiri < a.length - 1; indexKiri++ ){
        //loop index kanan sebagai pembanding
        for ( int indexKanan = indexKiri + 1; indexKanan < a.length; indexKanan++ ){
            if ( a[ indexKanan ] > a[ indexKiri ] ){
                //swap nilai index kiri dengan index kanan
                int temporary = a[ indexKiri ]; //simpan nilai index kiri sementara di temporary
                a[ indexKiri ] = a[ indexKanan ]; //ubah nilai index kiri dengan index kanan
                a[ indexKanan ] = temporary; //ubah nilai index kanan dengan temporary
            }
        }
    }

    System.out.println(a[0]+" "+a[1]+" "+a[2]+" "+a[3]+" "+a[4]+" "+a[5]+" "+a[6]+" "+a[7]+" "+a
[8]+" "+a[9]+" "+a[10]+" "+a[11]+" "+a[12]+" "+a[13]+" "+a[14]+" "+a[15]+" "+a[16]+" "+a[17]+"
"+a[18]+" "+a[19]);

    }
}
```

Gambar G.7 Source Code User1 dan User8

Halaman ini Sengaja Dikosongkan

Ucapan Terima Kasih

Penulis ingin mengucapkan banyak terima kasih kepada pihak yang tidak terlibat langsung dalam pengerjaan tugas akhir namun tetap memberikan doa dan dukungannya hingga penulis mampu menyelesaikan tugas akhir ini. Penulis akan menyampaikan ucapan terima kasih yang sedalam-dalamnya kepada:

1) Kedua orang tua yang selalu memberikan kasih sayang, doa, dukungan secara materil dan moril, nasihat, dan selalu menyempatkan waktu untuk mendengarkan keluh kesah selama pengerjaan tugas akhir.
2) Adik, kakak, serta anggota keluarga lainnya yang selalu mendukung dan mendokan dari jauh untuk kelancaran penulis dalam mengerjakan tugas akhir ini.
3) Seluruh Bapak dan Ibu dosen yang telah membagi banyak ilmu dan inspirasi kepada penulis. Terimakasih telah membagi ilmu-ilmu yang semoga selalu bermanfaat bagi penulis.
4) Pak *Radityo Prasetianto Wibowo, S.Kom, M.Kom*, selaku dosen wali selama perkuliahan di Jurusan Sistem Informasi. Yang dari semester satu sampai tujuh selalu memberikan nasihat dan motivasi untuk menyelesaikan kuliah.
5) Ibu*Renny Pradina, S.T, M.T*, selaku dosen wali selama perkuliahan semester delapan di Jurusan Sistem Informasi yang telah membantu urusan akademik sampai penulis lulus.
6) M. Ginanjar Pradana, Rezki Aditian, Bimo Sasongko, M. Aditya Al Rasyid, Ryco Puji Setyono, Faiz Fanani, Hudalizaman, Alfan Eko Prasetyo yang selama 4 tahun ini selalu membantu penulis menjalani proses perkuliahan dan saling bertukar pikiran ketika penulis mengalami masalah.
7) Annisa Husna yang selalu memberikan nasihat, semangat, doa, hiburan dan candaan selama

pengerjaan tugas akhir yang membantu penulis agar selalu lebih baik.

8) Aditya Dana Iswara dan Rendy Krisnanta Putra yang sejak sekolah menengah sampai sekarang saling mendukung satu sama lain untuk mencapai cita-cita. Terima kasih atas inspirasi yang selalu kalian berikan dan menjadi motivasi penulis.
9) Kabinet BEM FTIf Prestatif Bermanfaat yang berisi cerita dan tawa. Terima kasih atas kebersamaan yang selalu membuat penulis merasa bahagia sempat menjadi bagian dari kalian.
10) Seluruh keluarga BASILISK 2011, teman-teman E-Business, teman-teman PPSI, teman-teman SPK atas segala pengalaman dan cerita yang telah kita lewati bersama mulai dari awal perkuliahan hingga saat ini. Terima kasih telah membuat masa studi di kampus menjadi menarik dan tak terlupakan.
11) Seluruh staf dan karyawan di Jurusan Sistem Informasi, FTIF ITS Surabaya yang telah memberikan ilmu dan bantuan kepada penulis selama ini.

Dan masih banyak berbagai pihak yang tidak dapat penulis tuliskan namanya satu per satu. Semoga Tuhan membalas semua kebaikan yang telah dilakukan.

RIWAYAT PENULIS

Penulis lahir di Surabaya pada 18 Maret 1994. Penulis menempuh pendidikan formal mulai dari TK pada tahun 1997 sampai dengan 1999. Kemudian melanjutkan ke tingkat sekolah dasar di SDN Airlangga 6 Surabaya pada tahun 1999 sampai dengan 2005. Pada tahun 2005, penulis melanjutkan sekolah di SMPN 1 Surabaya sampai dengan tahun 2008. Pada jenjang selanjutnya, penulis melanjutkan sekolah di SMAN 2 Surabaya sampai dengan tahun 2011. Pada tahun 2011, penulis berkesempatan kuliah di Institut Teknologi Sepuluh Nopember Surabaya lewat SNMPTN Undangan jalur Bidik Misi. Selama kuliah, penulis aktif berorganisasi di Badan Eksekutif Mahasiswa Fakultas Teknologi Informasi. Penulis dapat dihubungi lewat email teguhsaasmito@gmail.com.

Halaman ini Sengaja Dikosongkan