SKRIPSI

SISTEM PENDETEKSI DINI PLAGIARISME PADA PROSES PENGAJUAN SKRIPSI MAHASISWA TEKNIK INFORMATIKA MENGGUNAKAN ALGORITMA *LEVENSHTEIN DISTANCE*

M. HENDRA FEBIAWAN
16.0504.0161

PROGRAM STUDI TEKNIK INFORMATIKA S1
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH MAGELANG

Januari, 2019

HALAMAN PENEGASAN

Skripsi ini adalah hasil karya saya sendiri, dan semua sumber baik yang dikutip maupun dirujuk telah saya nyatakan dengan benar

Nama : Muhamad Hendra Febiawan

NPM : 16.0504.0161

Magelang, 23 Januari 2019

MUHAMAD HENDRA FEBIAWAN
16.0504.0161

PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini:

Nama	**:** Muhamad Hendra Febiawan
NPM	**:** 16.0504.0161
Program Studi	**:** Teknik Informatika S1
Fakultas	**:** Teknik
Alamat	**:** Ds. Kalitidu RT 07 / RW 01 Kec. Kalitidu, Bojonegoro, Jawa Timur
Judul Skripsi	**:** SISTEM PENDETEKSI DINI PLAGIARISME PADA PROSES PENGAJUAN SKRIPSI MAHASISWA TEKNIK INFORMATIKA MENGGUNAKAN ALGORITMA *LEVENSHTEIN DISTANCE*

Dengan ini menyatakan bahwa Skripsi ini merupakan hasil karya sendiri dan bukan merupakan plagiat dari hasil karya orang lain. Dan bila di kemudian hari terbukti bahwa karya ini merupakan plagiat, maka saya bersedia menerima sanksi administrasi maupun sanksi apapun.

Demikian surat pernyataan ini saya buat dengan penuh kesadaran dan sebenarnya serta penuh tanggungjawab.

Magelang, 23 Januari 2019

MUHAMAD HENDRA FEBIAWAN
16.0504.0161

HALAMAN PENGESAHAN

SKRIPSI

SISTEM PENDETEKSI DINI PLAGIARISME
PADA PROSES PENGAJUAN SKRIPSI
MAHASISWA TEKNIK INFORMATIKA
MENGGUNAKAN ALGORITMA *LEVENSHTEIN DISTANCE*

Disusun Oleh :

MUHAMAD HENDRA FEBIAWAN

NPM. 16.0504.0161

Telah dipertahankan di depan Dewan Penguji
Pada Tanggal 23 Januari 2019

Susunan Dewan Penguji

Pembimbing I	Pembimbing II
Agus Setiawan, M. Eng.	**Ardhin Primadewi, S. Si, M. TI.**
NIDN. 0617088801	**NIDN. 0619048501**
Penguji I	Penguji II
Andi Widiyanto,S. Kom.,M. Kom	**Sunarni, M. T**
NIDN. 0623087901	**NIDN. 0620079101**

Skripsi ini telah diterima sebagai salah satu persyaratan
untuk memperoleh gelar Sarjana Komputer
Tanggal, 23 Januari 2019

Dekan

Yun Arifatul Fatimah, ST., MT., Ph.D
NIK. 987408139

KATA PENGANTAR

Puji syukur dipanjatkan kehadirat Allah SWT, karena atas segala rahmat dan hidayah-Nya sehingga dapat diselesaikannya laporan skripsi ini dengan lancar. Skripsi ini disusun sebagai salah satu syarat untuk mencapai gelar Sarjana Komputer di Program Studi S1 Teknik Informatika Fakultas Teknik Universitas Muhammadiyah Magelang. Pada kesempatan ini diucapkan terimakasih sebesar-besarnya kepada:

1. Ir. Eko Muh. Widodo, MT selaku Rektor Universitas Muhammadiyah Magelang.
2. Yun Arifatul Fatimah, S.T., M.T., Ph.D. selaku Dekan Fakultas Teknik Universitas Muhammadiyah Magelang.
3. Agus Setiawan, M.Kom. selaku Ketua Program Studi Teknik Informatika S1 Universitas Muhammadiyah Magelang.
4. Agus Setiawan, M.Kom dan Ardhin Primadewi, S. Si., M. TI. selaku Dosen Pembimbing yang telah memberikan nasehat dan bimbingan dalam penyusunan skripsi ini.
5. Seluruh Dosen Fakultas Teknik Universitas Muhammadiyah Magelang yang telah memberikan ilmu dan pengetahuan yang bermanfaat.
6. Kedua orang tua dan keluarga yang telah memberikan dukungan baik secara moril maupun materi hingga terselesaikannya skripsi ini.
7. Eka Putri Aprilia yang telah memberikan doa, dukungan dan semangatnya.
8. Teman-teman Kampus Universitas Muhammadiyah Teknik Informatika yang telah banyak membantu dalam pengujian.

9. Semua pihak yang telah membantu namun tidak dapat disebutkan namanya satu persatu. Semoga Allah membalas kebaikan semua pihak yang telah membantu dan semoga skripsi ini dapat bermanfaat bagi semua pihak.

Magelang, 23 Januari 2019

MUHAMAD HENDRA FEBIAWAN
16.0504.0161

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

INTISARI

SISTEM PENDETEKSI DINI *PLAGIARISME* PADA PROSES PENGAJUAN SKRIPSI MAHASISWA TEKNIK INFORMATIKA MENGGUNAKAN ALGORITMA *LEVENSHTEIN DISTANCE*

Oleh : Muhamad Hendra Febiawan

Pembimbing : 1. Agus Setiawan, M. Eng.

2. Ardhin Primadewi, S. Si, M. TI.

Dunia akademis di Indonesia sudah berkembang pesat. Hal ini ditandai dengan perkembangan ilmu pengetahuan dan teknologi. Dengan segala kemudahannya, teknologi membawa dampak positif dan negatif pada kehidupan. Salah satu dampak negatif yang ditimbulkan adalah plagiarisme. Fenomena plagiarisme sering terjadi pada kalangan mahasiswa. Oleh karena itu, pendeteksian plagiarisme perlu dilakukan. Penelitian ini bertujuan untuk mendeteksi kemiripan isi teks dokumen menggunakan algoritma *Levenshtein Distance*. Tipe dokumen yang digunakan adalah .pdf . Dokumen yang digunakan adalah proposal skripsi dan Naskah Publikasi. Pada uji kecepatan perhitungan, dokumen sumber yang digunakan memiliki *count words* 4405, dengan 3 dokumen pembanding yang memiliki *count words* 13465. Hal ini menghasilkan lama perhitungan selama 3,57 detik.

Kata Kunci : *Plagiarisme,* Algoritma *Levenshtein Distance*, Dokumen.

ABSTRACT

DYNAMIC PLAGIARISM DETECTION SYSTEM IN THE SUBMISSION PROCESS OF STUDENTS ESSAY INFORMATICS ENGINEERING USING ALGORITHM LEVENSHTEIN DISTANCE

By : Muhamad Hendra Febiawan

Supervisors : 1. Agus Setiawan, M. Eng.

2. Ardhin Primadewi, S. Si, M. TI.

The academic world in Indonesia has been growing rapidly. This is noticed by the development of science and technology. With all the facilities, technology has a positive and negative impact on life. One of negative impacts is plagiarism. The plagiarism often occurs among students. Therefore detection of plagiarism needs to be done. This aims to detect the similarity of the text content of the document using the Levenshtein Distance algorithm. The type of document used is .pdf . The documents used are thesis proposals and publication papers. In the calculation speed test, the source document used has count words of 4405 with 3 comparative documents that has count words of 13465. This results in a calculation duration of 3.57 seconds.

***Keywords* :** *Plagiarism, Levenshtein Distance Algorithm, Documents*.

BAB I
PENDAHULUAN

A. Latar Belakang

Di Indonesia terdapat berbagai macam perguruan tinggi baik negeri atau swasta. Menurut data dari Pangkalan Data Pendidikan Tinggi (PDDIKTI) Kementerian Riset, Teknologi dan Pendidikan Tinggi (KEMENRISTEKDIKTI, 2018), terdapat total 4.696 perguruan tinggi di Indonesia dengan rincian 1.062 Akademi, 279 Politeknik, 2.535 Sekolah Tinggi, 218 Institut, 584 Universitas dan 20 Akademi Komunitas. Data tersebut adalah total keseluruhan perguruan tinggi baik negeri dan swasta, sedangkan rincian total perguruan tinggi negeri di Indonesia adalah 436 perguruan tinggi dan total perguruan tinggi swasta di Indonesia adalah 4.260 perguruan tinggi.

Di Provinsi Jawa Tengah sendiri menurut (KEMENRISTEKDIKTI, 2018), terdapat 380 perguruan tinggi dengan rincian 38 perguruan tinggi negeri dan 342 perguruan tinggi swasta. Salah satu perguruan tinggi swasta di Provinsi Jawa Tengah adalah Universitas Muhammadiyah Magelang yang memiliki jumlah total dosen tetap berjumlah 183 Dosen dan 6.592 Mahasiswa dengan rasio dosen tetap/jumlah mahasiswa adalah 1 : 36 berdasarkan Data Pelaporan Tahun 2017/2018 menurut PDDIKTI (PDDIKTI, 2018).

Berdasarkan data diatas dunia akademis di Indonesia sudah semakin pesat berkembang yang ditandai dengan semakin banyaknya perguruan tinggi di Indonesia juga dengan perkembangan ilmu pengetahuan dan teknologi. Kemampuan mahasiswa dibidang ilmu pengetahuan dan teknologi terutama dibidang komputer menjadi salah satu faktor yang mendukung dengan berkembangnya dunia akademis di Indonesia. Dengan segala kemudahan yang ditawarkan teknologi modern akan sangat membantu proses kegiatan belajar mengajar di lingkungan kampus, salah satunya dalam penulisan karya tulis ilmiah di Perguruan Tinggi yang merupakan bagian dari tuntutan formal akademik.

Dilihat dari tujuan penulisannya, karya tulis ilmiah dibedakan menjadi dua, pertama untuk memenuhi tugas-tugas perkuliahan seperti makalah, laporan dan

lain-lain. Kedua, karya tulis ilmiah yang digunakan sebagai salah satu syarat mahasiswa untuk menyelesaikan program studi, yaitu skripsi untuk S1 dan tugas akhir untuk D3 (Panji Novantara, 2018).

Seperti di perguruan tinggi pada umumnya, skripsi di Universitas Muhammadiyah Magelang menjadi sangat penting dan merupakan bagian dari karya tulis ilmiah untuk menyelesaikan Program Sarjana S1. Skripsi merupakan kemampuan akademik mahasiswa dalam meneliti suatu kasus yang sesuai bidang keilmuannya, melalui skripsi mahasiswa dituntut untuk berpikir sistematis.

Hasil akhir dari karya tulis skripsi sendiri berbentuk Laporan Skripsi dan Naskah Publikasi. Naskah Publikasi merupakan naskah skripsi mahasiswa yang ditulis kembali dalam bentuk jurnal. Naskah publikasi ini merupakan salah satu bentuk karya tulis ilmiah mahasiswa pada kegiatan penelitian yang terbimbing oleh dosen. Penulisan naskah publikasi dapat diketahui terdapat konten penjiplakan atau *plagiarisme*.

Tahap penyimpanan data – data Skripsi pada Teknik Informatika di Universitas Muhammadiyah masih manual sehingga seorang mahasiswa memiliki peluang untuk melakukan *copy – paste* atau *plagiarisme* proposal atau laporan akhir Skripsi dari awal hingga akhir tanpa diketahui oleh jurusan atau dosen. Dengan demikian *Plagiarisme* adalah tindakan mengambil ide orang lain, mengambil tulisan orang lain, dan mengambil teks secara keseluruhan dan mengakuinya sebagai milik pribadi (Rocky Yefrenes Dillak, 2016).

Fenomena *plagiarisme* yang lebih spesifik sering terjadi pada dunia akademis, khususnya dilakukan mahasiswa. Hal ini dikarenakan kegiatan tulis menulis sering dilakukan mahasiswa untuk menyelesaikan tugas kuliah maupun tugas akhir. Mahasiswa sering berinteraksi dengan komputer, sehingga mempermudah praktik plagiat (Tudesman, 2014).

Ada dua cara untuk mengurangi tingkat *plagiarisme*, yaitu dengan mencegah dan mendeteksi. Mencegah berarti menjaga atau menghalangi agar *plagiarisme* tidak dilakukan. Usaha ini harus dilakukan sedini mungkin terutama pada sistem pendidik dan moral masyarakat (Tudesman, 2014).

Cara mendeteksi dokumen yang memiliki indikasi plagiarisme dapat dilakukan secara manual dengan cara membandingkan dua dokumen secara manual namun hal tersebut tidak efektif yang memakan banyak waktu dan tenaga. Oleh karena itu diperlukan sebuah sistem pendeteksi *plagiarisme* pada dokumen teks yang dilakukan secara kompatibel.

Salah satu metode yang dapat digunakan dalam melakukan deteksi kemiripan dokumen teks adalah dengan melakukan perhitungan dengan metode *Levenshtein Distance*. *Levenshtein Distance* sendiri merupakan salah satu algoritma *text similarity,* yaitu algoritma untuk menghitung kemiripan dua *string input* yang dibandingkan (Hamidillah Ajie, 2017).

Berdasarkan pada permasalahan diatas maka penulis merancang "Sistem Pendeteksi Dini Plagiarisme Pada Pengajuan Skripsi Mahasiswa Jurusan Teknik Informatika Menggunakan Algoritma *Levenshtein Distance*", dengan *output* dari aplikasi berupa persentase kemiripan dan perbedaan isi teks dokumen agar dapat membantu dosen mendeteksi dini tindak *plagiarisme* sehingga dapat mengurangi tingkat *plagiarisme* pada kalangan mahasiswa Teknik Informatika Universitas Muhammadiyah Magelang.

B. Rumusan Masalah

Bagaimana membuat sistem pendeteksi dini *plagiarisme* pada proses pengajuan skripsi mahasiswa Teknik Informatika di Universitas Muhammadiyah Magelang dengan algoritma *Levenshtein Distance* ?

C. Tujuan Penelitian

Untuk membuat Sistem Pendeteksi Dini *Plagiarisme* Pada Proses Pengajuan Skripsi Mahasiswa Teknik Informatika Menggunakan Algoritma *Levenshtein Distance* di Universitas Muhammadiyah Magelang.

D. Manfaat Penelitian

Manfaat yang diharapkan dari hasil penelitian ini adalah :

1. Bagi peneliti hal ini dapat membantu menambah wawasan ilmu pengetahuan baik teori maupun praktik khususnya perancangan sistem pendeteksi dini *plagiarisme* menggunakan Algoritma *Levenshtein Distance.*
2. Bagi dosen diharapkan dapat membantu dalam membimbing mahasiswa untuk mengurangi tingkat *plagiarisme* dijurusan Teknik Informatika Universitas Muhammadiyah Magelang.
3. Bagi mahasiswa diharapkan dapat mengurangi tindakan *plagiarisme* dalam penulisan karya tulis ilmiah.

BAB II
TINJAUAN PUSTAKA

A. Penelitian Relevan

1. Penelitian yang dilakukan (Dwi Susanto, 2016) berjudul "Deteksi Plagiat Dokumen Tugas Daring Laporan Praktikum Mata Kuliah Desain Web Menggunakan Metode Naive Bayes". Hasil dari penelitian tersebut adalah aplikasi pendeteksi plagiarisme yang dibuat untuk mendeteksi kesamaan antar dokumen dan menghasilkan nilai persentase plagiarisme. *Naive Bayes* digunakan sebagai metode untuk mendeteksi kesamaan antar dokumen tugas mahasiswa, dengan hal tersebut maka dapat memudahkan Dosen Jaga praktikum mahasiswa untuk memeriksa kesamaan dokumen unggahan mahasiswa.
2. Penelitian yang dilakukan (Rocky Yefrenes Dillak, 2016) berjudul "Sistem Deteksi Dini Plagiarisme Tugas Akhir Mahasiswa Menggunakan Algortima N-Grams dan Winnowing". Hasil dari penelitian tersebut adalah sistem plagiarisme yang membandingkan file dokumen yang berbeda dari Tugas Akhir Mahasiswa memberikan hasil berupa persentase *similarity* , namun semakin banyak isi sebuah file yang dideteksi, waktu prosesnya akan semakin lama.
3. Penelitian yang dilakukan (Panji Novantara, 2018) berjudul "Implementasi Algoritma Jaro-Winkler Distance Untuk Sistem Pendeteksi Plagiarisme Pada Dokumen Skripsi". Hasil dari penelitian tersebut adalah sistem pendeteksi plagiarisme yang dapat digunakan untuk mendeteksi kesamaan dokumen skripsi degan cara melakukan perbandingan antara dokumen asli dan dokumen uji yang diinputkan untuk mengetahui tingkat kemiripan (*similarity*) dari dokumen skripsi yang diuji.
4. Penelitian yang dilakukan (Na'firul Hasna Ariyani, 2016) berjudul "Aplikasi Pendeteksi Kemiripan Isi Teks Dokumen Menggunakan Metode *Levenshtein Distance*". Hasil dari penelitian tersebut adalah sistem pendeteksi kemiripan dokumen teks dengan tipe dokumen yang diuji .pdf .docx dan .txt. Dokumen yang digunakan untuk perbandingan teks ini adalah dokumen berbahasa

Indonesia. Pada pengujian data real yaitu data dokumen berplagiat dengan algoritma *Levenshtein Distance* menghasilkan nilai similarity yang tinggi yaitu diatas 77% sampai 100% dengan ketentuan proporsi nilai plagiat dibawah 40%.

5. Penelitian yang dilakukan (Mufti Ari Bianto, 2018) berjudul "Perancangan Sistem Pendeteksi Plagiarisme Terhadap Topik Penelitian Menggunakan Metode *K-Means Clustering* dan Model *Bayesian"*. Hasil dari penelitian tersebut adalah sistem pendeteksi plagiarisme dengan metode K-Means yang dapat mempartisi data ke dalam *cluster* sehingga data yang memiliki karakteristik sama dikelompokkan ke dalam satu *cluster* yang sama dan data yang mempunyai karakteristik berbeda di kelompokkan ke dalam *cluster* lain yang kemudian dicari nilai terdekat dari kemiripan antar dokumen. Hasil perhitungan menggunakan *Microsoft Excel* menampilkan sebuah persentase kemiripan.

Dari lima penelitian tersebut, dapat diambil kesimpulan bahwa terdapat perbedaan pada setiap metode yang diterapkan dalam mencari nilai kesamaan dokumen. Selain itu juga terdapat perbedaan pada platform yang digunakan sebagai sistemnya yang kebanyakan masih berbasis bahasa *java.* Sistem pendeteksi dini plagiarisme pada konten teks digital dokumen naskah publikasi dengan algoritma *Levenshtein Distance,* karena *Levenshtein Distance* sendiri merupakan salah satu algoritma *text similarity,* yaitu algoritma untuk menghitung kemiripan dua *string input* yang dibandingkan. Algoritma ini memiliki kompleksitas yang kecil, yaitu O(m*n) dimana m dan n adalah panjang dari *string input* 1 dan *string input* 2 sehingga lebih efektif dalam melakukan perhitungan dibanding algoritma *text similarity* lain, terutama untuk kasus *string* yang panjang. Diharapkan akan menjadi solusi bagi pihak dosen dan prodi dalam membimbing mahasiswa mengerjakan penelitian skripsi agar menghindari tindakan plagiarisme yang merugikan diri sendiri dan kampus.

B. Penjelasan Secara Teoritis Masing-Masing Variable

1. Algoritma Levenshtein Distance

Levenshtein Distance dibuat oleh Vladimir Levenshtein tahun 1965. Perhitungan edit distance didapatkan dari matriks yang digunakan untuk menghitung jumlah perbedaan string antara dua string. Perhitungan jarak antara dua string ini ditentukan dari jumlah minimum operasi perubahan untuk membuat string A menjadi string B. Ada 3 macam operasi utama yang dapat dilakukan oleh algoritma ini yaitu :

a. Operasi Pengubahan Karakter

Operasi pengubahan karakter merupakan operasi menukar sebuah karakter dengan karakter lain contohnya penulis menuliskan *string* 'yamg' menjadi 'yang'. Dalam kasus ini karakter 'm' diganti dengan huruf 'n'.

b. Operasi Penambahan Karakter

Operasi penambahan karakter berarti menambahkan karakter ke dalam suatu *string.* Contohnya *string* 'kepad' menjadi 'kepada', dilakukan penambahan karakter 'a' diakhir *string.* Penambahan karakter tidak hanya dilakukan diakhir kata, namun bisa ditambahkan diawal maupun disisipkan ditengah *string*.

c. Operasi Penghapusan Karakter

Operasi penghapusan karakter dilakukan untuk menghilangkan karakter dari suatu *string* . Contohnya *string* 'barur' diubah menjadi 'baru'. Pada operasi ini dilakukan penghapusan huruf 'r'.

Algortima ini berjalan mulai dari pojok kiri atas sebuah *array* dua dimensi yang telah diisi sejumlah karakter *string* awal dan *string* target dan diberikan nilai *cost.* Nilai *cost* pada ujung kanan bawah menjadi nilai *edit distance* yang menggambarkan jumlah perbedaan dua *string.*

Tabel 2. 1Tabel Matriks Perhitungan *Edit Distance*

		S	A	Y	A
	0	1	2	3	4
S	1	0	1	2	3
Y	2	1	2	1	2
A	3	2	1	2	1

Contoh dari perhitungan *levenshtein* menggunakan 2 *string* yang berbeda kemudian dihitung *edit distance* pada tabel 2.1 dapat dilihat hasil perhitungan *edit distance* antara 2 *string* 'sya' dan 'saya' adalah 1 perbedaan. Pengecekan dimulai dari iterasi awal dari kedua *string* kemudian dilakukan operasi penambahan, penyisipan dan penghapusan. Nilai *edit distance* yaitu pada ujung kanan bawah matriks. Hanya ada satu proses penyisipan yang dilakukan yaitu penyisipan karakter 'a' pada *string* 'sya' sehingga menjadi 'saya'.

2. Plagiarisme

Plagiarisme merupakan tindakan kriminal yang sering terjadi dalam dunia akademis. Plagiarisme itu sendiri berasal dari kata latin "*Plagiarus*" yang berarti penculik dan "*Plagiare*" yang berarti mencuri. Jadi, secara sederhana plagiat berarti mengambil ide, kata-kata, dan kalimat seseorang dan memposisikannya sebagai hasil karyanya sendiri atau mengunakan ide, kata-kata, dan kalimat tanpa mencantumkan sumber dimana seorangpenulis mengutipnya. (Sastroasmoro, 2007)

Jenis *plagiarisme* berdasarkan klasifikasinya diantaranya, Jenis *plagiarisme* berdasarkan aspek yang dicuri yaitu kategori *plagiarisme* ide, *plagiarisme* isi, *plagiarisme* kata, kalimat, paragraf, dan *plagiarisme* total. Klasifikasi berdasarkan sengaja atau tidaknya *plagiarisme* yaitu *plagiarisme* sengaja dan *plagiarisme* tidak sengaja. Berdasarkan pada pola *plagiarisme* yaitu *plagiarisme* kata demi kata (*word for word plagiarizing*) dan *plagiarisme mosaik*. Klasifikasi berdasarkan proporsi atau persentase kata, kalimat, paragraf yang dibajak yaitu :

a) Plagiarisme ringan, plagiarisme yang jumlah proporsi atau persentase kata, kalimat, paragraf yang dibajak tidak melebihi 30 persen (<30%).
b) Plagiarisme sedang, plagiarisme yang jumlah proporsi atau persentase kata, kalimat, paragraf yang dibajak antara 30-70%.
c) Plagiarisme berat, plagiarisme yang jumlah proporsi atau persentase kata, kalimat, paragraf yang dibajak lebih dari 70 persen (>70%).

3. Dokumen

Dokumen menurut bahasa Inggris berasal dari kata "*document*" yang mempunyai arti suatu yang tertulis atau yang tercetak dan segala benda yang memiliki berbagai keterangan dipilih untuk di disusun, di kumpulkan, di sediakan ataupun untuk disebarkan (Sora, 2018). Sedangkan menurut Kamus Umum Bahasa Indonesia menyebutkan dokumen adalah sesuatu yang tertulis atau tercetak yang dapat dipergunakan sebagai bukti atau keterangan. Dokumen merupakan salah satu hal yang sangat penting karena merupakan sumber informasi yang diperlukan oleh suatu instansi, organisasi, atau negara. Tanpa dokumen kita akan kehilangan data – data yang diperlukan untuk kegiatan kantor/organisasi masa yang akan datang.

4. Teks Mining

Text Mining memiliki definisi menambang data yang berupa teks dimana sumber data biasanya di dapatkan dari dokumen, dan tujuannya adalah mencari katakata yang dapat mewakili isi dari dokumen sehingga dapat dilakukan analisa keterhubungan antar dokumen. Text mining merupakan penerapan konsep dan teknik data mining untuk mencari pola dalam teks, yaitu proses penganalisisan teks guna menyarikan informasi yang bermanfaat untuk tujuan tertentu. Berdasarkan ketidakteraturan struktur data teks, maka proses text mining memerlukan beberapa tahap awal yang pada intinya adalah mempersiapkan agar teks dapat diubah menjadi lebih terstruktur. (Harlian, 2015)

5. Basis Data (*Database*)

Sistem basis data merupakan suatu gabungan dan juga perpaduan antara basis data (*database*) dengan suatu sistem manajemen basis data (SMBD) atau yang juga lebih sering dikenal dengan istilah *Database Management System* (DBMS) (Waliyanto, 2000). Basis data dapat diartikan sebagai kumpulan informasi yang saling berhubungan satu dengan lainnya dan tersimpan dalam komputer secara terstruktur. Basis data juga dapat digunakan sebagai media penyimpanan yang dapat diubah, ditambah, dan dihapus. Basis data merupakan komponen utama dari sistem informasi, karena dengan adanya basis data maka informasi dapat didefinisikan sesuai dengan jenisnya, dan basis data juga berguna sebagai perlindungan data, karena data hanya bisa diakses dan diubah oleh pihak yang diberikan hak tertentu.

6. *Unified Modeling Languange* (UML)

"Beberapa literature menyebutkan bahwa UML menyediakan sembilan jenis diagram, yang lain menyebutkan delapan karena ada beberapa diagram yang digabung, misanya diagram komunikasi, diagram urutan dan diagram pewaktuan digabung menjadi diagram interaksi" (Widodo, 2011). Namun demikian model-model itu dapat dikelompokkan berdasarkan sifatnya yaitu statis atau dinamis. Jenis diagram itu antara lain :

a. Diagram Statis

1) *Use Case Diagram*

UseCase Diagram adalah diagram yang mendeskripsikan interaksi antara pengguna dengan aplikasi. Kesimpulannya *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan dapat dilihat pada tabel 2.2 (Nugroho, 2015).

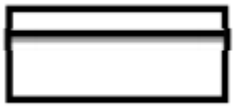
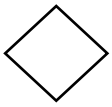
Tabel 2. 2 Simbol dan Keterangan *Use case Diagram*

Simbol	Keterangan
Aktor 	Mewakili peran orang, sistem yang lain atau alat ketika berkomunikasi dengan *use case*
UseCase 	Abstraksi dan interaksi antara sistem dan aktor
Association 	Abstraksi dari penghubung antara aktor dan *use case*
Generalisasi 	Menunjukkan spesialisasi aktor untuk dapat berpartisipasi dengan *use case*
<<*include*>> 	Menunjukkan bahwa suatu *use case* seluruhnya merupakan fungsionalitas dari *use case* lainnya
<<*extend*>> 	Menunjukkan bahwa suatu *use case* merupakan tambahan fungsional dari *use case* lainnya jika suatu kondisi terpenuhi

2) *Class Diagram*

Class Diagram merupakan diagram yang memodelkan sekumpulan kelas, interface, kolaborasi dan relasinya. Diagram kelas digambarkan dengan bentuk kotak dapat dilihat pada table dibawah ini (Nugroho, 2015).

Tabel 2. 3 Simbol dan Keterangan *Class Diagram*

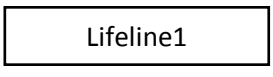
Simbol	Keterangan
Class 	Himpunan dari objek-objek yang berbagai atribut serta operasi yang sama.
Nary Association 	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
Generalization	Hubungan dimana objek anak (*descendent*) berbagai perilaku dan struktur data dari objek yang ada diatasnya objek induk (*oncestor*)
Realization 	Operasi yang benar-benar dilakukan suatu objek.

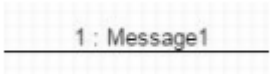
b. Diagram Dinamis

1) *Squence Diagram*

Sequence Diagram biasanya digunakan untuk tujuan analisa dan desain, memfokuskan pada identifikasi metode di dalam sebuah sistem (Nugroho, 2015). Simbol dan keterangan *Sequence Diagram* dapat dilihat pada tabel dibawah ini (Nugroho, 2015).

Tabel 2. 4 Simbol dan Keterangan Sequence Diagram

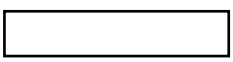
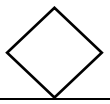
Simbol	Keterangan
Objek 	Berpartisipasi secara berurutan dengan mengirimkan atau menerima pesan dan ditempatkan di bagian atas diagram
Garis Hidup Objek	

	Menandakan kehidupan objek selama urutan dan diakhiri tanda X pada titik dimana kelas tidak lagi berinteraksi
Objek sedang aktif berinteraksi 	Persegi panjang yang sempit panjang ditempatkan diatas sebuah garis hidup dan menandakan ketika suatu onjek mengirim atau menerima pesan
Message 	Perilaku sistem yang menandai adanya suatu alur informasi atau transisi kendali antar elemen

2) *Activity Diagram*

Activity Diagram menggambarkan alur aktivitas dalam aplikasi, menjelaskan proses masing-masing alur berawal dan proses aplikasi berakhir. Diagram aktivitas juga menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi, dapat dilihat pada table dibawah ini (Nugroho, 2015).

Tabel 2. 5 Simbol dan Keterangan Activity Diagram

Simbol	Keterangan
Status Awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki status awal
Aktivitas 	Aktivitas yang dilakukan sistem, aktiv.itas biasanya diawali dengan kata kerja
Percabangan/Decision 	Asosiasi percabangan dimana jika ada pilihan aktivitass lebih dari satu.

Status Akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir

7. *Perl Hypertext Preprocessor* (PHP)

Perl Hypertext Preprocessor (PHP) adalah bahasa serverside scripting yang menyatu dengan HTML untuk membuat halaman web yang dinamis (Rudianto, 2011). Karena PHP merupakan serverside-scripting maka sintaks dan perintah-perintah PHP akan diesksekusi diserver kemudian hasilnya akan dikirimkan ke *browser* dengan format HTML.

PHP adalah sebuah bahasa pemrograman yang digunakan untuk membangun halaman web. Salah satu keunggulan dari PHP adalah kode program yang ditulis dalam PHP tidak akan terlihat oleh user, sehingga keamanan halaman web lebih terjamin. PHP dirancang untuk membuat halaman web menjadi lebih dinamis. Kelebihan lainnya dari PHP adalah bahasa pemrograman atau sebuah bahasa *script* yang tidak melakukan sebuah kompilasi dalam penggunaanya, Web Server yang mendukung PHP mudah untuk ditemukan seperti, IIS, Lighttpd, hingga Xitami dengan konfigurasi yang relatif mudah, mudah untuk dikembangkan, karena banyaknya milis-milis dan *developer* yang siap membantu dalam pengembangan, dalam sisi pemahaman, PHP adalah bahasa scripting yang paling mudah karena refrensinya sangat banyak dan mudah untuk ditemukan (Rudianto, 2011).

8. *Web Application Framework*

Web Application Framework merupakan sebuah kerangka perangkat lunak yang dirancang untuk membantu pembangunan web dinamis, aplikasi web, web services, dan web resources. Dengan menggunakan sebuah framework, proses pembangunan web akan menjadi semakin mudah, cepat dan hemat biaya. Ini dikarenakan sebagian besar *framework* telah mengimplementasikan fitur-fitur seperti *Data Persistence, Session Management, User Authentication, Security, Caching,* dan *Administrative Interface* (DocForge, 2014)..

9. Laravel

Laravel adalah salah satu *web application framework* yang bersifat open source. *Framework* ini berjalan diatas PHP 5 dan berbasis MVC (*Model View Controller*). Laravel pertama kali dirilis pada 22 Februari 2012 dan versi stabil adalah 4.2.11 yang dirilis 04 Oktober 2014 (F, 2014).

10. Arsitektur *Model View Controller (MVC)*

MVC adalah sebuah metode untuk membuat sebuah aplikasi dengan memisahkan data (Model) dari tampilan (*View*) dan cara bagaimana memprosesnya (*Controller*) (LLC, 2015). Terdapat tiga komponen pembangun suatu MVC yaitu :

a) Model

Model adalah bagian yang berperan menghubungkan *controller* dengan *database*. Tugas dari model adalah melakukan manipulasi data ke *database* seperti CRUD (*create, read, update, delet*).

b) *View*

View mengatur bagaimana data akan ditampilkan kepada user. Data yang didapat dari model akan diproses oleh *controller* kemudian oleh *view* ditampilkan ke user.

c) Controller

Controller merupakan bagian yang menjadi penghubung antara *model* dan *view*. *Controller* berfungsi memroses fungsi atau perintah dari user kemudian menentukan bagaimana aplikasi dijalankan.

C. Landasan Teori

Berdasarkan teori-teori diatas, beberapa jurnal telah membuktikan ada banyak algortima yang dapat digunakan untuk mendeteksi plagiarisme, namun penulis menggunakan algoritma *Levenshtein Distance* untuk mengukur kesamaan antara 2 string, yaitu string sumber (s) dengan string target (t) atau string pembanding. Selain itu, algoritma ini juga telah digunakan untuk *Spell Checking, Speech Recognition, DNA Alanlysis, Plagiarism Detection* dan untuk penerapan pada sistem

juga sudah dapat diterapkan melalui bahasa pemrograman PHP. Maka dari itu algoritma *Levenshtein Distance* dapat diterapkan untuk mengurangi tingkat *plagiarisme* pada sistem pendeteksi dini *plagiarisme* pada konten teks *digital* Naskah Publikasi mahasiswa di jurusan Teknik Informatika Universitas Muhammadiyah Magelang.

BAB III
ANALISIS DAN PERANCANGAN SISTEM

A. Analisis Sistem

1. Algoritma Levenshtein Distance

Levenshtein Distance sendiri merupakan salah satu algoritma *text similarity*, yaitu algoritma untuk menghitung kemiripan dua *string input* yang dibandingkan. Dalam kasus ini pengecekan *plagiarisme* memanfaatkan dokumen sumber dan dokumen target melalui tahap sebagai berikut :

a. Case Folding

Proses *Case Folding* adalah tahap mengubah semua huruf dalam dokumen menjadi huruf kecil hanya huruf a sampai z yang diterima.

b. Tokenizing

Proses *Tokenizing* yaitu proses memisahkan setiap kata yang menyusun suatu dokumen. Umumnya setiap kata teridentifikasi atau terpisahkan dengan kata lain oleh karakter spasi.

c. Operasi Pengubahan Karakter

Operasi pengubahan karakter merupakan operasi menukar sebuah karakter dengan karakter lain contohnya penulis menuliskan *string* ‘yamg’ menjadi ‘yang’. Dalam kasus ini karakter ‘m’ diganti dengan huruf ‘n’.

d. Operasi Penambahan Karakter

Operasi penambahan karakter berarti menambahkan karakter ke dalam suatu *string.* Contohnya *string* ‘kepad’ menjadi ‘kepada’, dilakukan penambahan karakter ‘a’ diakhir *string.* Penambahan karakter tidak hanya dilakukan diakhir kata, namun bisa ditambahkan diawal maupun disisipkan ditengah *string*.

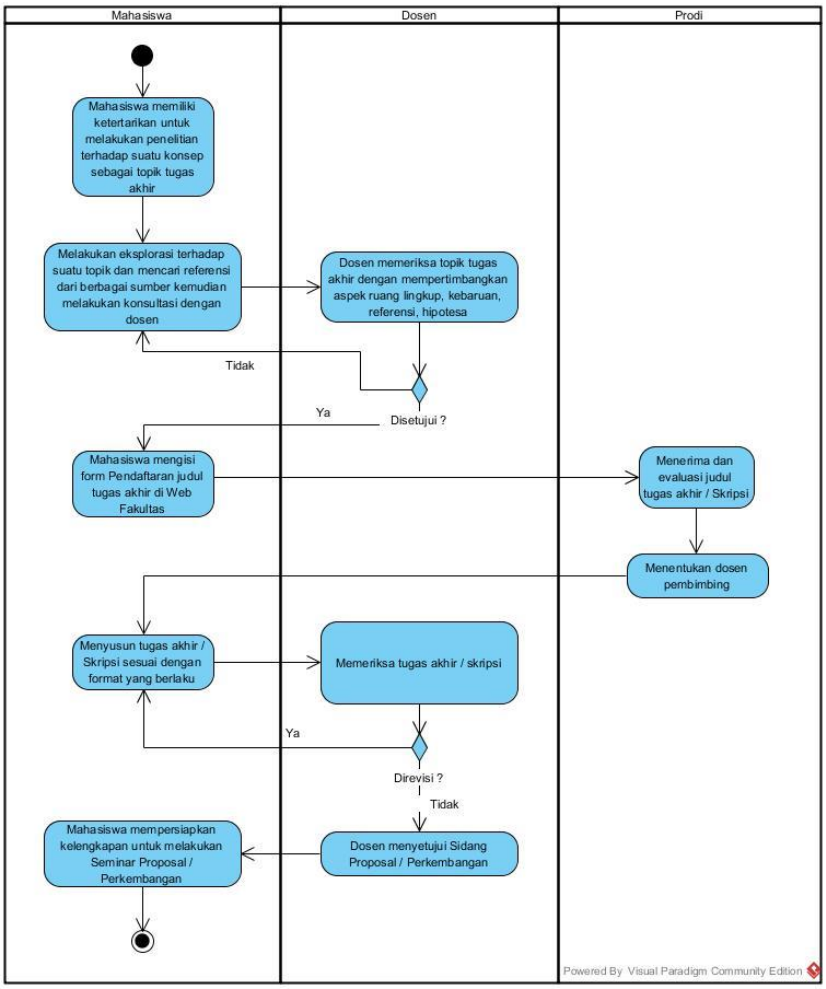
e. Operasi Penghapusan Karakter

Operasi penghapusan karakter dilakukan untuk menghilangkan karakter dari suatu *string* . Contohnya *string* ‘barur’ diubah menjadi ‘baru’. Pada operasi ini dilakukan penghapusan huruf ‘r’.

2. Sistem Yang Berjalan

a. Uraian Sistem Yang Berjalan

Alur sistem pengerjaan tugas akhir / skripsi mulai pendaftaran tugas akhir / skripsi, bimbingan dosen, sampai disetujui mengikuti sidang seminar perkembangan di Teknik Informatika masih bersifat manual tanpa ada sistem cek *plagiarisme* didalamnya, seperti yang dapat dilihat pada Gambar 3.1



Gambar 3. 1 *Flowchart* Alur Pendaftaran Sampai Seminar Proposal / Perkembangan

Gambar 3.1 menjelaskan alur sistem pengerjaan skripsi mulai dari pendaftaran sampai disetujui untuk mengikuti seminar proposal / perkembangan. Pada sistem manual, mahasiswa memiliki ketertarikan untuk melakukan penelitian terhadap suatu konsep sebagai topik skripsi. Kemudian mahasiswa melakukan eksplorasi terhadap topik yang akan diangkat dengan mencari referensi dari berbagai sumber setelah itu mahasiswa melakukan konsultasi dengan dosen. Dosen memeriksa topik yang akan diangkat dengan memperhatikan aspek ruang lingkup, kebaruan, referensi, dan hipotesa. Apabila dosen tertarik dan menyetujui dengan topik yang diangkat, maka mahasiswa dapat mendaftarkan diri untuk mengambil skripsi ke Prodi. Setelah masa pendaftaran, prodi melakukan evaluasi dan membagi dosen pembimbing sesuai topik yang diangkat masing-masing mahasiswa. Setelah dosen pembimbing dibagi, mahasiswa dapat mulai menyusun format laporan sesuai aturan baku dan melakukan bimbingan seperti biasa kepada dosen pembimbing.

b. Identifikasi Permasalahan

Berdasarkan sistem yang berjalan, diperoleh permasalahan yang ada pada sistem yang sedang berjalan. Permasalahan sebagai berikut :

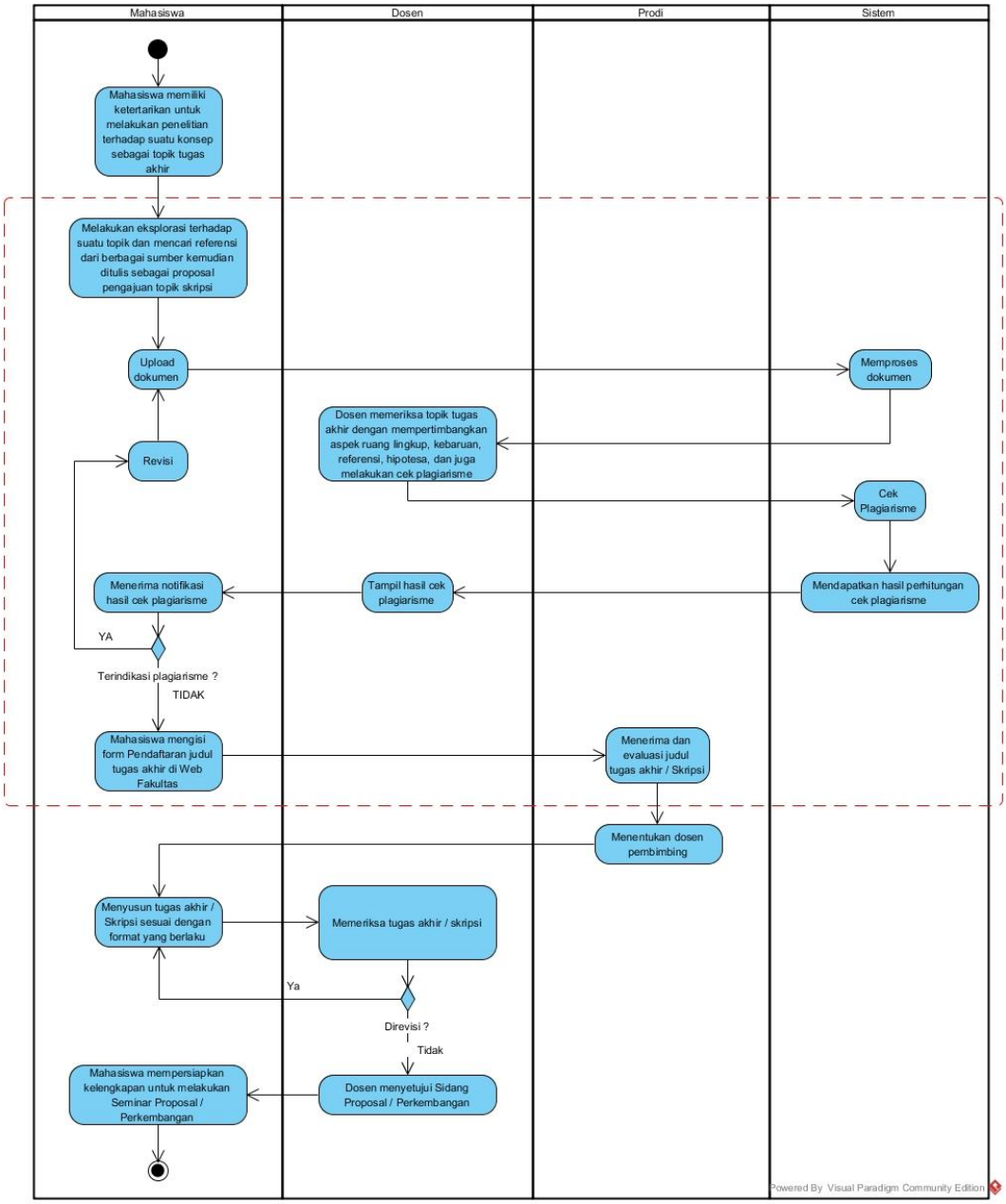
1) Setelah mahasiswa melakukan eksplorasi terhadap topik yang diangkat dengan mencari referensi dari berbagai sumber, mahasiswa melakukan konsultasi kepada dosen, kemudian dosen memeriksa topik yang akan diangkat tanpa adanya cek *plagiarisme* pada dokumen mahasiswa.
2) Dosen tidak bisa mengetahui karya ilmiah dari mahasiswa yang terindikasi *plagiarisme*.

3) Tanpa melalui cek *plagiarisme* pada saat konsultasi, indikasi penjiplakan akan terus berlanjut pada pengerjaan penulisan karya ilmiah seperti tugas akhir / skripsi.

3. Sistem Yang Diusulkan

Setelah menganalisa sistem yang berjalan di Jurusan Teknik Informatika, maka sistem yang diusulkan sebagai berikut. Pada awal pengerjaan karya ilmiah tugas akhir / skripsi mulai dari ketertarikan terhadap topik, konsultasi, pandaftaran, pembagian dosen pembimbing, penulisan, sampai bimbingan sama seperti dengan sistem yang sedang berjalan. Pada tahap bimbingan dosen tidak hanya memeriksa format penulisan, referensi, dan isi, namun dosen juga dapat melakukan cek *plagiarisme* dengan cara mahasiswa mengunggah dokumen dalam format .pdf ke dalam sistem cek *plagiarisme*, kemudian dosen membuka sistem dan memeriksa dokumen terbaru yang diunggah mahasiswa, setelah itu dosen memilih dokumen naskah publikasi milik mahasiswa yang telah melakukan sidang pendadaran dan juga telah mengunggah dokumen naskah publikasi kedalam sistem.

Fungsinya adalah untuk membandingkan dua dokumen atau lebih untuk mengecek indikasi adanya tindak *plagiarisme* pada karya ilmiah mahasiswa. Setelah sistem berhasil melakukan perhitungan dan mendapatkan hasil perhitungan cek *plagiarisme* , maka hasil ditampilkan dalam bentuk *pop up web* di halaman cek *plagiarisme* dosen, kemudian dosen mengirim notifikasi ke mahasiwa berupa hasil cek *plagiarisme*. Apabila terdeteksi *plagiarisme* , maka mahasiswa wajib melakukan revisi dan mengupload dokumen baru ke dosen agar dapat melakukan bimbingan lagi sampai tidak terdeteksi *plagiarisme*. Gambar 3.2 menjelaskan alur sistem cek *plagiarisme* yang diajukan ditunjukkan dengan garis putus – putus.



Gambar 3. 2 Flowchart Sistem yang Diajukan

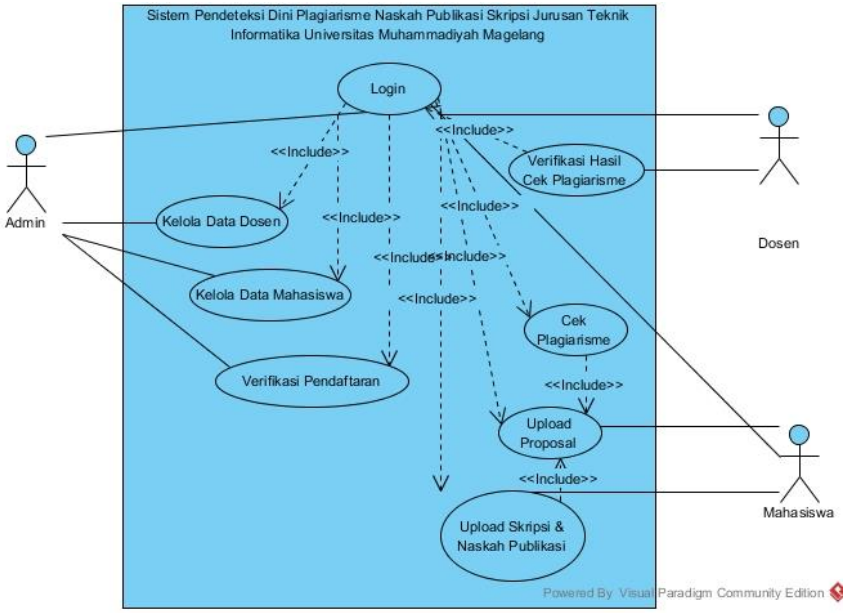
B. Perancangan Sistem

1. Perancangan UML

a. Diagram Use Case Sistem Pendeteksi Dini *Plagiarisme*

Sistem pendeteksi dini *plagiarisme* yang akan dibuat memiliki 3 aktor utama, yaitu dosen, mahasiswa, dan admin. Dosen dapat melakukan login sistem dan cek *plagiarisme*. Mahasiswa dapat melakukan *login* sistem, *upload* proposal tugas akhir / skripsi dan *upload* naskah publikasi setelah sidang

pendadaran. Admin dapat melakukan *login* sistem, mengelola data dosen, data mahasiswa, dan data bimbingan. Rancangan *Use Case* dapat dilihat pada Gambar 3.3 dibawah ini.



Gambar 3. 3 *Use Case* Sistem Pendeteksi Dini *Plagiarisme*

Tabel 3. 1 *Use Case Description* dari *Login* Admin

Use Case Name	*Login* Admin
Primary Actor	Admin
Supporting Actor	Sistem Pendeteksi Dini *Plagiarisme*
Summary	Admin *login* ke Sistem dengan memasukkan *username* dan *password* kemudian klik tombol *login.* Sistem mengecek kebenaran *username* dan *password*.
Pre-condition	Admin memiliki *password* darn *username*. Sistem Pendeteksi Dini *Plagiarisme* beroperasi.
Normal Flor of Event	1. Admin mengeksekusi sistem. 2. Aplikasi menampilkan *login dialog.* 3. Admin memasukkan *username* dan *password.* 4. Klik tombol *login.* 5. Sistem validasi *login.* 6. *Dashboard* terbuka dan menu tersedia.
Post-condition	1. *Dashboard* Sistem terbuka. 2. Menu tersedia.

Tabel 3. 2 Use Case Description Login Mahasiswa

Use Case Name	*Login* Mahasiswa
Primary Actor	Mahasiswa
Supporting Actor	Sistem Pendeteksi Dini *Plagiarisme*
Summary	Mahasiswa *login* ke Sistem dengan memasukkan *username* dan *password* kemudian klik tombol *login.* Sistem mengecek kebenaran *username* dan *password*.
Pre-condition	Mahasiswa memiliki *password* dan *username* yang diberikan Admin. Sistem Pendeteksi Dini *Plagiarisme* beroperasi.
Normal Flor of Event	1. Mahasiswa mengeksekusi sistem. 2. Aplikasi menampilkan *login dialog.* 3. Mahasiswa memasukkan *username* dan *password.* 4. Klik tombol *login.* 5. Sistem validasi *login.* 6. *Dashboard* terbuka dan menu tersedia.
Post-condition	*1.* *Dashboard* Sistem terbuka. *2.* Menu tersedia.

Tabel 3. 3 Use Case Description Login Dosen

Use Case Name	*Login* Dosen
Primary Actor	Dosen
Supporting Actor	Sistem Pendeteksi Dini *Plagiarisme*
Summary	Dosen *login* ke Sistem dengan memasukkan *username* dan *password* kemudian klik tombol *login.* Sistem mengecek kebenaran *username* dan *password*.
Pre-condition	Dosen memiliki *password* darn *username*. Sistem Pendeteksi Dini *Plagiarisme* beroperasi.
Normal Flor of Event	1. Dosen mengeksekusi sistem. 2. Aplikasi menampilkan *login dialog.* 3. Dosen memasukkan *username* dan *password.* 4. Klik tombol *login.* 5. Sistem validasi *login.* 6. *Dashboard* terbuka dan menu tersedia.
Post-condition	1. *Dashboard* Sistem terbuka. 2. Menu tersedia.

Tabel 3. 4 Use Case Description Mahasiswa Upload Proposal Skripsi

Use Case Name	Mahasiswa *Upload* Proposal Skripsi
Primary Actor	Mahasiswa
Supporting Actor	Sistem Pendeteksi Dini *Plagiarisme*
Summary	Mahasiswa memilih menu Dokumen. *Layout* terbuka dan tersedia tombol tambah.
Pre-condition	Mahasiswa memiliki dokumen proposal skripsi dengan format .pdf (Minimal BAB 1) Sistem Pendeteksi Dini *Plagiarisme* beroperasi.
Normal Flor of Event	1. Mahasiswa mengeksekusi sistem. 2. Aplikasi menampilkan *layout* Dokumen. 3. Tampil *button* Tambah. 4. Klik *button* Tambah. 5. Tampil Modal untuk *Upload* dokumen. 6. Tersedia *button* browse *file.* 7. Klik Simpan. 8. Tampil hasil dokumen yang di*upload.*
Post-condition	1. *Layout* menu Dokumen terbuka. 2. Menu tersedia.

Tabel 3. 5 Use Case Description Cek Plagiarisme

Use Case Name	Cek *Plagiarisme*
Primary Actor	Dosen
Supporting Actor	Sistem Pendeteksi Dini *Plagiarisme*
Summary	Dosen memilih menu Cek *Similarity.* *Layout* terbuka dan tampil daftar dokumen proposal skripsi yang di*upload* mahasiswa.
Pre-condition	Daftar dokumen yang akan dicek telah di*upload.* Sistem Pendeteksi Dini *Plagiarisme* beroperasi.
Normal Flor of Event	1. Dosen mengeksekusi sistem. 2. Aplikasi menampilkan daftar dokumen yang akan dicek. 3. Dosen memilih dokumen yang akan dicek. 7. Klik tombol cek. 8. Tampil Modal informasi dokumen sumber dan daftar dokumen pembanding. 9. Klik tombol cek *plagiarisme*. 10. Sistem melakukan perhitungan. 11. Tampil persentase hasil. 12. Tersedia tombol Terima dan Tolak 13. Klik Tombol Terima, mahasiswa bisa daftar skripsi. 14. Klik Tombol Tolak, mahasiswa revisi dokumen.
Post-condition	*1.* *Dashboard* Sistem terbuka. *2.* Menu tersedia.

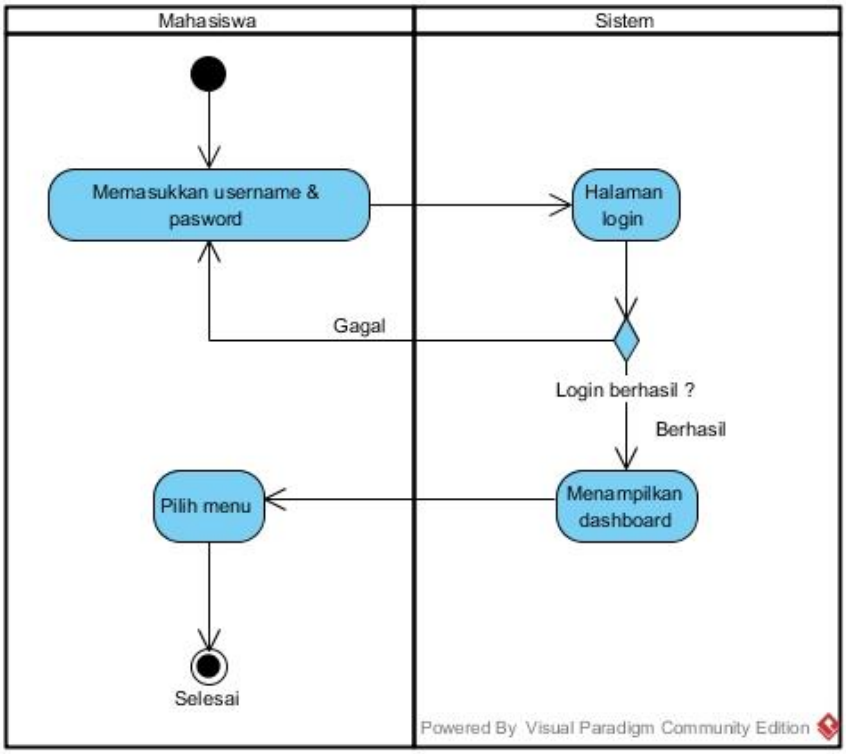
Tabel 3. 6 *Use Case Description* Mahasiswa *Upload* Naskah Publikasi

Use Case Name	Mahasiswa *Upload* Naskah Publikasi
Primary Actor	Mahasiswa
Supporting Actor	Sistem Pendeteksi Dini *Plagiarisme*
Summary	Mahasiswa memilih menu Pendaftaran. *Layout* terbuka dan tersedia tombol *Upload* dokumen. Sudah melakukan Sidang Pendadaran. Sudah diverifikasi Admin.
Pre-condition	Sudah terverifikasi oleh Admin Mahasiswa memiliki dokumen Naskah Publikasi dengan format .pdf. Sistem Pendeteksi Dini *Plagiarisme* beroperasi.
Normal Flor of Event	1. Mahasiswa mengeksekusi sistem. 2. Aplikasi menampilkan *layout Upload* dokumen Naskah Publikasi. 3. Tampil *button* Tambah. 4. Klik *button* Tambah. 5. Tampil Modal untuk *Upload* dokumen. 6. Tersedia *button* browse *file.* 7. Klik Simpan. 8. Tampil hasil dokumen yang di*upload.*
Post-condition	*1.* *Layout* menu Pendaftaran terbuka. *2.* Menu tersedia.

b. Diagram Activity Sistem Pendeteksi Dini *Plagiarisme*

1) Diagram Activity *Login* Mahasiswa

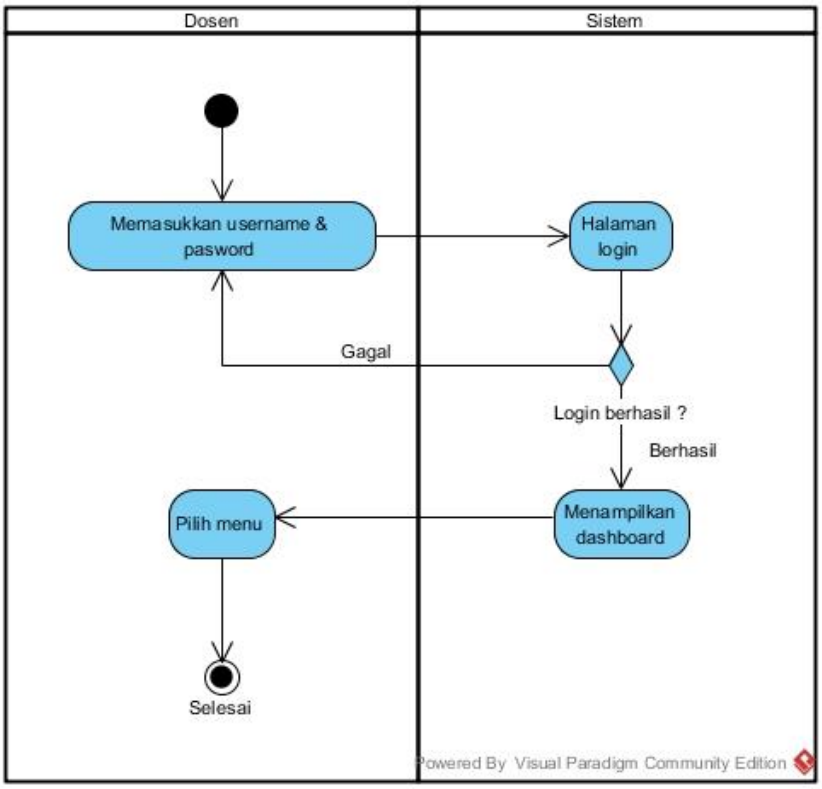
Pada Gambar 3.4 dibawah ini dijelaskan urutan tahap *login* sistem dari mahasiswa. Mahasiwa membuka web sistem kemudian memasukkan *username* dan *password*, setelah itu sistem akan melakukan cek validasi apakah data yang dimasukkan sudah benar, apabila gagal *login* maka mahasiswa harus melakukan *input* data lagi sampai benar. Apabila *login* berhasil akan muncul *dashboard* dan mahasiswa dapat memilih menu yang tersedia.



Gambar 3. 4 Activity Diagram Login Mahasiswa

2) Diagram Activity *Login* Dosen

Pada Gambar 3.5 dibawah ini dijelaskan urutan tahap *login* sistem dari dosen. Dosen membuka web sistem kemudian memasukkan *username* dan *password*, setelah itu sistem akan melakukan cek validasi apakah data yang dimasukkan sudah benar, apabila gagal *login* maka dosen harus melakukan *input* data lagi sampai benar. Apabila *login* berhasil akan muncul *dashboard* dan dosen dapat memilih menu yang tersedia.

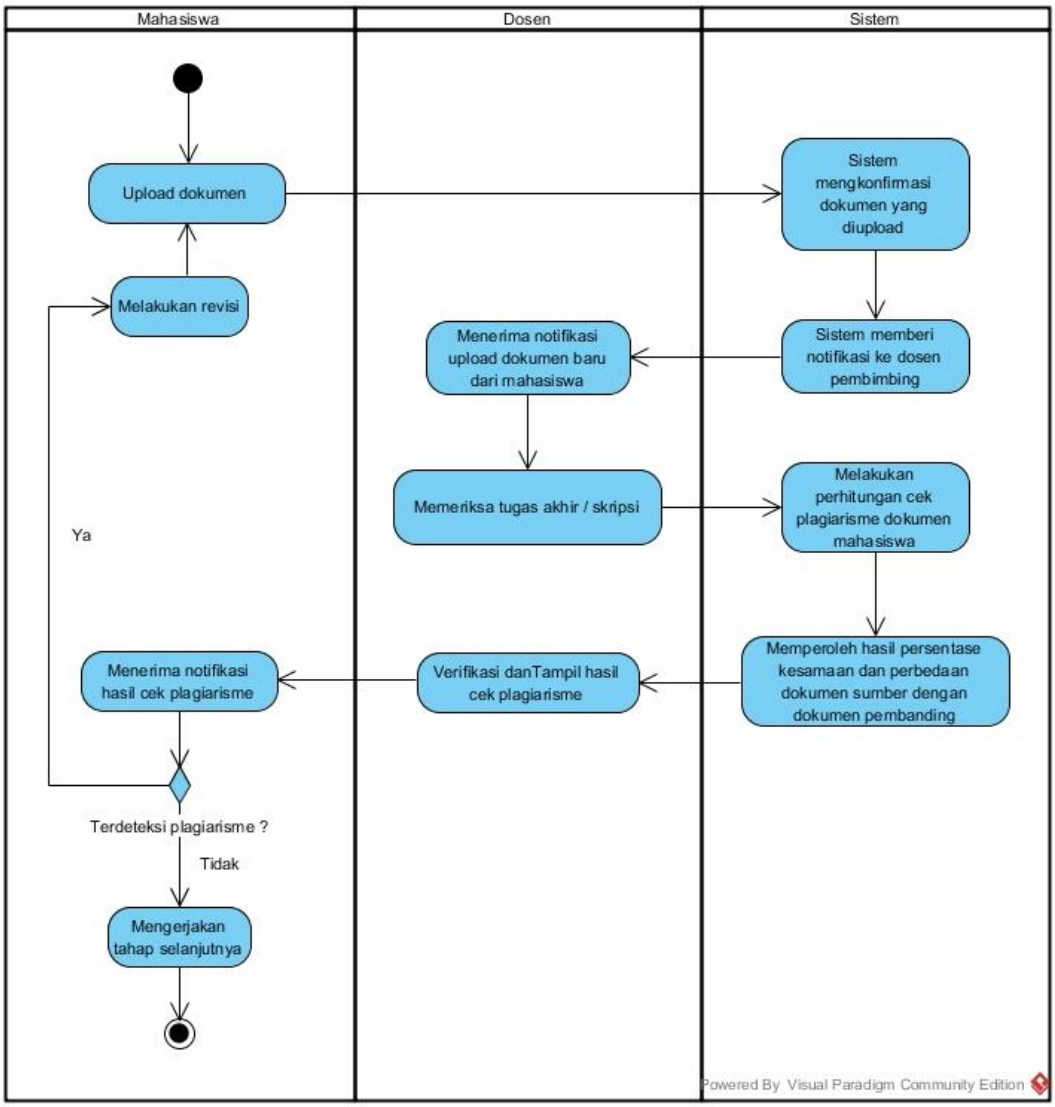


Gambar 3. 5 Activity Diagram Login Dosen

3) Diagram Activity Cek *Plagiarisme*

Diagram activity cek *plagiarisme* dilakukan oleh dua aktor utama dan dijalankan oleh sistem. Pertama setelah mahasiswa melakukan *login*, mahasiswa dapat memilih menu *upload document*. Setelah itu dosen akan mendapat notifikasi bahwa terdapat dokumen baru yang diunggah oleh mahasiswa. Kemudian dosen akan memeriksa dokumen baru yang diunggah, tahap selanjutnya dosen akan melakukan cek *plagiarisme* dengan dokumen sumber dari proposal tugas akhir / skripsi mahasiswa dan mencari dokumen pembanding berupa naskah publikasi yang mungkin menurut calon dosen pembimbing memiliki kemiripan dengan dokumen sumber. Tahap selanjutnya sistem akan melakukan proses cek *plagiarisme*, setelah sistem mendapatkan hasil maka akan muncul *pop up* dihalaman web dosen pembimbing. Kemudian

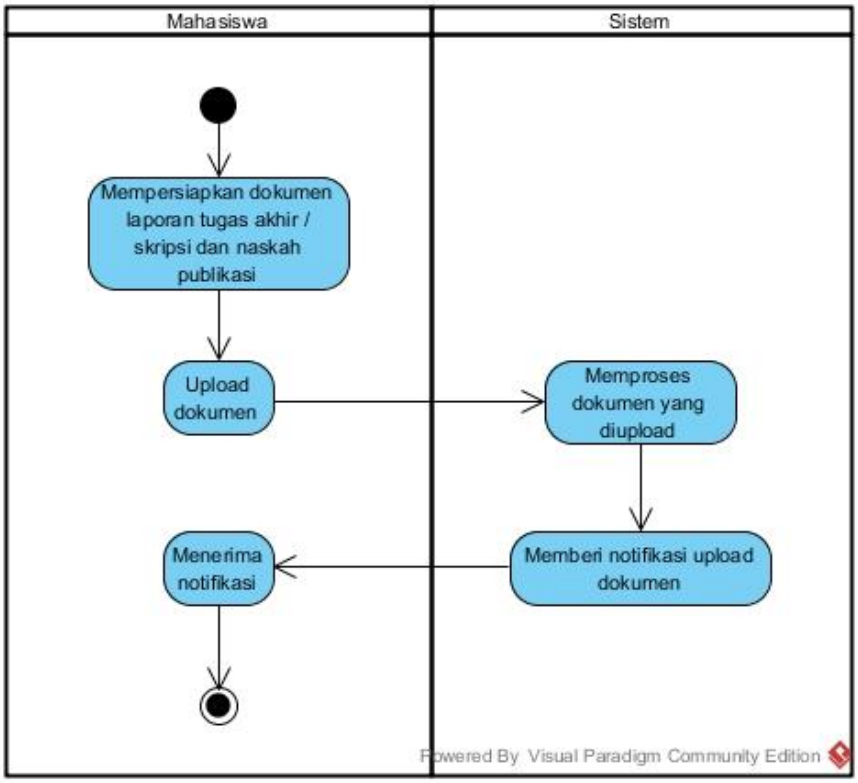
dosen akan mengirim notifikasi ke mahasiswa mengenai hasil cek *plagiarisme.* Apabila dokumen terdeteksi *plagiarisme*, maka mahasiswa wajib melakukan revisi dan mengulangi tahap sebelumnya sampai tidak terdeteksi *plagiarisme*. Jika dokumen tidak terdeteksi *plagiarisme,* maka mahasiwa dapat melanjutkan ke tahap berikutnya seperti ditunjukkan pada proses Gambar 3.6 dibawah ini.



Gambar 3. 6 *Activity Diagram* Cek *Plagiarisme*

4) Diagram Activity *Upload* dokumen Naskah Publikasi

Diagram activity *upload* dokumen naskah publikasi dilakukan oleh mahasiswa, namun mahasiswa yang boleh melakukan *upload* dokumen naskah publikasi adalah mahasiswa yang sudah melakukan sidang pendadaran dan sudah menyelesaikan revisi. Seperti tahap sebelumnya, mahasiswa melakukan *login* sistem kemudian memilih menu untuk *upload* dokumen naskah publikasi. Selenjutnya sistem akan memproses dokumen dan disimpan pada database dan mahasiswa mendapat notifikasi seperti pada Gambar 3.7 dibawah.

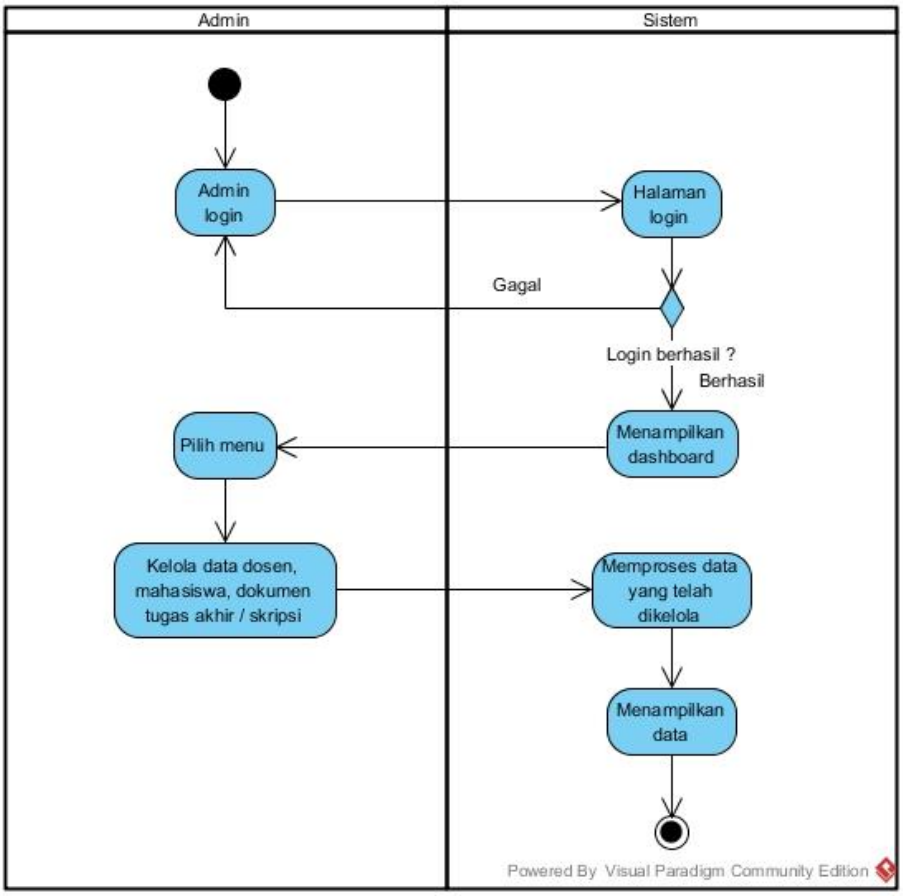


Gambar 3. 7 *Activity Diagram Upload* Naskah Publikasi

5) Diagram Activity Admin Kelola Data

Pada Gambar 3.8 dibawah dijelaskan mengenai diagram activity admin kelola data. Admin pada sistem ini dapat mengelola data mahasiswa, data dosen, data bimbingan. Alur pertama yang dilakukan admin adalah melakukan *login*, apabila *login* berhasil maka akan tampil halaman *dashboard*,

apabila gagal admin akan diarahkan ke halaman *login* lagi. Setelah masuk *dashboard*, admin dapat memilih menu kelola data.



Gambar 3. 8 *Activity Diagram* Admin Kelola Data

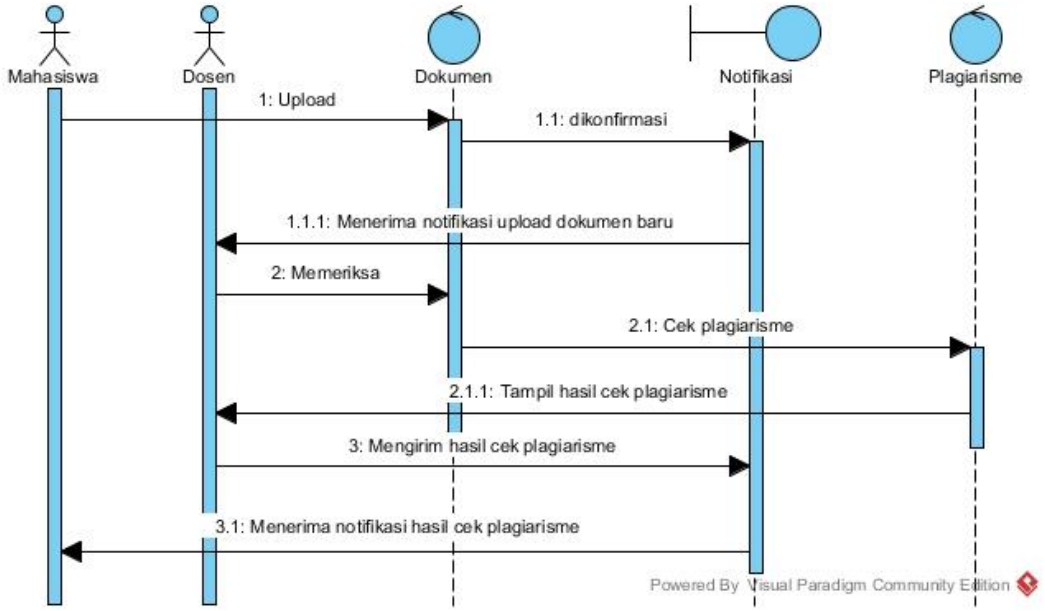
c. *Sequence Diagram* Sistem Pendeteksi Dini *Plagiarisme*

Sequence Diagram digunakan untuk memberikan rangkaian pesan antar objek pada aktivitas tertentu yang selanjutnya akan dijalankan oleh sistem. Pada sistem ini terdapat 3 *Sequence Diagram* , yaitu sebagai berikut.

1) *Sequence Diagram* Cek *Plagiarisme*

Cek *plagiarisme* dilakukan oleh dua aktor, mahasiswa dan dosen. Mahasiswa melakukan *upload* dokumen ke dosen melalui sistem pendeteksi dini *plagiarisme* dokumen akan diproses sistem dan dosen mendapatkan notifikasi bahwa terdapat dokumen baru yang di *upload* mahasiswa.

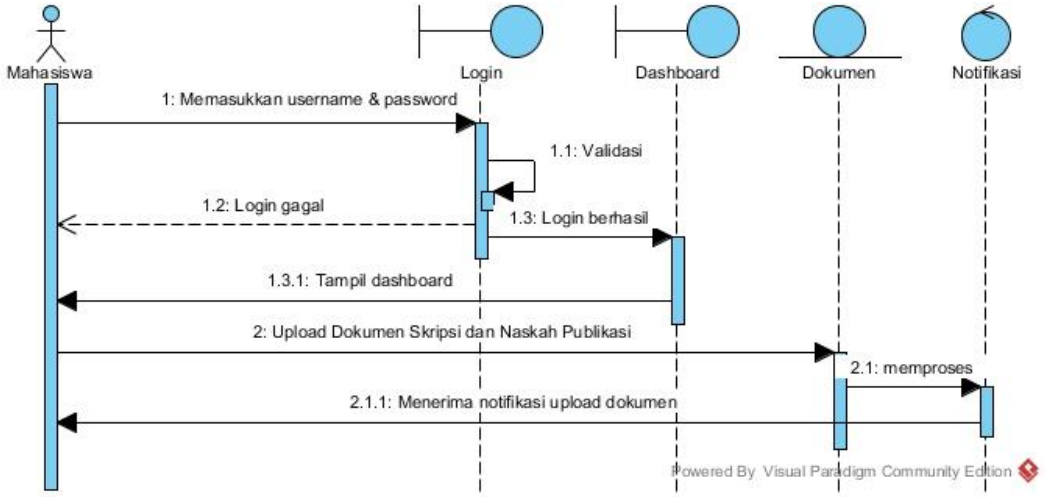
Kemudian dosen memeriksa dokumen dan melakukan cek *plagiarisme*. Sistem melakukan perhitungan cek *plagiarisme*, setelah mendapatkan hasil, selanjutnya hasil akan ditampilkan melalui *pop up* dihalaman dosen dan dosen memberi notifikasi hasil ke mahasiswa. *Sequence diagram* cek *plagiarisme* tersaji pada Gambar 3.9 seperti berikut.



Gambar 3. 9 *Sequence Diagram* Cek *Plagiarisme*

2) Sequence Diagram Upload Naskah Publikasi

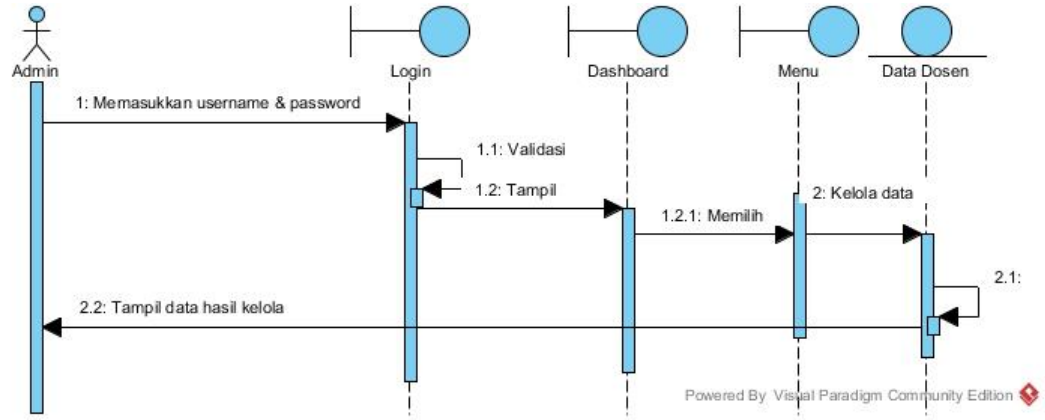
Naskah publikasi di*upload* oleh mahasiswa yang sudah mengikuti sidang pendadaran dan sudah melakukan revisi. Mahasiswa melakukan *login* sistem, apabila *login* gagal maka mahasiswa akan ditampilkan lagi halaman *login* dan jika *login* berhasil maka akan tampil halaman *dashboard* selanjutnya mahasiswa dapat memilih menu *upload* dokumen Naskah Publikasi. Alur *Sequence Diagram* dapat dilihat seperti Gambar 3.10 dibawah ini.



Gambar 3. 10 *Sequence Diagram Upload* Dokumen Naskah Publikasi

3) *Sequence Diagram* Admin Kelola Data Dosen

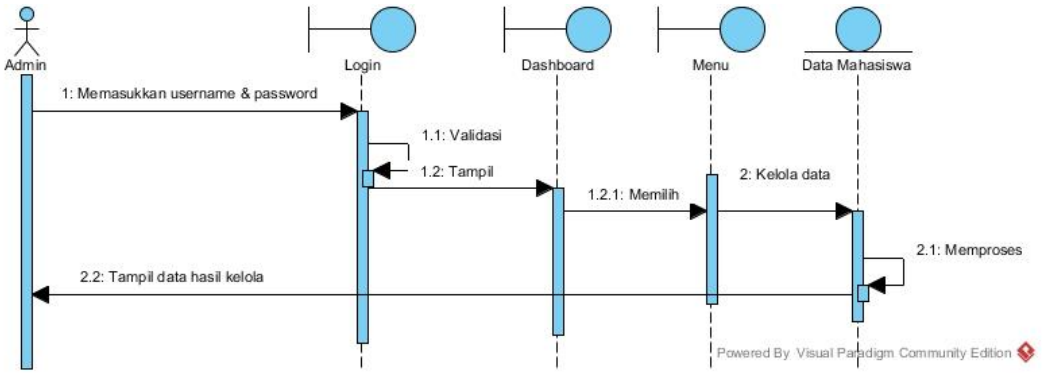
Data dosen dikelola oleh admin didalam sistem. Admin dapat melakukan *login*, kemudian bila berhasil admin akan ditampilka *dashboard* admin, selanjutnya admin dapat memilih menu untuk mengelola data. Alur diagram disajikan seperti pada Gambar 3.11 berikut



Gambar 3. 11 *Sequence Diagram* Admin Kelola Data Dosen

4) *Sequence Diagram* Admin Kelola Data Mahasiswa

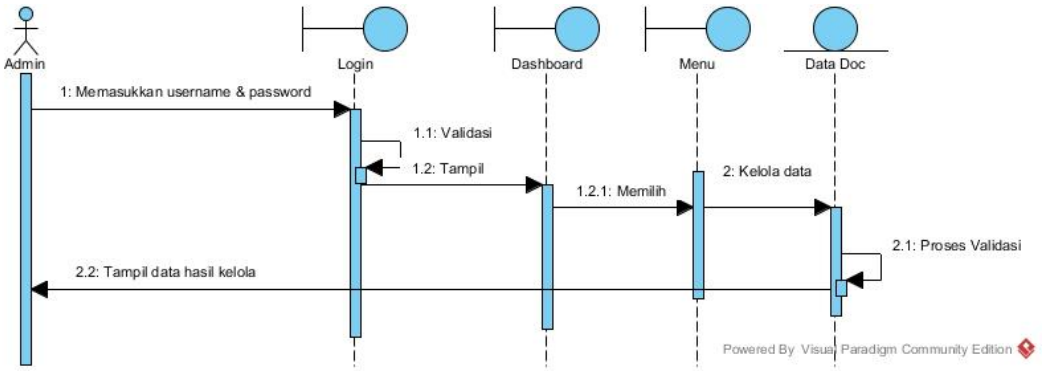
Data mahasiswa dikelola oleh admin didalam sistem. Admin dapat melakukan *login*, kemudian bila berhasil admin akan ditampilkan *dashboard* admin, selanjutnya admin dapat memilih menu untuk mengelola data. Alur diagram disajikan seperti pada Gambar 3.12 berikut



Gambar 3. 12 *Sequence Diagram* Admin Kelola Data Mahasiswa

5) *Sequence Diagram* Admin Kelola Data Dokumen

Data dokumen dikelola oleh admin didalam sistem. Admin dapat melakukan *login*, kemudian bila berhasil admin akan ditampilka *dashboard* admin, selanjutnya admin dapat memilih menu untuk mengelola data. Alur diagram disajikan seperti pada Gambar 3.13 berikut

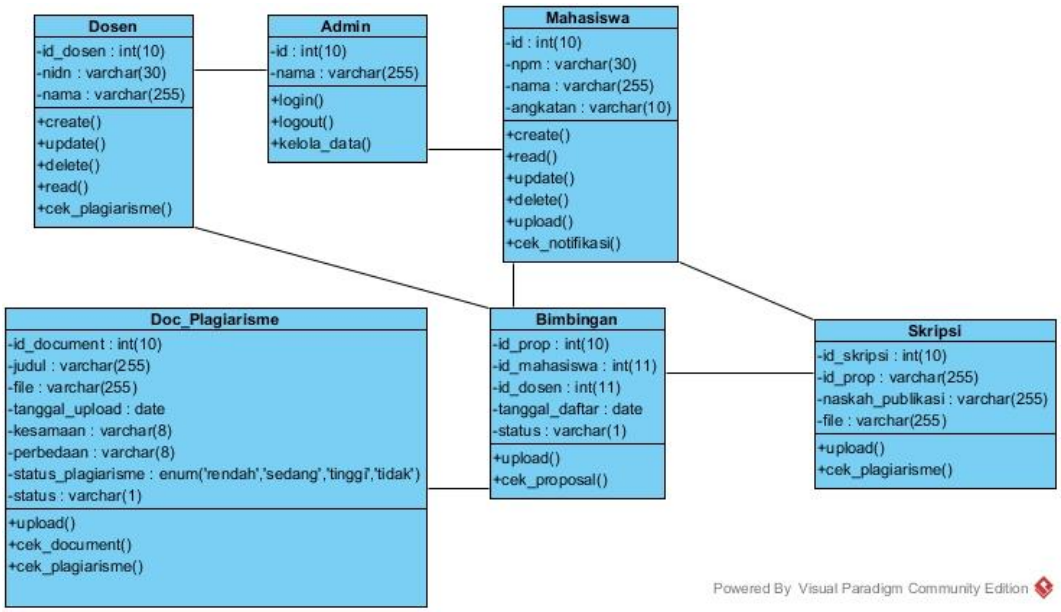


Gambar 3. 13 *Sequence Diagram* Admin Kelola Data Dokumen

d. *Class Diagram* Sistem Pendeteksi Dini *Plagiarisme*

Class Diagram menggambarkan struktur dan deskripsi *class, package*, dan objek berserta hubungan satu sama lain. Pada sistem ini terdapat 6 *class*, yaitu Admin, Mahasiswa, Dosen, Bimbingan,

Doc_Plagiarisme, dan Skripsi. *Class Diagram* pada sistem ini dapat dilihat pada Gambar 3.14 .



Gambar 3. 14 *Class Diagram* Sistem Pendeteksi Dini Plagiarisme

C. Pemodelan Data

Tahap pemodelan data ini meliputi deskripsi data objek, ERD, relasi dan kardinalitas.

1. Deskripsi Data Objek

Deskripsi Data Objek merupakan penjabaran entitas beserta atribut yang ada pada rancangan basis data dari sistem yang dibuat.

Tabel 3. 7 Admin

Admin	
Id	Kode unik dari admin
Nama	Nama dari admin

Tabel 3. 8 Mahasiswa

Mahasiswa	
Id_mhs	Kode unik setiap mahasiswa
NPM	Nomer Pokok Mahasiswa
Nama	Nama dari mahasiswa
Angkatan	Tahun angkatan mahasiswa

Tabel 3. 9 Dosen

Dosen	
Id_dosen	Kode unik tiap dosen
NIDN	Nomer Induk Dosen
Nama	Nama dari dosen

Tabel 3. 10 Bimbingan

Bimbingan	
Id_doct	Kode unik dari dokumen bimbingan
Tanggal_daftar	Tanggal pertama kali bimbingan
Status	Status pada sistem untuk mengetahui valid tidaknya data bimbingan

Tabel 3. 11 Doc_Plagiarisme

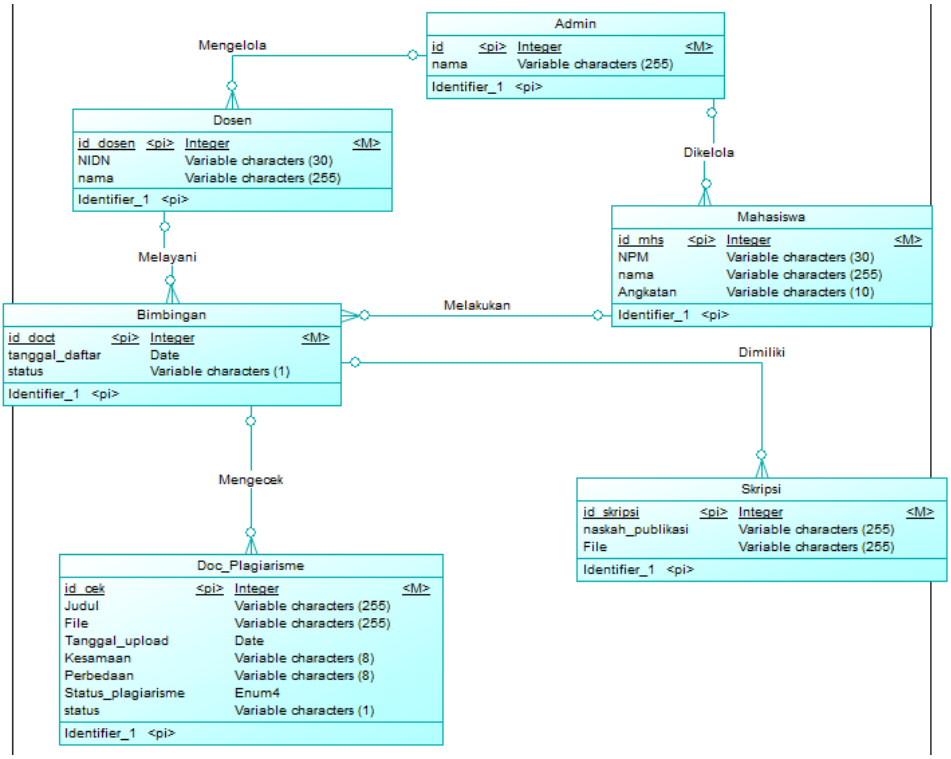
Doc_Plagiarisme	
Id_cek	Kode unik setiap melakukan cek plagiarisme
Judul	Judul proposal tugas akhir / skripsi
File	*Field* untuk menyimpan file
Tanggal_upload	Tanggal pada saat *upload*
Kesamaan	Persentase hasil kesamaan dokumen pada perhitungan cek *plagiarisme*
Perbedaan	Persentase hasil perbedaan dokumen pada perhitungan cek *plagiarisme*
Status_plagiarisme	Hasil yang menyatakan status *plagiarisme* ‘ringan’,’sedang’,’berat’,’tidak’

Tabel 3. 12 Skripsi

Skripsi	
Id_skripsi	Kode unik dokumen skripsi
Naskah_Publikasi	Judul dokumen
File	*Field* untuk menyimpan file

2. EER (*Enhanced Entity Relationship*) *Diagram*

EER (*Enhanced Entity Relationship*) *Diagram* adalah ERD yang diberikan kemampuan untuk memiliki *supertype*, *subtype* dan *instance*. EER pada sistem ini disajikan pada Gambar 3.15.



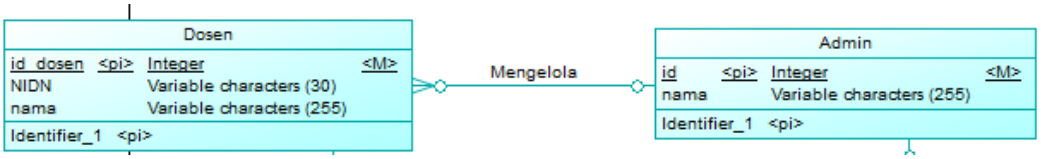
Gambar 3. 15 EER Sistem Pendeteksi Dini Plagiarisme

3. Relasi dan Kardinalitas

Relasi dan kardinalitas merupakan tahapan untuk memetakan model deskripsi data objek ke model basis data relasional.

a. Relasi Dosen dan Admin

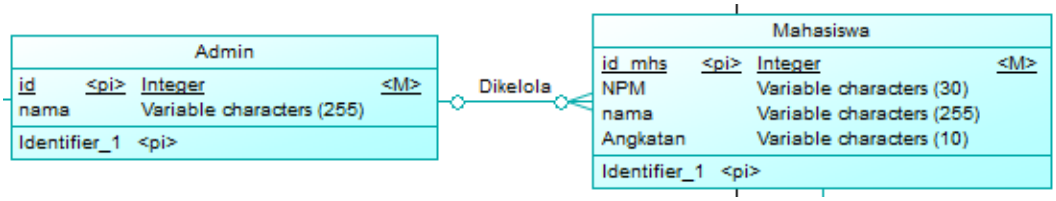
Entitas Dosen dengen admin memiliki hubungan 1:N karena satu admin dapat mengelola data dari beberapa dosen. Relasi disajikan pada gambar 3.16 berikut.



Gambar 3. 16 Relasi Dosen dan Admin

b. Relasi Mahasiswa dan Admin

Entitas mahasiswa dengan admin memiliki hubungan 1:N karena satu admin dapat mengelola data dari beberapa mahasiswa. Relasi disajikan pada Gambar 3.17 dibawah ini.



Gambar 3. 17 Relasi Mahasiswa dan Admin

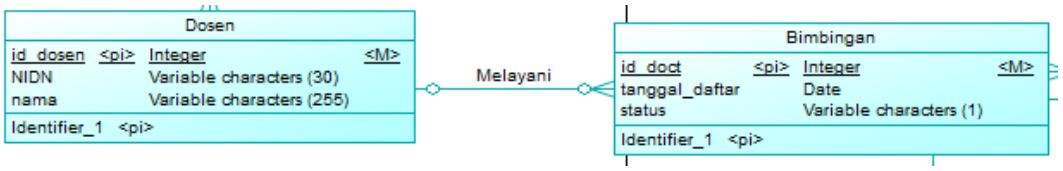
c. Relasi Mahasiswa dan Bimbingan

Entitas mahasiswa dengan bimbingan memiliki hubungan 1:N, karena satu mahasiswa dapat melakukan beberapa kali bimbingan. Relasi disajikan pada Gambar 3.18 dibawah ini.

Gambar 3. 18 Mahasiswa dan Bimbingan

d. Relasi Dosen dan Bimbingan

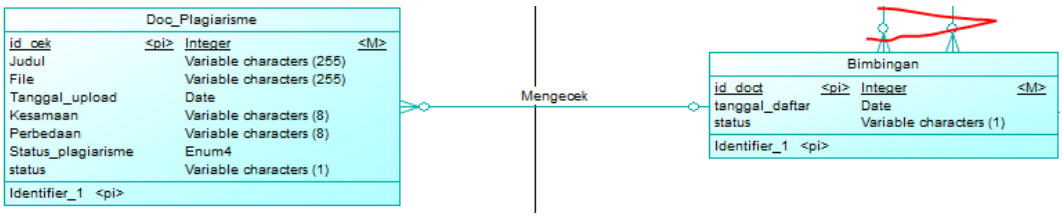
Entitas dosen dengan bimbingan memiliki hubungan 1:N, karena satu dosen dapat melayani beberapa kali bimbingan. Relasi disajikan pada Gambar 3.19 berikut.



Gambar 3. 19 Relasi Dosen dan Bimbingan

e. Relasi Bimbingan dan Doc_plagiarisme

Entitas Bimbingan dengan Doc_plagiarisme memiliki hubungan 1:N, karena pada satu bimbingan dapat dilakukan pengecekan *plagiarisme* beberapa kali. Relasi disajikan pada Gambar 3.20 berikut.



Gambar 3. 20 Relasi Bimbingan dan Doc_plagiarisme

f. Relasi Bimbingan dan Skripsi

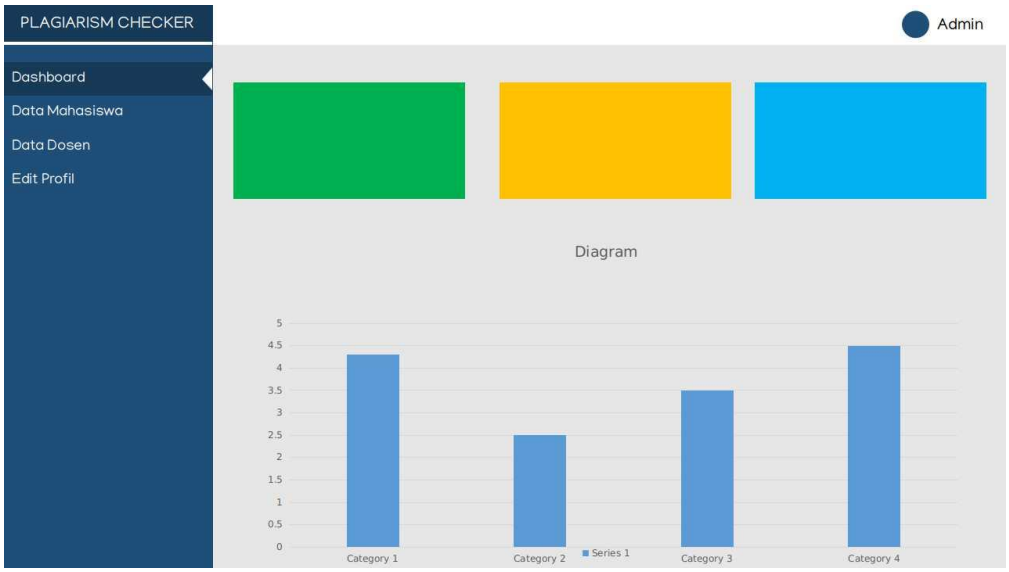
Entitas Bimbingan dengan skripsi memiliki hubungan 1:N, karena beberapa dokumen skripsi dimiliki pada saat satu bimbingan dan untuk mengetahui dokumen naskah publikasi yang d*iupload* dimiliki oleh mahasiswa siapa dan bimbingan dengan dosen siapa. Relasi disajikan pada Gambar 3.21.

Gambar 3. 21 Relasi Bimbingan dan Skripsi

D. Perancangan *Interface*

1. Rancangan Halaman *Dashboard* Admin

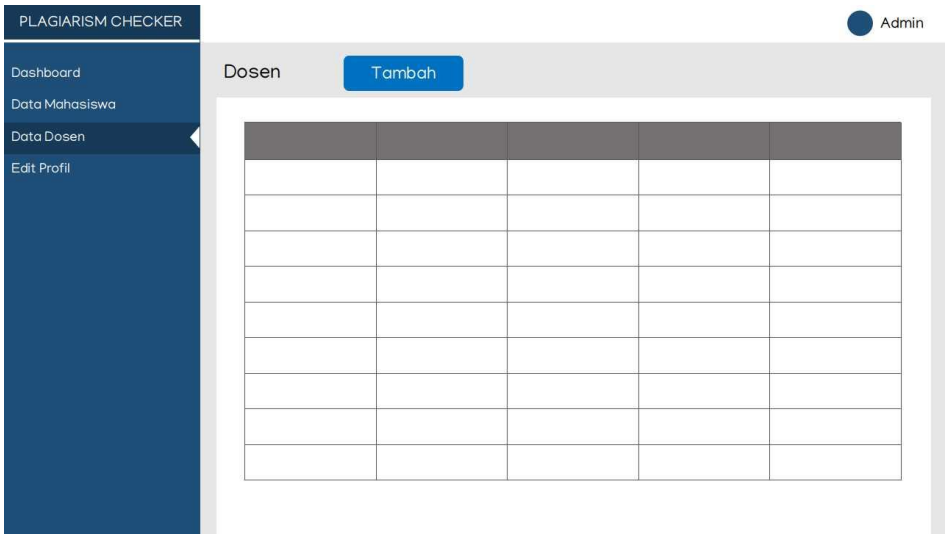
Halaman *dashboard* admin digunakan sebagai halaman utama admin untuk memonitoring data dosen, data mahasiswa, dan data bimbingan dapat dilihat pada Gambar 3.22.



Gambar 3. 22 Rancangan Halaman Dashboard Admin

2. Rancangan Halaman Data Dosen Pada Admin

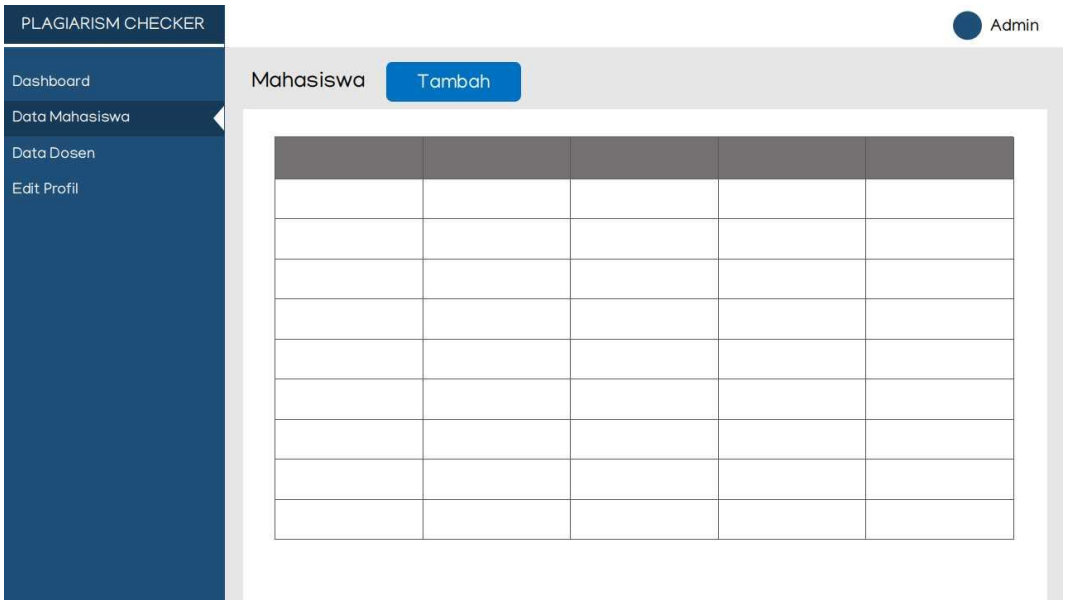
Halaman data dosen menampilkan data dosen yang sedang membimbing mahasiswa pada tahun ajaran tersebut. Dapat diliha pada Gambae 3.23



Gambar 3. 23 Rancangan Halaman Data Dosen Pada Admin

3. Halaman Data Mahasiswa Pada Admin

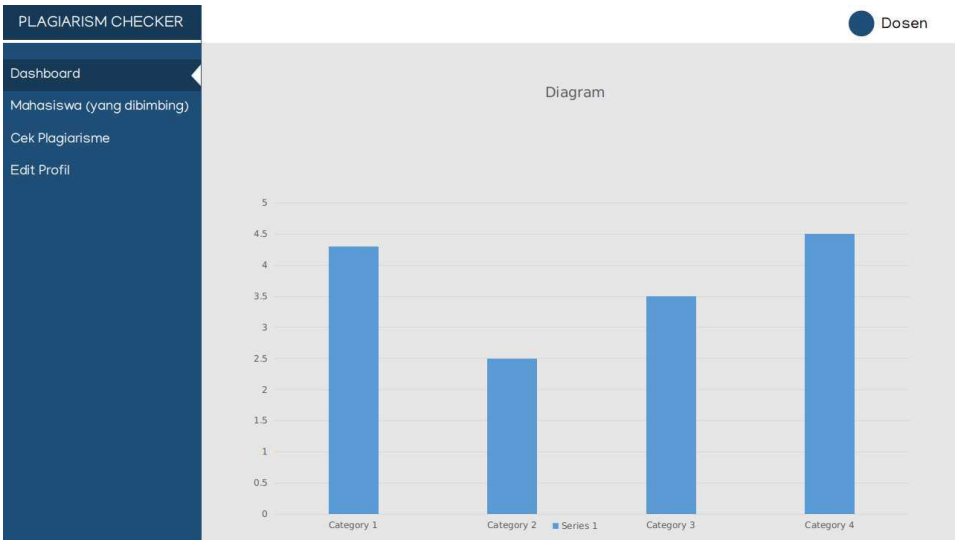
Halaman data mahasiswa menampilkan data mahasiswa aktif yang mengambil sks tugas akhir / skripsi pada tahun ajaran tersebut. Dapat dilihat pada Gambar 3.24



Gambar 3. 24 Rancangan Data Mahasiswa Pada Admin

4. Halaman *Dashboard* Dosen

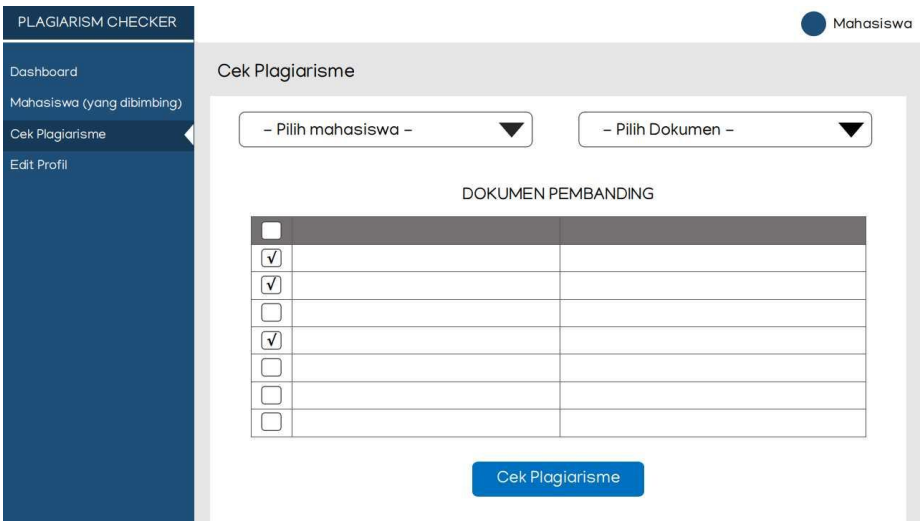
Halaman *dashboard* dosen sebagai halaman utama dosen menampilkan mahasiswa yang sedang dibimbing. Dapat dilihat pada Gambar 3.25



Gambar 3. 25 Halaman *dashboard* Dosen

5. Halaman Cek *Plagiarisme*

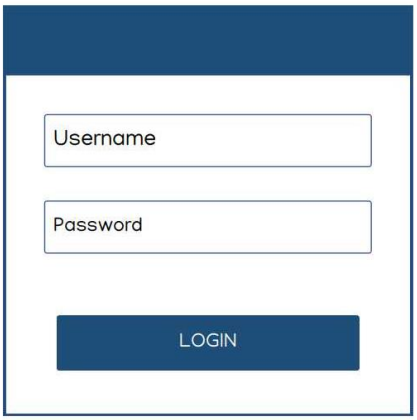
Halaman ini berfungsi sebagai tempat dosen melakukan cek *plagiarisme* pada dokumen milik mahasiswa yang sedang dibimbing. Pada tabel bagian bawah berisi dokumen pembanding Naskah Publikasi. Dapat dilihat pada Gambar 3.26



Gambar 3. 26 Halaman Cek Plagiarisme

6. Halaman Login

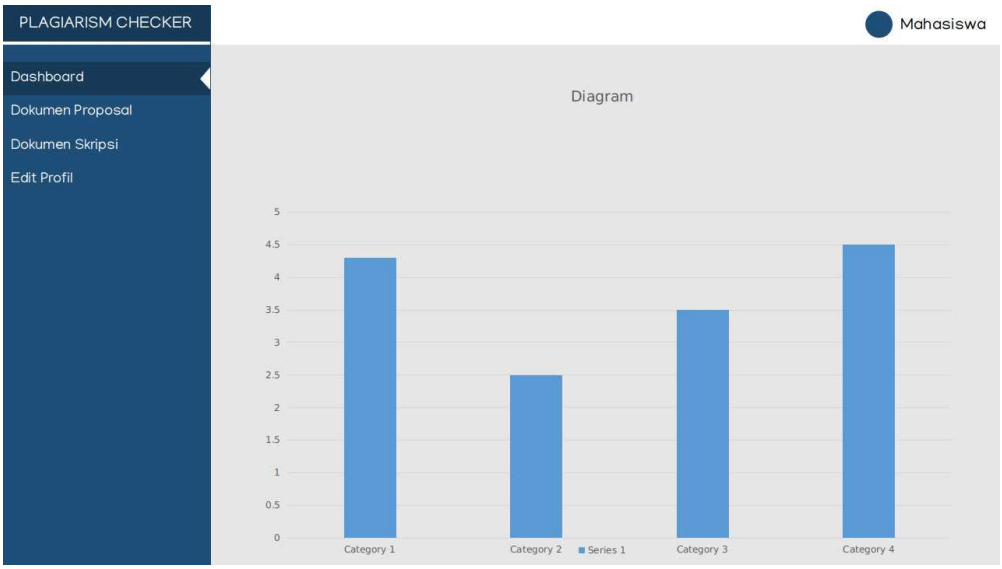
Halaman login digunakan untuk menampilkan menu login yang akan digunakan admin, mahasiswa, dosen untuk login kedalam sistem. Dapat dilihat pada Gambar 3.27.



Gambar 3. 27 Halaman Login

7. Halaman *Dashboard* Mahasiswa

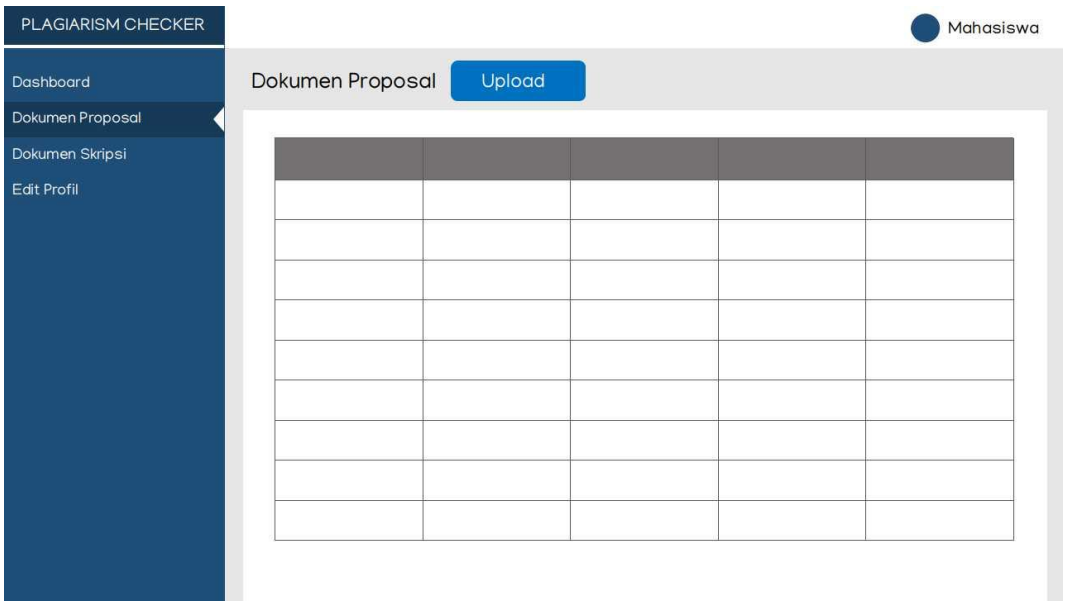
Halaman ini digunakan untuk halaman utama mahasiswa yang berisi data berapa kali melakukan revisi pada kegiatan karya tulis ilmiahnya. Dapat dilihat rancangnnya pada Gambar 3.28



Gambar 3. 28 Halaman Dashboard Mahasiswa

8. Halaman Data Proposal

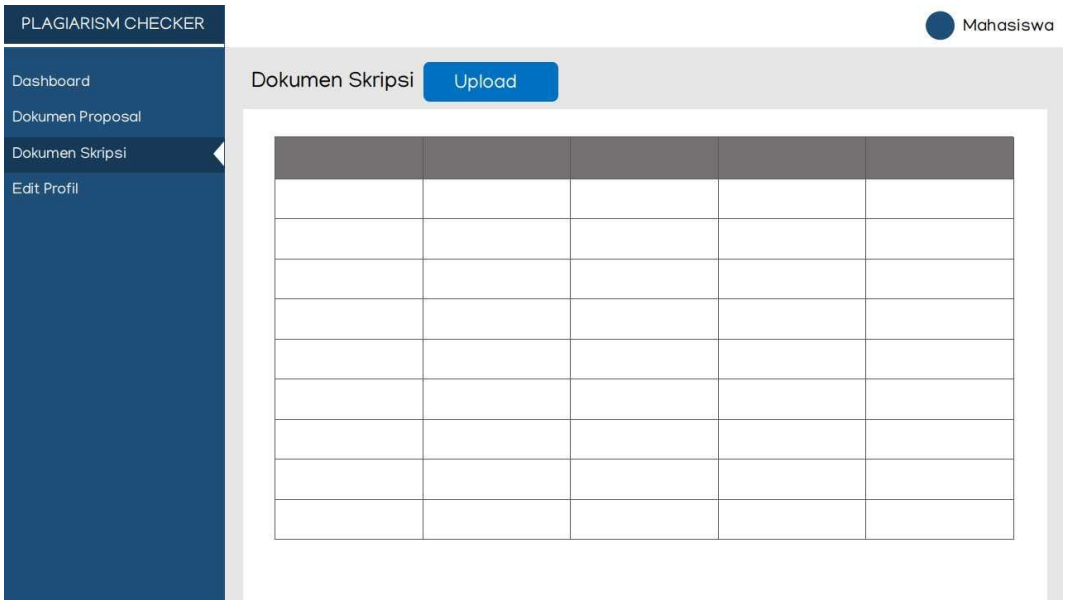
Halaman ini adalah untuk menampilkan halaman upload dokumen proposal. Dapat dilihat pada Gambar 3.29.



Gambar 3. 29 Halaman Data Proposal

9. Halaman Data Skripsi

Halaman ini berfungsi untuk menampilkan menu upload Naskah publikasi bagi mahasiswa yang sudah megikuti seminar pendadaran dan sudah revisi. Dapat dilihat pada Gambar 3.30



Gambar 3. 30 Halaman Data Skripsi

BAB IV

IMPLEMENTASI DAN PENGUJIAN SISTEM

A. Implementasi Sistem

1. Kebutuhan Sistem

a. Hardware

Tabel 4.1 merupakan tabel yang memberikan informasi kebutuhan hardware yang diperlukan agar sistem dapat berjalan.

Tabel 4. 1 Kebutuhan Hardware Sistem

Perangkat Keras	**Spesifikasi**
RAM	Minimum 4 GB
VGA	Minimum 1 GB
Hardisk	Minimun 100 GB Free
Processor	Minimum Intel Celeron
Layar Monitor	Minimum 12 inch

b. Software

Tabel 4.2 merupakan tabel yang memberikan informasi kebutuhan software yang diperlukan agar sistem dapat berjalan.

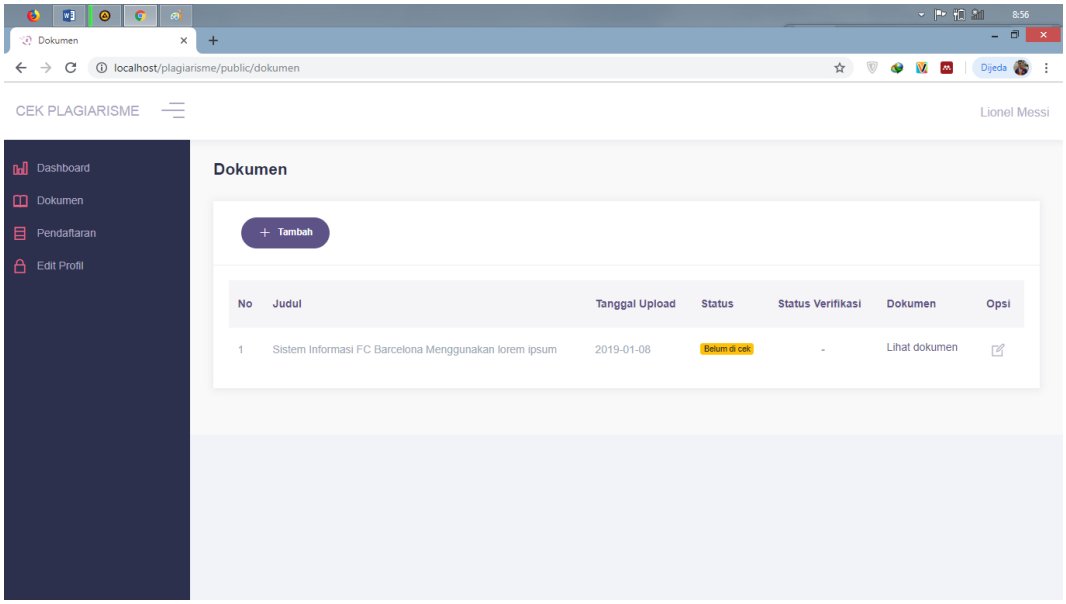
Tabel 4. 2 Kebutuhan Software

Perangkat Lunak
Sistem Operasi Windows 10
XAMPP Support PHP 7.+
Google Chrome

2. Implementasi Tampilan

a. Implementasi Tampilan Mahasiswa Upload Dokumen

Pada gambar 4.1 implementasi tampilan mahasiswa upload dokumen proposal skripsi terdapat sebuah tabel yang menampilkan dokumen proposal skripsi yang sudah diupload dan diatas tabel sebuah button untuk menampilkan layout upload dokumen, dapat dilihat pada gambar dibawah ini.



Gambar 4. 1 Implementasi Tampilan Mahasiswa Upload Dokumen

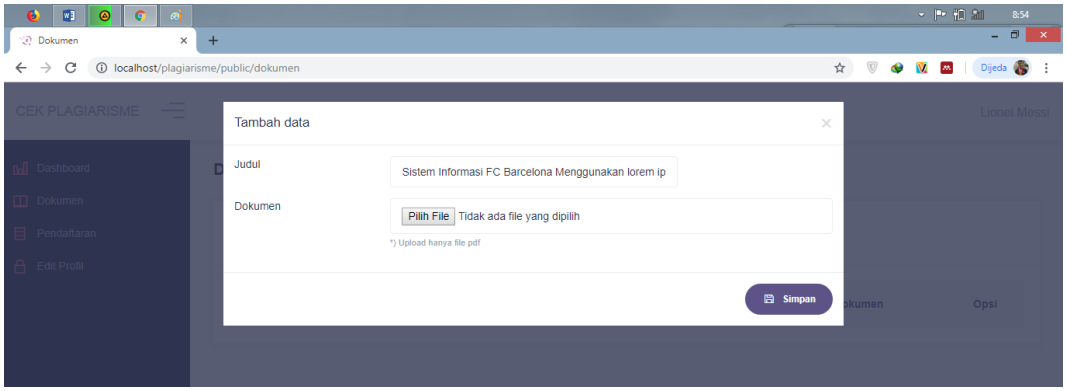
```
.....
                        <tbody>
                            @foreach($data as $row)
                            <tr>
                                <td>{{ ($data-
>currentpage()-1) * $data->perpage() + $loop->index + 1
}}</td>
                                <td>{{ $row->judul }}</td>
                                <td>{{ $row-
>tanggal_upload }}</td>
                                <td>
                                    @if($row->status == 0)
                                        <label
class="badge badge-warning">
                                            Belum di cek
                                        </label>
                                    @else
                                        <label
class="badge badge-success">
                                            Sudah di cek
                                        </label>
```

```
                                    @endif
                                </td>
                                <td>
                                    @if($row->status == 1)
                                        @if($row-
>status_verifikasi == 1)
                                            <label
class="badge badge-success" data-toggle="tooltip">
                                                Diterima
oleh: <br/> {{ $row->dosen->nama }}
                                            </label>
                                        @else
                                            <label
class="badge badge-danger">
                                                Ditolak
oleh:
<br/> {{ $row->dosen->nama }}
                                            </label>
                                        @endif
                                    @else
                                        <center><b>-
</b></center>
                                    @endif
                                </td>
                                <td>
                                    <h5><a href="{{
url('upload/'.$row->document) }}" target="_blank">Lihat
dokumen</a></h5>
                                </td>
                                <td class="td-actions">
                                    @if($row->status == 0)
                                        <center>
                                            <a
href="javascript:void(0)" onclick="edit('{{ $row->id
}}')"><i class="la la-edit edit"></i></a>
                                        </center>
                                    @else
                                        <center>
                                            <button
class="btn btn-info">Dokumen sudah di cek</button>
                                        </center>
                                    @endif
```

Gambar 4. 2 Source Code Tampilan Mahasiswa Upload Dokumen

b. Implementasi Tampilan Upload File

Pada gambar 4.3 merupakan implementasi tampilan dari mahasiswa upload file dokumen proposal skripsi pada saat mahasiswa melakukan klik pada tombol tambah. Berikut adalah tampilannya



Gambar 4. 3 Implementasi Tampilan Upload File

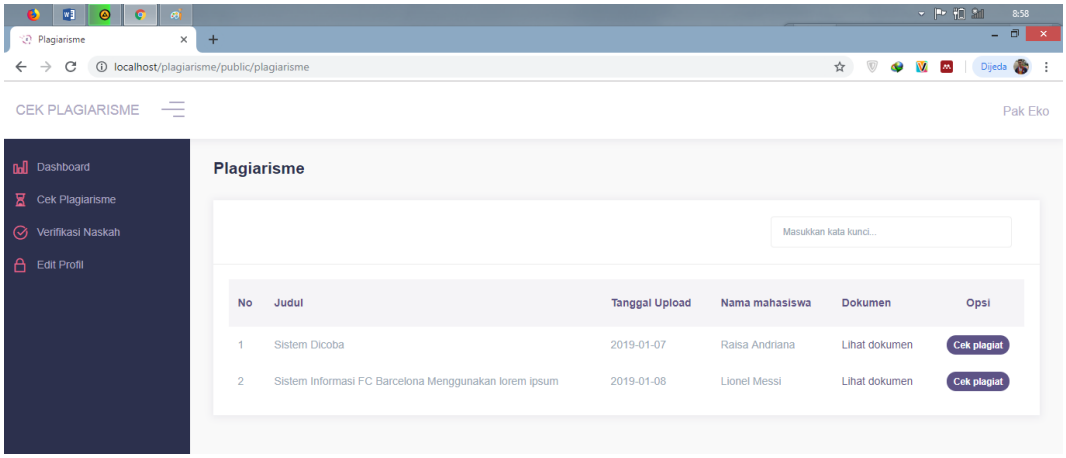
```
.....
{{-- Tambah --}}
<div id="tambah" class="modal fade show">
    <div class="modal-dialog modal-lg">
        <form onsubmit="simpan(); return false;"
class="form-horizontal tambah">
            <div class="modal-content">
                <div class="modal-header">
                    <h4 class="modal-title">Tambah
data</h4>
                    <button type="button" class="close"
data-dismiss="modal">
                        <span aria-hidden="true">×</span>
                        <span class="sr-only">close</span>
                    </button>
                </div>
                <div class="modal-body">
                    <div class="form-group row">
                        <label class="col-md-3 form-
control-label">Judul</label>
                        <div class="col-md-6">
                            <input type="text"
name="judul" class="form-control">
                        </div>
                    </div>
                    <div class="form-group row">
                        <label class="col-md-3 form-
control-label">Dokumen</label>
                        <div class="col-md-9">
                            <input type="file"
id="dokumen" class="form-control"
accept="application/pdf">
                            <small>*) Upload hanya file
pdf</small>
                        </div>
                    </div>
                </div>
                <div class="modal-footer">
```

```
                        <button type="submit" class="btn btn-
primary">
                            <i class="la la-save"></i> Simpan
                        </button>
....
<div id="modal"></div>
```

Gambar 4. 4 Mahasiswa Klik Tombol Tambah

c. Implementasi Tampilan Dokumen Akan Dicek Plagiarisme

Pada gambar 4.5 merupakan implementasi tampilan daftar dokumen – dokumen yang akan dicek plagiarisme pada layout dosen. Berikut dibawah ini merupkan tampilan yang diimplementasikan.



Gambar 4. 5 Implementasi Tampilan Dokumen Akan Dicek Plagiarisme

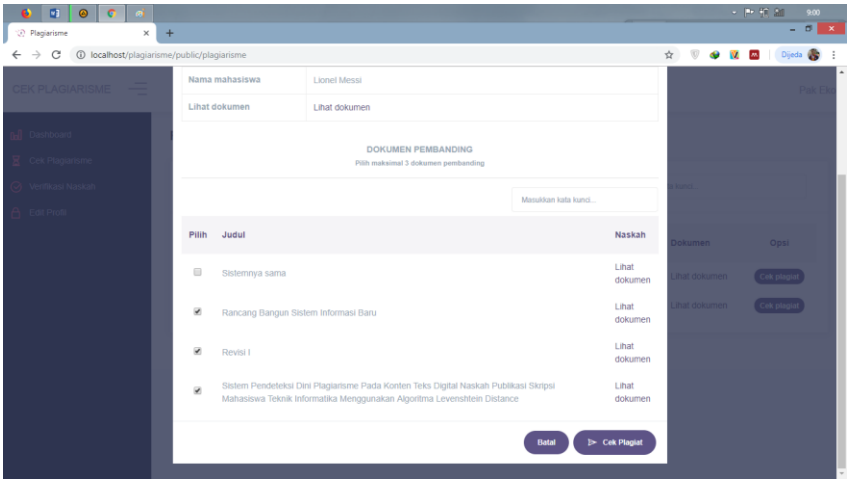
```
....
                        <thead>
                            <tr>
                                <th>No</th>
                                <th>Judul</th>
                                <th>Tanggal Upload</th>
                                <th>Nama mahasiswa</th>
                                <th>Dokumen</th>

<th><center>Opsi</center></th>
                            </tr>
                        </thead>
                        <tbody>
                            @foreach($data as $row)
                            <tr>
                                <td>{{ ($data-
>currentpage()-1) * $data->perpage() + $loop->index + 1
}}</td>
                                <td>{{ $row->judul }}</td>
```

```
                                    <td>{{ $row-
>tanggal_upload }}</td>
                                    <td>{{ $row->mahasiswa-
>nama }}</td>
                                    <td>
                                        <a href="{{
url('upload/'.$row->document) }}" target="_blank">Lihat
dokumen</a>
                                    </td>
                                    <td class="td-actions">
                                        <center>
                                            <a
href="javascript:void(0)" class="btn btn-primary btn-sm"
onclick="cek('{{ $row->id }}')"> Cek plagiat</a>
                                        </center>
                                    </td>
                                </tr>
                                @endforeach
                            </tbody>
                        </table>
.....
<div id="modal"></div>
```

Gambar 4. 6 Implementasi Tampilan Dokumen Akan Dicek Plagiarisme

d. Implementasi Tampilan Pilih Dokumen Pembanding

Pada gambar 4.7 merupakan implementasi tampilan ketika dosen sudah melakukan klik tombol cek plagiarisme, maka dosen diarahkan ke layout modal untuk memilih dokumen pembanding yang akan dicek plagiarisme. Berikut dibawah ini merupakan tampilannya.



Gambar 4. 7 Implementasi Tampilan Pilih Dokumen Pembanding

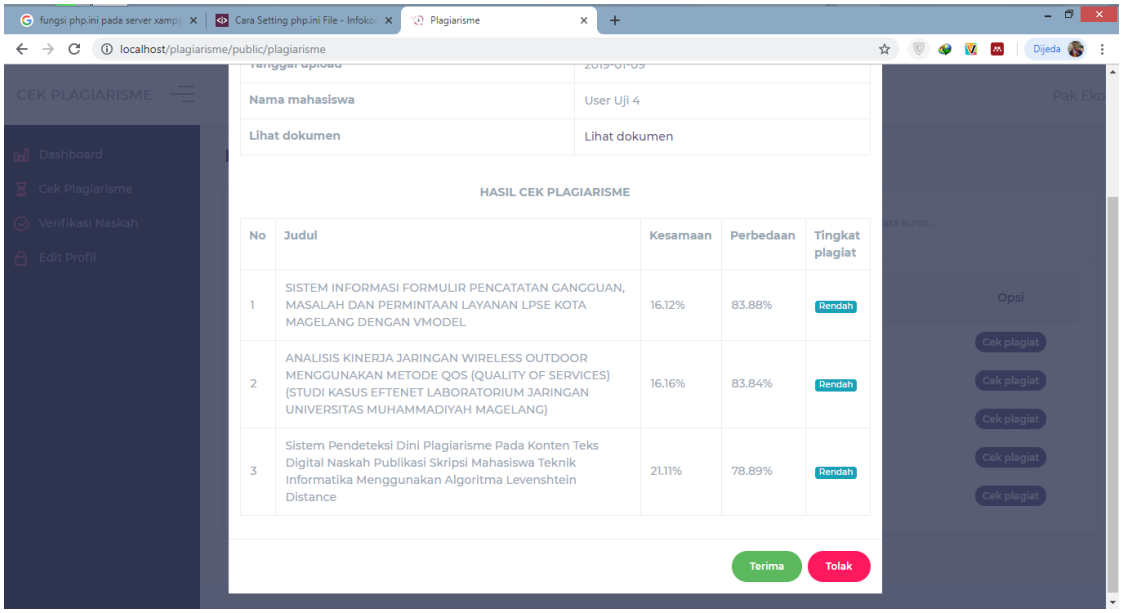
```
<div id="pembanding">
                        <center><b>DOKUMEN
PEMBANDING</b></center>
```

```
                        <center><small><b>Pilih maksimal 3
dokumen pembanding</b></small></center>
                        <hr/>
                        <div class="row mb-2">
                            <div class="col-md-2"></div>
                            <div class="col-md-6">
                                <div class="pull-right">{{
$pembanding->links() }}</div>
                            </div>
.....
                            <tbody>
                                @foreach($pembanding as
$row)
                                <tr>
                                    <td>
                                        <center><input
type="checkbox" name="id_pembanding" value="{{ $row->id
}}" onchange="pilihDokumen(this)"></center>
                                    </td>
                                    <td>{{ $row->judul
}}</td>
                                    <td>
                                        <a href="{{
url('upload/'.$row->naskah_publikasi) }}"
target="_blank">Lihat dokumen</a>
                                    </td>
                                </tr>
                                @endforeach
....
                    <div id="hasil"></div>
                </div>
                <div class="modal-footer">
                    <button type="button" class="btn btn-
primary" id="batal" onclick="$('#cek').modal('hide')" >
                        Batal
                    </button>
                    <button type="submit" class="btn btn-
primary" id="action" >
                        <i class="la la-paper-plane"></i>
Cek Plagiat
                    </button>
```

Gambar 4. 8 Implementasi Tampilan Pilih Dokumen Pembanding

e. Implementasi Tampilan Hasil Cek Plagiarisme

Pada gambar 4.9 merupakan tampilan modal hasil proses cek plagiarisme. Pada tampilan tersebut menampilan berapa persentase persamaan dan perbedaan dari dokumen sumber dengan dokumen pembanding, kemudian menampilkan tingkat plagiat. Berikut dibawah ini merupakan implementasi tampilannya.



Gambar 4. 9 Implementasi Tampilan Hasil Cek Plagiarisme

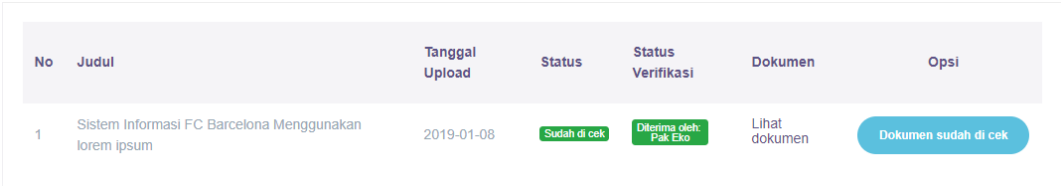
```
......
<center>Hasil Cek Plagiarisme</center>
<br/>
<table class="table table-bordered">
      <tr>
            <th>No</th>
            <th>Pembanding</th>
            <th>Kesamaan</th>
            <th>Perbedaan</th>
            <th>Status</th>
      </tr>
      @for($i = 0; $i < count($data); $i++)
            <tr>
                  <td>{{ $data[$i]['pembanding']['judul']
}}</td>
                  <td>{{ $data[$i]['kesamaan'] }}%</td>
                  <td>{{ $data[$i]['perbedaan'] }}%</td>
                  <td></td>
            <tr>
      @endfor
</table>
@if($row->status == 1)
    @if($row->status_plagiarisme == 'tidak')
    <span class="badge badge-success">{{ ucfirst($row-
>status_plagiarisme) }}</span>
    @elseif($row->status_plagiarisme == 'rendah')
        <span class="badge badge-info">{{ ucfirst($row-
>status_plagiarisme) }}</span>
    @elseif($row->status_plagiarisme == 'sedang')
        <span class="badge badge-warning">{{ ucfirst($row-
>status_plagiarisme) }}</span>
    @else
```

```
        <span class="badge badge-danger">{{ ucfirst($row-
>status_plagiarisme) }}</span>
    @endif
    <br/>
    Di cek oleh {{ $row->dosen->nama }}
@else
    -
@endif
```

Gambar 4. 10 Implementasi Tampilan Hasil Cek Plagiarisme

f. Implementasi Notifikasi Mahasiswa Telah Dicek Plagiarisme

Pada gambar 4.11 merupakan implementasi tampilan pada layout mahasiswa yang dokumennya telah dilakukan cek plagiarisme. Pada layout tersebut menampilkan informasi telah dilakukan cek plagiarisme oleh dosen yang bersangkutan dan telah diterima atau belum, apabila telah diterima mahasiswa dapat melakukan daftar skripsi. Berikut implementasi tampilannya.



Gambar 4. 11 Implementasi Notifikasi Mahasiswa Telah Dicek Plagiarisme

```
......
<div class="widget-body">
                <div class="table-responsive">
                    <table class="table mb-0">
                        <thead>
                            <tr>
                                <th>No</th>
                                <th>Judul</th>
                                <th>Tanggal Upload</th>
                                <th>Status</th>
                                <th>Status Verifikasi</th>
                                <th>Dokumen</th>

<th><center>Opsi</center></th>
                            </tr>
                        </thead>
                        <tbody>
                            @foreach($data as $row)
                            <tr>
```

```
                                    <td>{{ ($data-
>currentpage()-1) * $data->perpage() + $loop->index + 1
}}</td>
                                    <td>{{ $row->judul }}</td>
                                    <td>{{ $row-
>tanggal_upload }}</td>
                                    <td>
                                        @if($row->status == 0)
                                            <label
class="badge badge-warning">
                                                Belum di cek
                                            </label>
                                        @else
                                            <label
class="badge badge-success">
                                                Sudah di cek
                                            </label>
                                        @endif
                                    </td>
                                    <td>
                                        @if($row->status == 1)
                                            @if($row-
>status_verifikasi == 1)
                                                <label
class="badge badge-success" data-toggle="tooltip">
                                                    Diterima
oleh: <br/> {{ $row->dosen->nama }}
                                                </label>
                                            @else
                                                <label
class="badge badge-danger">
                                                    Ditolak
oleh: <br/> {{ $row->dosen->nama }}
                                                </label>
                                            @endif
                                        @else
                                            <center><b>-
</b></center>
                                        @endif
```

Gambar 4. 12 Implementasi Notifikasi Mahasiswa Telah Dicek Plagiarisme

g. Implementasi Algoritma *Levenshtein Distance*

Pada Source Code 4.13 merupakan algoritma yang diterpakan pada sistem untuk melakukan perhitungan cek plagiarisme pada dokumen sumber dengan dokumen pembanding dan menampilkan hasil berupa persentasi kesamaan dan perbedaan pada cek plagiarisme. Berikut adalah source code yang diterapkan.

```
<?php
      namespace App\Library;
```

```
class Plagiarisme {
    private $str1="";
    private $str2="";

    private $arr1=array();
    private $arr2=array();

    function __construct($str1,$str2) {
        $str1=trim($str1); //Menghilangkan spasi awal
        $str2=trim($str2);

        //Cek apakah paragraf kosong atau tidak
        if($str1=="") {
            trigger_error("Paragraf   harus   diisi",
E_USER_ERROR);
        }
        elseif($str2=="") {
            trigger_error("Paragraf   harus   diisi",
E_USER_ERROR);
        }
        else{
            //Deklarasi variabel ke property
            $this->str1=$str1;
            $this->str2=$str2;
            //Memecah  variabel/paragraf  berdasarkan
spasi (jadi array)
            $this->arr1=explode(" ",$str1);
            $this->arr2=explode(" ",$str2);
        }
    }

    //Menghitung persentase kata yang sama
    public function getSimilarityPercentage(){

        //Ambil string dari property
        $str1=$this->str1;
        $str2=$this->str2;

        //Ambil string yang telah di explode
        $tmp1=$this->arr1;
        //Menghitung jumlah array
        $c1=count($tmp1);

        $tmp2=$this->arr2;
        $c2=count($tmp2);

        $count=$c1;
        $t1=$tmp1;
        $t2=$tmp2;

        //Jika jumlah kata string ke-2 lebih besar
dari string ke-1
```

```
            if($c2 > $c1){
                $count=$c2;
                $t1=$tmp1;
                $t2=$tmp2;
            }

            //Mendefinisikan array
            $result=array();
            //Melakukan perulangan berdasarkan jumlah
    kata yang paling banyak
            for($i=0;$i<$count;$i++) {

                //Jika ada kata yang sama
                if(@$t1[$i] == @$t2[$i]) {
                    $result[] = 1; //1 menandakan ada
    string yang sama
                    $resultSame[] = 0; //0 menandakan
    string yang tidak sama
                }
                //Jika tidak ada kata yang sama, proses
    menyamakan string dengan algoritma levenshtein
                else {
                    $result[] = 0;
                    $resultSame[] = levenshtein(@$t1[$i],
    @$t2[$i]);
                }
            }

            //Menghitung total kesamaan
            $countArray = array_count_values($result);
            $one = 0;
            $zero = 0;

            //Menghitung kata yang tidak sama
            if(isset($countArray[0])) {
                $zero=$countArray[0];
            }
            if(isset($countArray[1])) {
                $one=$countArray[1];
            }
            //Jika ada kata yang sama 0
            if($one===0) {
                $percent=number_format(0,2);
            }
            //Jika ada kata yang tidak sama 0
            elseif($zero===0) {
                $percent=number_format(100,2);
            }
            //Jika ada kata yang sama dan tidak sama
            else{
                $per=($one/($one+$zero))*100;
                $percent=number_format($per,2);
            }
```

```
            //Jika semua kata persamaannya 100%
            if($c1===$c2) {
                $words1 = array_diff_assoc($tmp1,$tmp2);
                $words2 = array_diff_assoc($tmp2,$tmp1);
                $sum=array_sum($resultSame);
                $sum=($sum/100);
                $percent=($percent-$sum);
            }

            //Mengembalikan perhitungan persentase kata
     yang sama
            return $percent;
        }

        //Menghitung persentase kata yang berbeda
        public function getDifferencePercentage(){
            $per=$this->getSimilarityPercentage();
            return 100-$per;
        }
}
```

Gambar 4. 13 Implementasi Algoritma Levenshtein Distance

B. Pengujian Sistem

Pengujian sistem merupakan suatu proses pelaksanaan program untuk menguji program atau sistem yang telah dibuat dengan tujuan untuk mengetahui apakah sistem sudah berjalan baik atau masih terdapat error. Pengujian algoritma *Levenshtein Distance* dilakukan dengan cara manual dengan cara melakukan *copy* terhadap *string* sumber dan *string* pembanding kemudian melakukan *paste* pada salah satu *variable script* program. Pengujian ini berfungsi untuk mengetahui *script* yang telah dibuat sudah dapat berjalan dengan baik atau belum, sebelum nanti diimplementasikan sebagai *library* pada sistem yang akan dibuat untuk mendeteksi *plagiarisme* pada dokumen.

1. Pengujian Script Algoritma Levenshtein Distance

Script yang berisi algoritma *Levenshtein Distance* akan dipasang sebagai *library* pada sistem yang akan dibuat nanti. Sebelum digunakan, maka terlebih dahulu dilakukan uji tes agar dapat memberikan respon yang diinginkan.

a. Pengujian *Script*

Pengujian ini dilakukan secara manual dengan meletakkan *string* sumber dan pembanding pada *script* dengan cara melakukan

copy paste kemudian dijalankan pada browser. Dibawah ini merupakan gambar baris kode untuk meletakkan *string* sumber dan pembanding, ditunjukkan pada gambar 4.14.

```
<?php
//Paragraf sumber
$string1=" ";

//Paragraf pembanding
$string2=" ";

 //Memanggil class Plagiarisme.php
require("Plagiarisme.php");

 //Proses komparasi string
$phpCompareStrings=new Plagiarisme($string2, $string1);

//Mendapatkan hasil persentase kesamaan paragraf
$percent=$phpCompareStrings->getSimilarityPercentage();

//Mendapatkan hasil persentase ketidaksamaan paragraf
$percent2=$phpCompareStrings->getDifferencePercentage();

//Menampilkan hasil
echo '$string1 dan $string2 memiliki kesamaan '.$percent.'% dan
perbedaan '.$percent2.'% <br/><br/><br/>';

if($percent >= 50) {
  echo '<br/> Terdeteksi plagiarisme';
}
else {
  echo '<br/> Tidak terdeteksi plagiarisme';
}
?>
```

Gambar 4. 14 Baris Kode String Sumber dan Pembanding

Pada gambar 4.14 diatas telah diletakkan potongan paragraf sumber dan pembanding pada *variable* $string1 dan $string2 yang akan dicek *plagiarisme* dengan perhitungannya akan dijalankan oleh *class plagiarisme*.php untuk mendapatkan persentase kesamaan dan perbedaan. Ditunjukkan *class plagiarisme*.php pada gambar 4.15 dibawah ini.

```
public function getSimilarityPercentage(){

        //Ambil string dari property
        $str1=$this->str1;
        $str2=$this->str2;

        //Ambil string yang telah di explode
        $tmp1=$this->arr1;
        //Menghitung jumlah array
        $c1=count($tmp1);
```

```
        $tmp2=$this->arr2;
        $c2=count($tmp2);

        $count=$c1;
        $t1=$tmp1;
        $t2=$tmp2;

        //Jika jumlah kata string ke-2 lebih besar dari string
ke-1
        if($c2 > $c1){
            $count=$c2;
            $t1=$tmp1;
            $t2=$tmp2;
        }

        //Mendefinisikan array
        $result=array();
        //Melakukan perulangan berdasarkan jumlah kata yang
paling banyak
        for($i=0;$i<$count;$i++) {

            //Jika ada kata yang sama
            if(@$t1[$i] == @$t2[$i]) {
                $result[] = 1; //1 menandakan ada string yang
sama
                $resultSame[] = 0; //0 menandakan string yang
tidak sama
            }
            //Jika tidak ada kata yang sama, proses menyamakan
string dengan algoritma levenshtein
            else {
                $result[] = 0;
                $resultSame[] = levenshtein(@$t1[$i], @$t2[$i]);
            }
        }

        //Menghitung total kesamaan
        $countArray = array_count_values($result);
        $one = 0;
        $zero = 0;
       .........
        //Jika semua kata persamaannya 100%
        if($c1===$c2) {
            $words1 = array_diff_assoc($tmp1,$tmp2);
            $words2 = array_diff_assoc($tmp2,$tmp1);
            $sum=array_sum($resultSame);
            $sum=($sum/100);
            $percent=($percent-$sum);
        }

        //Mengembalikan perhitungan persentase kata yang sama
        return $percent;
    }
```

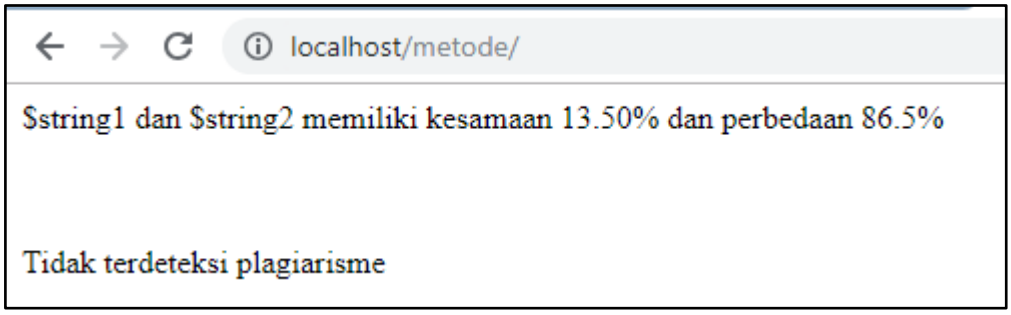
Gambar 4. 15 Class Plagiarisme .php

Pada gambar 4.15 diatas terdapat *function* getSimilarityPercentage() yang nantinya akan digunakan untuk menghitung persamaan antara kedua *string* yang akan dicek *plagiarisme*. Setelah selesai melakukan perhitungan kesamaan, maka akan dihitung juga perhitungan perbedaan antara kedua *string* dengan menjalankan *function* getDifferene Similarity() yang ditunjukkan pada gambar 4.16 dibawah ini.

```
if(isset($countArray[0])) {
            $zero=$countArray[0];
        }
        if(isset($countArray[1])) {
            $one=$countArray[1];
        }
        //Jika ada kata yang sama 0
        if($one===0) {
            $percent=number_format(0,2);
        }
        //Jika ada kata yang tidak sama 0
        elseif($zero===0) {
            $percent=number_format(100,2);
        }
        //Jika ada kata yang sama dan tidak sama
        else{
            $per=($one/($one+$zero))*100;
            $percent=number_format($per,2);
        }
    .......
    //Menghitung persentase kata yang berbeda
    public function getDifferencePercentage(){
        $per=$this->getSimilarityPercentage();
        return 100-$per;
    }
```

Gambar 4. 16 Function getDifferenceSimilarity()

Pada gambar 4.16 melakukan perhitungan perbedaan antara dua *string* yang berbeda setelah pada awal proses melakukan proses menghitung nilai kesamaan. Kemudian hasil akan ditampilkan berupa nilai persentase kesamaan dan perbedaan serta kesimpulan. Seperti yang ditunjukkan pada gambar 4.17 dibawah ini.



Gambar 4. 17 Hasil Perhitungan Algortima Levenshtein Distance

Berikut tabel 4.3 yang menunjukkan hasil uji *script plagiarisme* menggunakan algoritma *Levenshtein Distance.*

Tabel 4. 3 Hasil Uji Script Plagiarisme

No.	Proses	Hasil yang Diharapkan	Kesimpulan
1.	*Variable* $string1 dapat digunakan sebagai *string* sumber.	Dapat digunakan untuk meletakkan *string* sumber.	Diterima
2.	*Variable* $string2 dapat digunakan sebagai *string* pembanding.	Dapat digunakan untuk meletakkan *string* pembanding.	Diterima
3.	*Class Plagiarisme*.php menjadi *requirement* untuk mendapatkan *string* sumber dan pembanding dari *class index*.php	Dapat membaca *string* sumber dan pembanding pada *class index*.php	Diterima
4.	*Function* getSimilarityPercentage() dapat menghitung nilai kesamaan kedua *string.*	Dapat menghitung nilai kesamaan.	Diterima
5.	*Function* getDifferencePercentage() dapat menghitung nilai perbedaan kedua *string*	Dapat mengitung nilai perbedaan.	Diterima

b. Pengujian Waktu Perhitungan

Selain pengujian ketepatan perhitungan algoritma, juga dilakukan uji kecepatan waktu pada algoritma yang digunakan untuk melakukan sebuah perhitungan cek *plagiarisme* sebelum nantinya bisa diterapkan pada sistem yang akan dibuat. Pengujian dilakukan secara manual dan bertahap. Caranya dengan meletakkan paragraf sumber pada *variable* $string1 dan meletakkan paragraf pembanding pada *variable* $string2 yang jumlah wordnya dinaikkan secara berkala. Berikut merupakan tabel hasil uji kecepatan perhitungan algoritma *Levenshtein Distance* pada tabel 4.4

Tabel 4. 4 Hasil Uji Kecepatan Perhitungan

No.	*Count Words String* 1	*Count Words String* 2	*Count Words*	Waktu (H:M:S)	Kesamaan	Perbedaan
1	629	128	757	00:00:00,5	13.50%	86.50%
2	2514	2176	4690	00:00:00,8	3.84%	96.16%
3	17592	10880	28472	00:00:00,9	0.85%	99.15%
4	35184	32640	67824	00:00:01,1	0.84%	99.16%
5	37755	68373	106128	00:00:01,2	0.89%	99.11%
	Rata – Rata Waktu			00:00:00,9		

Gambar 4. 18 Grafik Uji Kecepatan Perhitungan

2. Pengujian Sistem Pendeteksi Dini *Plagiarisme*

Setelah *library* yang dipasang pada komputer server *plagiarisme* dapat bekerja dengan baik, kemudian akan dilakukan pengujian pada sistem Pendeteksi Dini *Plagiarisme* untuk menerima respon dan proses yang nantinya akan ditampilkan pada pengguna. Berikut tabel hasil pengujian yang ditunjukkan pad tabel 4.5

Tabel 4. 5 Hasil Uji Sistem Pendeteksi Dini Plagiarisme

No.	Aktifitas/Menu	Hasil yang Diharapkan	Tercapai	
			Ya	Tidak
1.	Login • Admin	1. Pengguna masuk ke aplikasi Sistem Pendeteksi Dini *Plagiarisme* sebagai "Admin" dengan *username* dan *password* yang sesuai. 2. *Dashboard* terbuka dan Pilihan Menu tersedia.	√	
	• Mahasiswa	1. Pengguna masuk ke aplikasi Sistem Pendeteksi Dini *Plagiarisme* sebagai Mahasiswa dengan *username* dan *password* yang sesuai. 2. *Dashboard* terbuka dan Pilihan Menu tersedia.	√	
	• Dosen	1. Pengguna masuk ke aplikasi Sistem Pendeteksi Dini *Plagiarisme* sebagai Dosen dengan *username* dan *password* yang sesuai. 2. *Dashboard* terbuka dan Pilihan Menu tersedia.	√	
2.	Menu Data Mahasiswa	1. Pengguna yang hanya *login* sebagai Admin dapat menambah, mengedit, menghapus Data Mahasiswa untuk keperluan login Mahasiswa.	√	

		2. Dapat menyimpan konfigurasi ke *database*.		
3.	Menu Data Dosen	1. Pengguna yang hanya *login* sebagai Admin dapat menambah, mengedit, menghapus Data Dosen untuk keperluan login Mahasiswa. 2. Dapat menyimpan konfigurasi ke *database.*	√	
4.	Menu Pendaftaran	1. Pengguna yang hanya *login* sebagai Admin dapat memverifikasi pendaftaran skripsi Mahasiswa agar dapat *upload* file Naskah Publikasi. 2. User mahasiswa dapat *upload* file Naskah Publikasi.	√	
5.	Menu Dokumen	1. Pengguna yang hanya *login* sebagai Mahasiswa dapat *upload* dokumen proposal skripsi untuk dicek *similarity*. 2. Dokumen tersimpan pada database dan tampil tabel dokumen yang telah di*upload*. 3. Jika dokumen hasil cek *similarity* diterima dosen, maka *button* Tambah berubah menjadi Daftar.	√	
6.	Menu Pendaftaran	1. Pengguna yang hanya *login* sebagai Mahasiswa dapat melakukan *upload* dokumen Naskah Publikasi jika pendaftaran sudah diverifikasi Admin. 2. Dokumen tersimpan pada *database* dan	√	

		tampil tabel dokumen yang telah di*upload.*		
7.	Menu *Similarity*	1. Pengguna yang hanya *login* sebagai Dosen dapat melakukan cek *similarity* dokumen proposal skripsi yang telah di *upload.* 2. Tampil tabel daftar dokumen.	√	
8.	Klik *button* Cek *Similarity*	1. Tampil modal detail dokumen proposal dan daftar dokumen pembanding. 2. Dapat memilih dokumen pembanding. 3. Dapat melakukan perhitungan cek *similarity* dan tampil hasil. 4. Setelah selesai melakukan perhitungan, muncul *button* Terima dan Tolak. 5. Dapat melakukan verifaksi Terima atau Tolak hasil perhitungan.	√	
9.	Menu Verifikasi	1. Pengguna yang hanya *login* sebagai Dosen dapat melakukan verifikasi Naskah Publikasi yang telah di*upload* oleh mahasiswa. 2. Tampil tabel daftar dokumen Naskah Publikasi.	√	

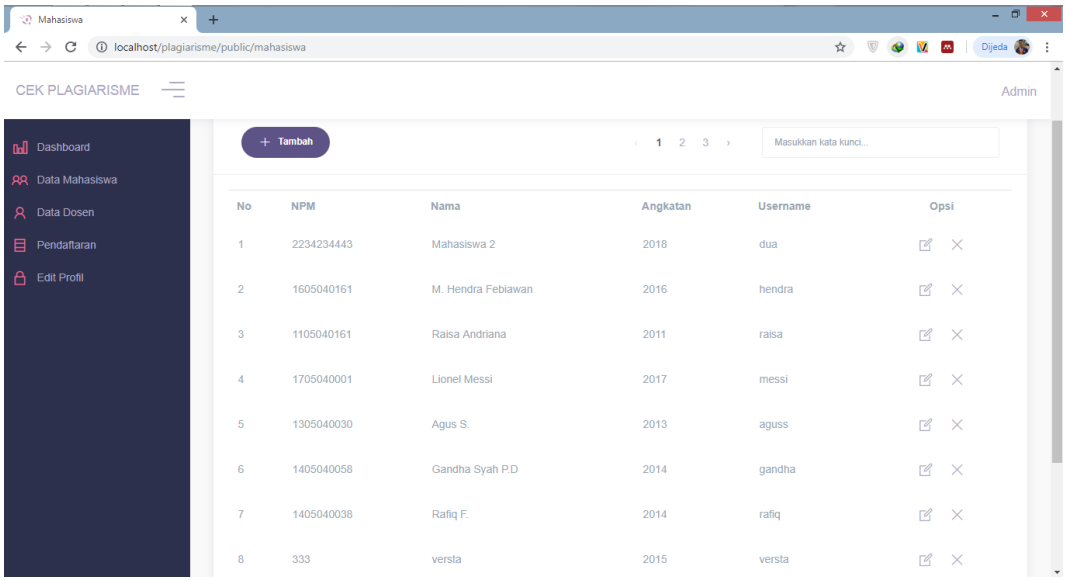
BAB V
HASIL DAN PEMBAHASAN

A. Hasil Pengujian Sistem

Pengujian yang telah dilakukan, kemudian akan memberikan hasil yang menentukan apakah sistem dapat bekerja dengan baik dan sesuai hasil yang diharapkan. Berikut ini hasil setiap bagian sistem yang dijalankan.

1. Hasil Admin *Input* Data Mahasiswa

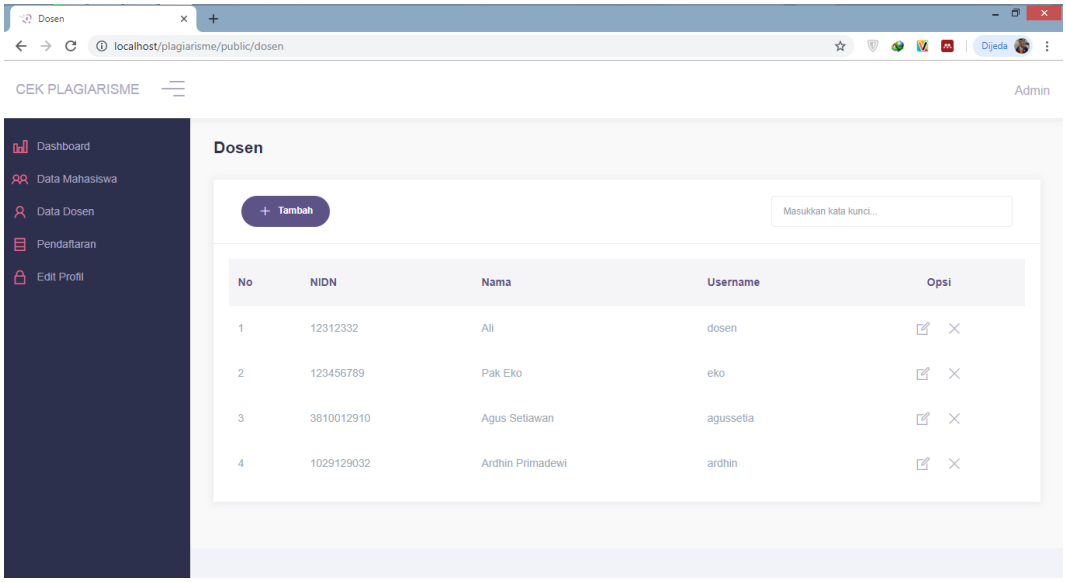
   Berikut ini adalah hasil yang didapat setelah admin menginputkan data *dummy* mahasiswa pada sistem dan dapat tersimpan pada database. Dapat dilihat pada gambar 5.1



Gambar 5. 1 Hasil Admin Input Data Mahasiswa

2. Hasil Admin *Input* Data Dosen

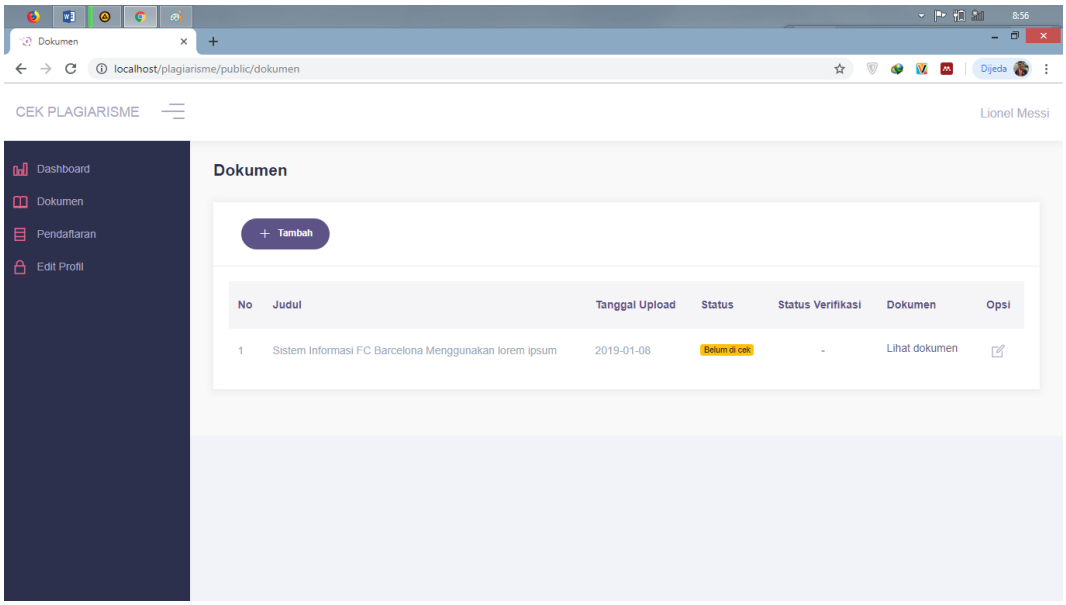
   Berikut ini merupakan hasil input data dosen pada sistem dan dapat tersimpan pada database agar bisa melakukan cek plagiarisme. Dapat dilihat pada gambar 5.2 dibawah.



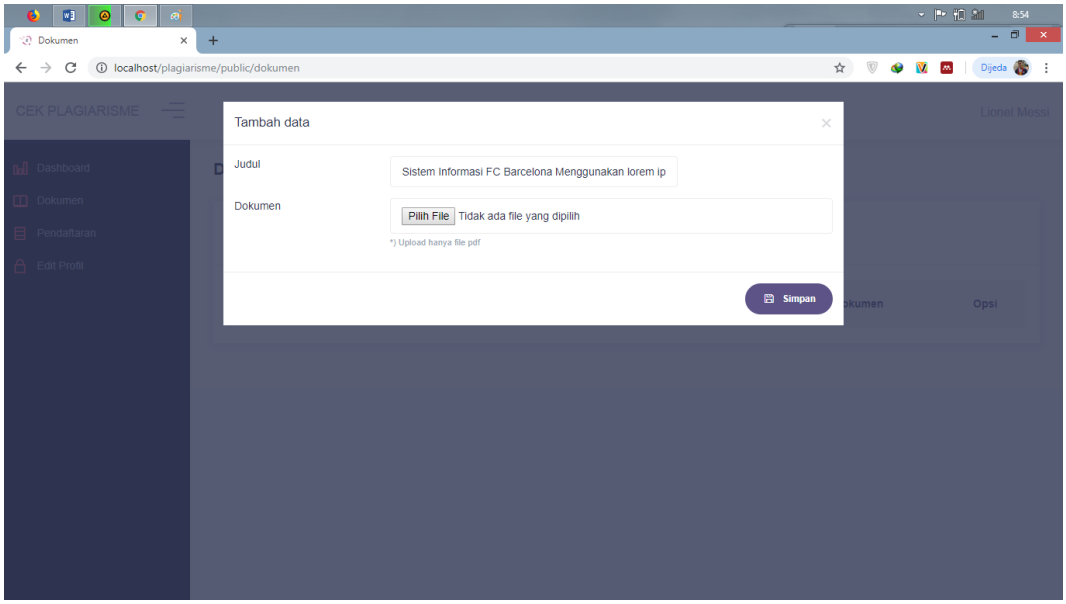
Gambar 5. 2 Hasil Admin Input Data Dosen

3. Hasil Mahasiswa Tambah Dokumen

Pada gambar 5.3 dan 5.4 dibawah ini, mahasiswa setelah melakukan login akan diarahkan ke layout mahasiswa kemudian dapat memilih menu dokumen agar dapat mengupload atau menambah dokumen proposal skripsi untuk dicek *plagiarisme*. Dapat dilihat pada gambar dibawah ini.



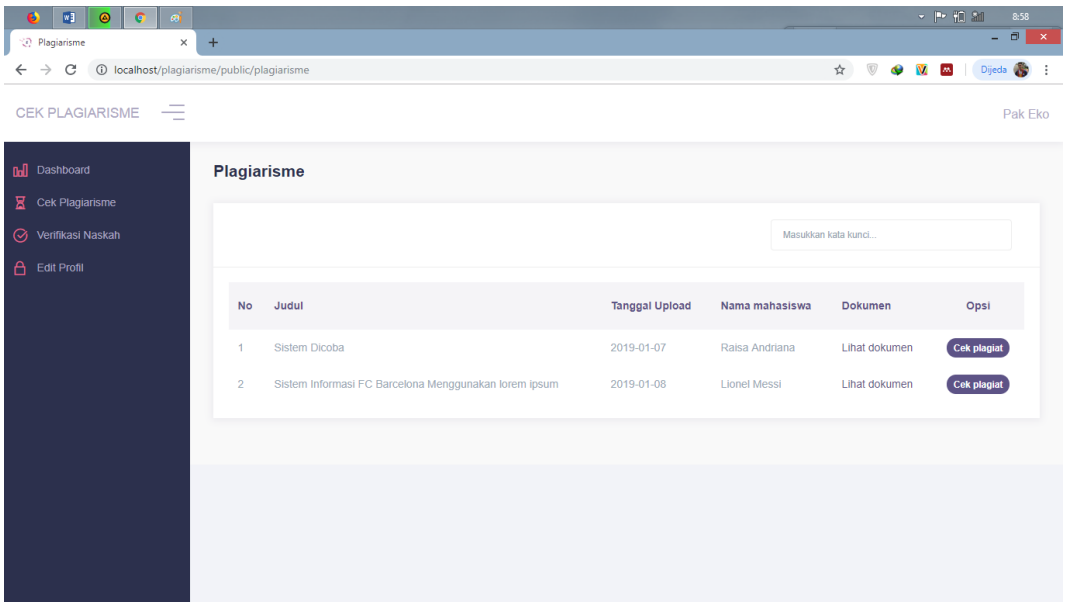
Gambar 5. 3 Hasil Klik Tombol Menu Dokumen Mahasiswa



Gambar 5. 4 Hasil Klik Tombol Tambah Untuk Upload Dokumen

4. Hasil Dashboard Menu Dosen Cek Plagiarisme

Berikut ini merupakan gambar 5.5 pada sisi dosen setelah login maka akan muncul dashboard menu Cek Plagiarisme, apabila diklik maka dosen akan diarahkan ke daftar dokumen mana saja yang akan dilakukan cek *plagiarisme*. Dapat dilihat pada gambar dibawah ini.

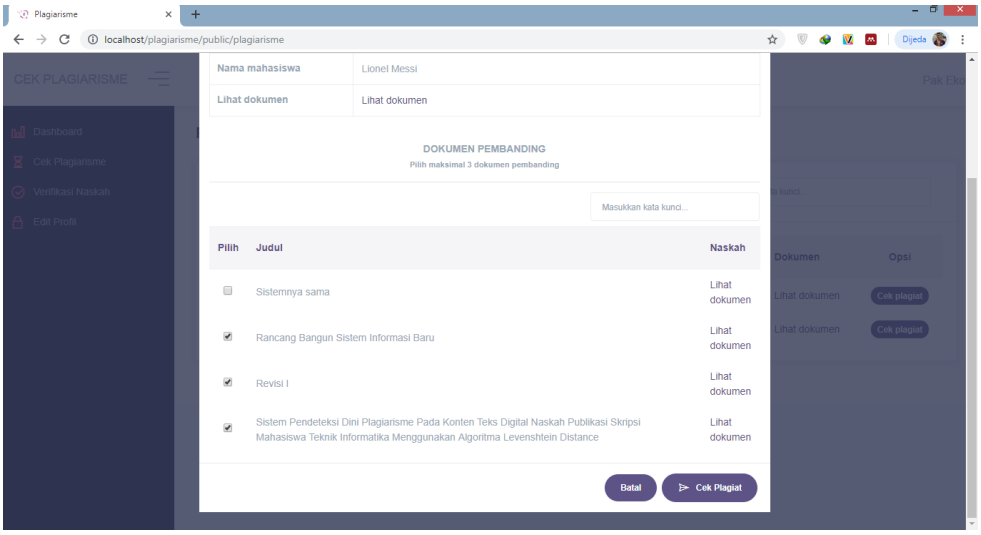


Gambar 5. 5 Hasil Dashboard Menu Dosen Cek Plagiarisme

5. Hasil Tampilan Modal Cek Plagiat

Pada gambar 5.6 akan ditunjukkan sebuah tampilan modal apabila dosen sudah melakukan tombol Cek Plagiat pada sebuah dokumen.

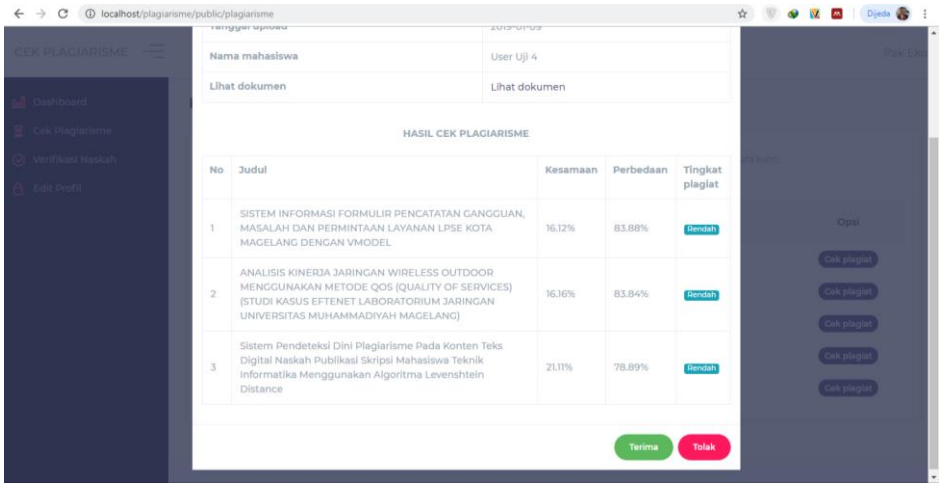
Modal berisi informasi dokumen sumber, kemudian dibawah diikuti dengan beberapa daftar dokumen pembanding yang akan dipilih untuk dibandingkan dengan dokumen sumber. Dapat dilihat gambar dibawah ini.



Gambar 5. 6 Hasil Tampilan Modal Cek Plagiat

6. Hasil Perhitungan Cek Plagiat

Gambar 5.7 akan menunjukkan sebuah *output* yang dihasilkan apabila dosen melakukan klik pada tombol Cek Plagiat. Hasil akan ditampilkan berupa persentase kesamaan dan perbedaan dokumen serta kesimpulan dari hasil persentase nilai cek *plagiarisme*. Berikut hasil yang ditampilkan pada gambar dibawah.



Gambar 5. 7 Hasil Perhitungan Cek Plagiat

B. Pembahasan

Berdasarkan pengujian diatas, terdapat beberapa *output* yang dihasilkan sehingga sistem dapat berjalan sesuai dengan apa yang diharapkan. Secara garis besar sebelum dapat dilakukan perhitungan cek *plagiarisme* alur yang dijalankan sistem seperti berikut ini. Pertama mahasiswa login, kemudian *upload* dokumen dengan cara memilih menu Dokumen, selanjutnya pada aktor dosen juga terlebih dahulu melakukan login kemudian memilih dokumen yang akan dicek, lalu cek plagiat dengan dokumen pembanding, jika hasil cek diterima dosen maka mahasiswa melakukan daftar skripsi apabila ditolak maka akan melakukan revisi lalu upload lagi. Jika sudah maka admin verifikasi pendaftaran skripsi, apabila telah sampai tahap selesai melakukan sidang akhir maka mahasiswa baru dapat *upload* dokumen naskah publikasi untuk dijadikan dokumen pembanding, terakhir dosen verifikasi dokumen naskah publikasi.

1. Perhitungan Mencari Nilai Kesamaan

   Pada proses cek *plagiarisme*, tahapan paling utama adalah melakukan perhitungan mencari persentase nilai *string* yang sama. Secara umum mencari nilai persentase kesamaan *string* dengan memanfaatkan algoritma *Levenshtein Distance* dapat ditulis dengan rumus :

   $$Plagiarisme\ Value = \left(\frac{diff}{\max(CS, ST)}\right) \times 100\%$$

   Keterangan :

   Diff = Jarak *Levenshtein*

   CS = *Source String*

   ST = *Target String*

   Contoh kasus, kita ambil kata “twitter v1” dan dengan kata pembanding “twitter v2”, kemudian kita lakukan perhitungan dengan memanfaatkan metode *trigrams* :

   Trigrams “twitter v1” : {_tw twi wit itt tte ter er_ _v1 v1_}

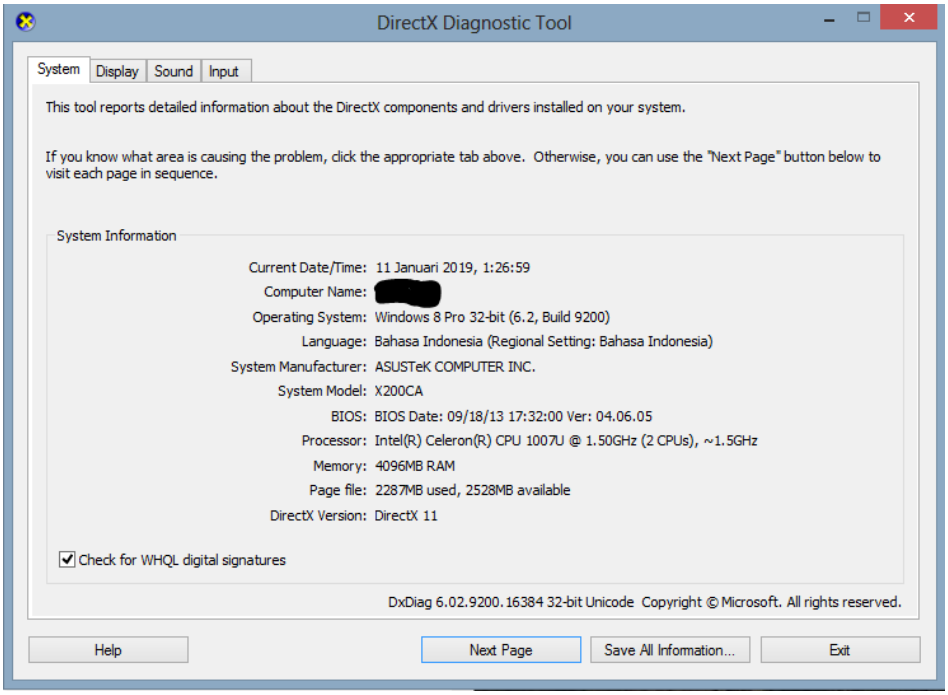
   Trigrams “twitter v2” : {_tw twi wit itt tte ter er_ _v2 v1_}

Sehingga kita mendapatkan 7 angka diff dari kedua *string* diatas melalui cara *trigrams* yaitu 7 diff {_tw twi wit itt tte ter er_} dengan 11 nilai *max(CS,ST)* yaitu {_tw twi wit itt tte ter er_ _v1 _v2 v1_ v2_} maka dapat dihitung,

$$Plagiarisme\ Value = \left(\frac{7}{11}\right) \times 100\% = 63{,}63\ \%\ Similarity$$

2. Waktu Respon Perhitungan Algoritma

Selain itu algoritma yang digunakan juga harus diuji lagi agar dapat mengetahui seberapa efektif algoritma yang digunakan untuk diterapkan pada sistem. Mengingat banyaknya *resources* yang dihitung termasuk banyak *words* yang terkandung dalam sebuah dokumen, maka telah dilakukan uji efektifitas kecepatan algoritma dalam membaca dan menghitung suatu dokumen. Selain itu dengan banyaknya *resource* yang akan disimpan pada server nantinya, kita juga harus memperhatikan seberapa mampu komputer yang akan digunakan sebagai server. Berikut ini adalah spesifikasi minimum yang digunakan sebagai server untuk membantu menghitung kecepatan menghitung cek *plagiarisme*.



Gambar 5. 8 Spesifikasi Minimum Komputer

Pada pengujian sistem yang dilakukan, sistem memeriksa dokumen sumber dengan dokumen pembanding. Daftar dokumen sumber yang diuji dapat ditunjukkan oleh tabel 5.1

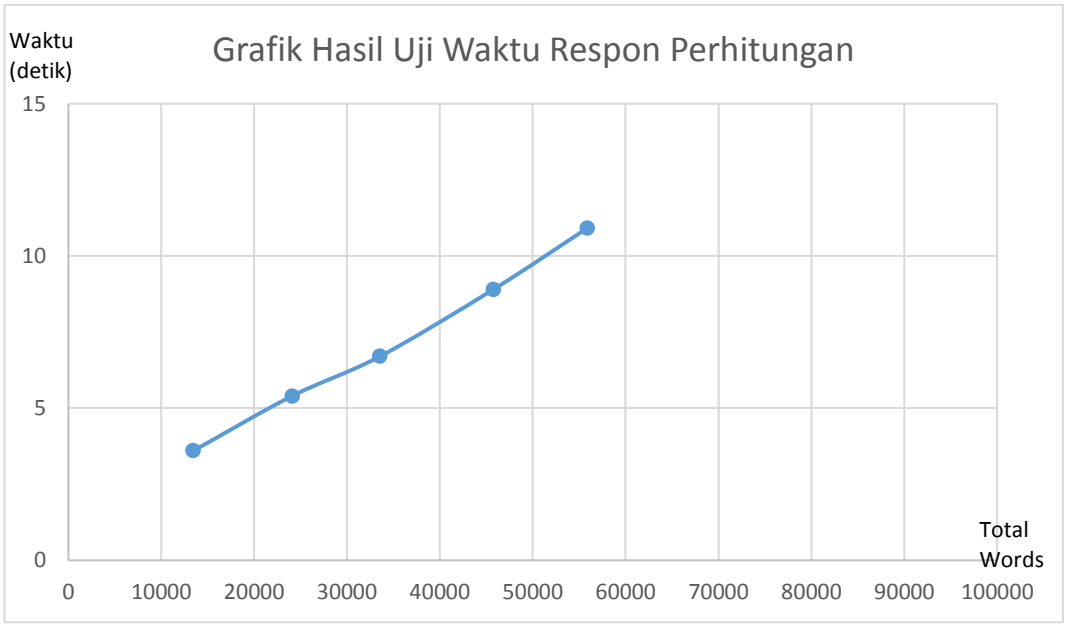
Tabel 5. 1 Daftar Dokumen Sumber

No.	Nama File	Tipe File	Ukuran File	*Count Words*
1.	Tes Plagiat 1	.pdf	597 KB	4405
2.	Tes Plagiat 2	.pdf	462 KB	3288
3.	Tes Plagiat 3	.pdf	295 KB	1980
4.	Tes Plagiat 4	.pdf	659 KB	2890
5.	Tes Plagiat 5	.pdf	852 KB	2626

Pada perbandingan antara dokumen sumber dengan dokumen pembanding saat ini langsung dilakukan melalui sistem, tidak seperti pada waktu uji coba *script* sebelum dimasukkan ke sistem sebagai *library*. Sebelum menjalankan tes uji coba, dilakukan konfigurasi terlebih dahulu pada **memory_limit** yang terletak difile **php.ini.** Hal ini karena sistem masih tahap pengembangan dan perlunya uji coba pada sistem serta komputer yang nanti akan digunakan, agar *software* yang berjalan terhindar dari kelebihan memori yang mengakibatkan *crash*. Selain itu konfigurasi ini agar memaksa *software* yang kita gunakan menggunakan kapasitas RAM yang tersedia pada komputer kita. **Memory_limit** yang diatur pada konfigurasi *default* adalah 128MB, diubah menjadi 512MB. Maka diperoleh hasil seperti terlihat pada tabel 5.2 dibawah.

Tabel 5. 2 Hasil Uji Waktu Respon

No.	Doc. Sumber	Jumlah Doc. Pembanding	*Count Words*	Waktu (H:M:S)
1.	Tes Plagiat 1	3	13465	00:00:03.57
2.	Tes Plagiat 2	5	24108	00:00:05.36
3.	Tes Plagiat 3	7	33554	00:00:06.67
4.	Tes Plagiat 4	9	45766	00:00:08.98
5.	Tes Plagiat 5	11	55871	00:00:10.91



Gambar 5. 9 Hasil Uji Waktu Respon Perhitungan

Dari hasil pengujian dokumen tersebut menunjukkan bahwa yang mempengaruhi lama tidaknya perhitungan nilai *similarity* adalah jumlah *string* dan *words* yang terkandung dalam dokumen. Dapat dilihat pada tabel pengujian diatas pada Tes Plagiat 1 ke Tes Plagiat 2 memiliki interval kenaikan waktu ± 2 detik dengan kenaikan jumlah *words* sebesar 10643, kemudian pada Tes Plagiat 2 ke 3 memiliki interval waktu ± 2 detik hampir sama dengan interval sebelumnya dengan selisih jumlah *words* 9446. Pada Tes Plagiat 3 ke 4 memiliki interval waktu ± 2 detik hampir 3 detik dengan selisih jumlah total *words* 12212, dan pada Tes Plagiat 4 ke 5 memiliki selang waktu selama ± 2 detik dengan selisih total *words* 10105. Selain itu, proses *parsing* dokumen dari format **.pdf** ke **.txt**, hal ini dapat dilihat melalui perbandingan sebelum dan sesudah *script* diterapkan pada sistem, yang pada awalnya belum dalam bentuk dokumen **.pdf** komputer dapat menghitung dengan sangat cepat, kemudian diubah dengan melalui proses *parsing* dokumen, komputer membutuhkan waktu yang sedikit lama.

BAB VI
KESIMPULAN DAN SARAN

A. Kesimpulan

Dari pembahasan yang sudah diuraikan maka dapat disimpulkan sebagai berikut :

1. Pada pengujian dengan menggunakan data real dokumen yang memiliki perbedaan, dari dokumen 1 dengan *count words* berjumlah 629 dan dokumen 2 dengan *counts words* 128 diketahui nilai kesamaan 13,50% dan nilai perbedaan 86,50%.
2. Pada pengujian efektifitas kecepatan perhitungan, dokumen sumber dengan *count words* 4405 diuji dengan 3 dokumen pembanding dengan total *count words* 13465 memiliki kecepatan menghitung selama 3,57 detik dan jika jumlah dokumen ditambah maka lama perhitungan juga akan naik.
3. Kecepatan perhitungan algoritma *Levenshtein Distance* tergantung pada jumlah total *words* yang terkandung dalam dokumen sumber dan dokumen pembanding. Semakin banyak *words* yang terkandung maka akan membutuhkan waktu yang sedikit lama.
4. Setelah menerapkan algoritma *Levenshtein Distance* pada sistem, proses *parsing* dokumen berbentuk **.pdf** ke dalam bentuk *string* mempengaruhi kecepatan waktu perhitungan cek *similarity*.

B. Saran

Dari hasil kesimpulan diatas, penggunaan Sistem Pendeteksi Dini *Plagiarisme* untuk mengetahui tingkat persentase kesamaan dan perbedaan isi dokumen memiliki potensi besar untuk dapat dikembangkan lagi agar lebih baik, oleh karena itu beberapa saran yang dapat digunakan sebagai pertimbangan yang sekiranya dapat berguna untuk masing-masing sistem :

1. File dokumen yang diupload tidak hanya berformat .pdf, tapi bisa ditambahkan format .doc, .docx agar lebih cepat melakukan perhitungan tanpa harus melalui tahap *parsing* dokumen.

2. Hasil perhitungan tidak hanya menampilkan besar persentase kesamaan dan perbedaan dari isi dokumen, melainkan juga dapat mengatahui kata atau kalimat mana saja yang memiliki kesamaan.

DAFTAR PUSTAKA

DocForge. (2014). *Web Application Framework*. Diambil kembali dari DocForge: http://docforge.com/wiki/Web_application_framework

Dwi Susanto, A. B. (2016). Deteksi Plagiat Dokumen Tugas Daring Laporan Praktikum Mata Kuliah Desain Web Menggunakan Metode Naive Bayes. *Nusantara Journal of Computers and its Applications*, 1.

F, L. J. (2014). *PHP Metrics*. Diambil kembali dari PHP Metrics: http://www.phpmetrics.org

Hamidillah Ajie, A. S. (2017). Aplikasi Pendeteksi Dugaan Awal Plagiarisme Pada Tugas SIswa dan Mahasiswa Berdasarkan Kemiripan Isi Teks Menggunakan Algoritma Levenshtein Distance. *Jurnal Pinter Vol. 1 No. 1 Juni 2017*, 24.

Harlian, M. (2015, Juni 4). *Lecterur Pens.* Diambil kembali dari http://iwanarif.lecterur.pens.ac.id/: http://iwanarif.lecturer.pens.ac.id/kuliah/dm/6Text%20 Mining.pdf

KEMENRISTEKDIKTI, P. (2018, 11 01). *Grafik Jumlah Perguruan Tinggi.* Diambil kembali dari Pangkalan Data Pendidikan Tinggi Kementerian Riset, Teknologi dan Perguruan Tinggi: https://forlap.ristekdikti.go.id/perguruantinggi/homegraphpt

LLC, M. S. (2015). *Source Code Size Metrics*. Diambil kembali dari MSquared Technologies: http://msquaredtechnologies.com/m2rsm/docs/rsm_metrics_narration.htm

M, A. N. (2010). Implementasi Algoritma Levenshtein Distance dan Metode Empiris untuk menampilkan saran perbaikan kesalahan pengetikan dokumen berbahasa Indonesia. *Skripsi*.

Mufti Ari Bianto, S. R. (2018). Perancangan Sistem Pendeteksi Plagiarisme Terhadap Topik Penelitian Menggunakan Metode K-Means Clustering dan Model Bayesian. *Seminar Nasional Teknologi Informasi dan Multimedia 2018*, 19.

Na'firul Hasna Ariyani, S. R. (2016). Aplikasi Pendeteksi Kemiripan Isi Teks Dokumen Menggunakan Metode Levenshtein Distance. *semanTIK Vol. 2, No. 1*, 279.

Nugroho, A. (2015). *Analisis dan Perancangan Sistem Informasi dengan Metodologi Berorientasi Objek.* Bandung: Informatika.

Panji Novantara, O. P. (2018). Implementasi Algoritma Jaro-Winkler Distance Untuk Sistem Pendeteksi Plagiarisme Pada Dokume Skripsi. *Jurnal Buffer Informatika*, 8.

PDDIKTI, F. (2018, 11 01). *Pencarian Perguruan Tinggi*. Diambil kembali dari Pangkalan Data Pendidikan Tinggi Kementerian Riset Teknologi dan Pendidikan Tinggi: https://forlap.ristekdikti.go.id/perguruantinggi/search

Rocky Yefrenes Dillak, F. L. (2016). Sistem Deteksi Dini Plagiarisme Tugas Akhir Mahasiswa Menggunakan Algortima N-Grams dan Winnowing. *Jurnal Ilmiah Flash, [S.1], v. 2, n. 1, p. 12-18,* 12.

Rudianto, A. M. (2011). Pemrograman Web Dinamis Menggunakan PHP. *Andi Offset*.

Sastroasmoro, S. (2007). Beberapa Catatan Tentang Plagiarisme. *Majalah Kedokteran Indonesia*, Volume 57.

Sora. (2018, November 07). *Mengetahui Pengertian Dokumen dan Dokumentasi*. Diambil kembali dari Pengertianku: http://www.pengertianku.net/2014/09/mengetahui-pengertian-dokumen-dan-dokumentasi.html

Tudesman, E. O. (2014). Sistem Deteksi Plagiarisme Dokumen Bahasa Indonesia Menggunakan Metode Vector Space Model. *Jurnal STMIK GI MDP*, Page 2.

Waliyanto. (2000). Sistem Basis Data Analisis dan Pemodelan Data. *J&J Learning*.

Widodo, P. P. (2011). UML (Unified Modeling Languange). *Herlawati*.