DETEKSI PLAGIARISME PADA DOKUMEN SKRIPSI BERDASARKAN TINGKAT KESAMAAN DENGAN MENGGUNAKAN METODE *Longest Common Subsequence*

PROPOSAL SKRIPSI

Oleh:

IMAM NAWAWI NIM. 1541180020

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2018

DETEKSI PLAGIARISME PADA DOKUMEN SKRIPSI BERDASARKAN TINGKAT KESAMAAN DENGAN MENGGUNAKAN METODE *Longest Common Subsequence*

PROPOSAL SKRIPSI

Digunakan Sebagai Syarat Maju Ujian Diploma IV

Politeknik Negeri Malang

Oleh:

IMAM NAWAWI NIM. 1541180020

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2018

HALAMAN PENGESAHAN

DETEKSI PLAGIARISME PADA DOKUMEN SKRIPSI BERDASARKAN TINGKAT KESAMAAN DENGAN MENGGUNAKAN METODE *Longest Common Subsequence*

Disusun oleh:

IMAM NAWAWI NIM. 1541180020

Proposal Skripsi ini telah diuji pada 7 Desember 2018

Disetujui oleh:

1. Penguji I : Dr.Eng. Faisal Rahutomo ST., M.Kom.
   NIP. 197711162005011008 .....
2. Penguji II : Mustika Mentari, S.Kom., M.Kom.
   NIDN. 0007060804 .....
3. Pembimbing I : Putra Prima Arhandi, S.T., M.Kom.
   NIP. 198611032 01404 1 001 .....

Mengetahui,

Ketua Jurusan Teknologi Informasi	Ketua Program Studi Manajemen Informatika
Rudy Ariyanto, S.T., M.CS	Ir. Deddy Kusbianto P. M. MKom
NIP. 19711110 199903 1 002	NIP. 19621128 198811 1 001

DAFTAR ISI

1. Judul Skripsi

Deteksi Plagiarisme Pada Dokumen Skripsi Berdasarkan Tingkat Kesamaan Dengan Menggunakan Metode *Longest Common Subsequence*

2. Latar Belakang

Di era kemajuan teknologi yang semakin pesat saat ini, tingkat publikasi dan jumlah plagiarisme yang terdeteksi semakin meningkat, situasi seperti ini menghambat perkembangan dari aktivitas kreatif mahasiswa maka diperlukan metode yang akurat untuk mendeteksi plagiarisme. Pada Jurusan Teknologi Informasi (JTI) masih belum ada aplikasi yang dapat digunakan oleh mahasiswa JTI untuk mendeteksi adanya plagiarisme pada dokumen skripsi.

Berdasarkan sumber yang penulis amati pada tahun 2014 penelitian yang dilakukan oleh Aini, Bariroh Isriya Nur dengan judul "Identifikasi Kemiripan Judul Tugas Akhir PSTI Dan PSMI Di Politeknik Negeri Malang" [1]. Dalam penelitianya mengenai judul tugas akhir atau sekripsi yang harus berbeda dari judul – judul yang lain. Algoritma yang digunakan pada penelitian adalah algoritma Nazief & Adriani, algoritma VSM (Vector Space Model) dan algoritma tfidf (Term Frequency – Inversed Document Frequency), dengan algoritma tersebut dihasilkan sebuah aplikasi yang dapat mengidentifikasi dan memberikan informasi kemiripan pada judul tugas akhir atau skripsi.

Penelitian yang dilakukan oleh Widiastuti, Inta pada tahun 2014 yang berjudul "Aplikasi Pendeteksi Kemiripan Dokumen Menggunakan Algoritma Rabin Karp" [2]. Dalam penelitiannya mengenai deteksi kemiripan antara dua dokumen dengan dilakukan perbandingan *pattern* pada dokumen teks, metode yang digunakan untuk mendeteksi kemiripan dokumen adalah Algoritma Rabin Karp. Penelitian ini berhasil menciptakan aplikasi deteksi plagiarisme dokumen yang bekerja dengan baik serta memperlihatkan hasil similarity-nya.

Dikarenakan pada JTI belum ada aplikasi deteksi plagiarisme dokumen skripsi, oleh karena itu dalam skripsi ini penulis akan membuat sistem deteksi plagiarisme dokumen skripsi menggunakan metode *longest common subsequence* (LCS). Proses deteksi menggunakan perbandingan pada dokumen yang di upload mahasiswa dengan *dataset* yang ada pada database. Sebelum dokumen yang di

upload tersebut dikelola menggunakan metode LCS, dokumen harus dilakukan beberapa proses terlebih dahulu yaitu *Case Folding* dan *Tokenezing. Case Folding* adalah proses mengkonversi semua teks dalam dokumen yang diterima ke huruf kecil (*lowercase*) secara keseluruhan. *Tokenezing* adalah memecah sekumpulan karakter dalam suatu teks ke dalam satuan kata. Setelah melewati tahapan tersebut, dokumen akan diolah menggunakan metode LCS. Metode ini mengubah teks menjadi angka, dari angka tersebut dapat diketahui panjang dari LCS, kemudian panjang LCS dari dokumen yang di *upload* dibagi dengan panjang LCS dari *dataset*. Berdasarkan perhitungan menggunakan metode LCS adalah persentase *similarity*. Dengan adanya aplikasi ini, diharapkan dapat membantu untuk mendeteksi adanya plagiarisme dokumen skripsi.

3. Rumusan Masalah

Berdasarkan latar belakang di atas, maka rumusan masalah yang dapat diambil adalah sebagai berikut:

1. Bagaimana menciptakan aplikasi yang dapat mendeteksi adanya plagiarisme pada dokumen skripsi?
2. Bagaimana mendeteksi plagiarisme pada dokumen skripsi menggunakan metode *longest common subsequence*?

4. Batasan Masalah

Agar skripsi penulis yang berjudul Deteksi Plagiarisme Pada Dokumen Berdasarkan Tingkat Kesamaan Dokumen Dengan Metode *longest common subsequence* dapat berjalan sesuai dengan rencana dan tujuan awal, maka penulis memberikan batasan-batasan masalah yaitu :

1. Deteksi yang dilakukan pada file dokumen dengan ekstensi doc/docx.
2. Aplikasi ini hanya digunakan pada lokal Jurusan Teknologi Informasi.
3. Aplikasi ini menggunakan *dataset* lokal Jurusan Teknologi Informasi.
4. Data yang di uji merupakan tugas akhir dan skripsi.

5. Tujuan

Tujuan penelitian ini adalah membuat sistem yang dapat mendeteksi plagiarisme pada dokumen skripsi berdasarkan tingkat kesamaan dengan menggunakan metode *longest common subsequence.*

6. Tinjauan Pustaka

6.1 Penelitian Terdahulu

Pada penelitian yang dilakukan oleh Kensuke Baba, Tetsuya Nakatoh, Toshiro Minami dengan judul "*Plagiarism detection using document similarity based on distributed representation"* dalam jurnal *Procedia Computer Science* 111 (2017)[6]. Penelitian ini membahas tentang perbandingan dari pengembangan metode *longest common subsequence*. Dari penelitian tersebut metode LCS berhasil mendapatkan tingkat akurasi yang tinggi.

6.2 PHP

Menurut Valade (2004:9) PHP akronim dari Hypertext Preprocessor adalah open source yang banyak digunakan sebagai tujuan utama *scripting language*. Di desain untuk digunakan pada pengembangan *website*. PHP berawal dari *personal home page tools*, yang di kembangkan oleh Rasmus Lerdorf untuk membantu *user* dengan *web page tasks*. PHP dibuktikan sangat berguna dan popular serta secara bertahap berkembang untuk menjadi *full-featured language*.

6.3 Codeigniter

Menurut Blanco & Upton (2009:7) CodeIgniter adalah *framework* open source yang powerful sebahai PHP *framework* yang mudah dikuasai, codeigneter dibangun untuk PHP programmers yang membutuhkan toolkit sederhana dan baik untuk membuat full-featured web applications. CodeIgniter adalah MVC *framework* yang di *design* untuk mempermudah penggunanya.

6.4 NodeJS

Menurut jurnal yang ditulis oleh Iqbal, Husni dan Studiawan (2012) Node.js adalah sistem perangkat lunak yang didesain untuk pengembangan aplikasi *web*. Node.js ditulis dalam bahasa *JavaScript* menggunakan basis event dan *asynchrounous* I/O. Tidak seperti kebanyakan Bahasa *JavaScript* yang ijalankan pada *web browser*, Node.js dieksekusi sebagai aplikasi server. Aplikasi ini terdiri dari *V8 JavaScript Engine* buatan Google dan beberapa modul bawaan yang terintegrasi.

Dalam buku Teixeira (2013,pV), *server side javascript* sudah beredar dari beberapa tahun lalu. Dalam versi serbelumnya berfokus pada menerjemahkan dari platform seperti Ruby, Phyton,PERL kedalam *Javascript*. Dengan menggunakan Node.js kita dapat membuat *programming model* yang mudah dengan skala server yang cukup besar dengan mudah. Tiga kelebihan Node.Js. Yaitu :

1. *Node is Easy* – Node membuat I/O pemogramman menjadi lebih mudah dan dimengerti dari sebelumnya.
2. *Node is Lean* – Node tidak mencoba untuk menyelesaikan semua masalah, tetapi ini bergantung dengan pondasi internet protocol yang menggunakan fungsi API.
3. *Node does note compromise* – Node tidak mencoba untuk jalan dengan software yang sudah *out of date*. Karena Node.Js memberikan tampilan baru yang *fresh.*

6.5 *Hypertext Markup Language* (HTML)

Menurut Shelly, Woods, & Dorin (2008:8) HTML adalah bahasa penulisan yang digunakan untuk membuat dokumen pada halaman website. HTML menggunakan sekumpulan instruksi khusus, yang dikenal dengan *tags* atau *markup* untuk mendefinisikan struktur dan susunan dari *web document* dan menetapkan apa yang akan ditampilkan pada *browser*

6.6 *Cascading Style Sheets* (CSS)

Menurut Collison, Budd, & Moll (2009:245) Dibandingkan dengan *programming language* lainnya, CSS merupakan bahasa yang mudah dimengerti karena kumpulan kode-kode yang berurutan dan saling berhubungan satu sama lain

untuk mengatur tampilan suatu HTML memiliki logika yang tidak sulit. Namun banyak kesulitan yang ditemui ketika melakukan *test* pada *browser* yang berbeda. *Browser bug* dan penamaan yang tidak konsisten sering kali menjadi masalah utama bagi kebanyakan CSS *developers*.

6.7 Bootstrap

Bootsrap adalah platform yang dikembangkan oleh tim twitter. Pertama kali muncul pada ajang *hackweek* dan kini sudah popular dan terus di kembangkan. Platform ini hanya menggunakan coding CSS dan *JavaScript* namun tetap bisa membuat *website* yang *powerfull* mengikuti perkembangan *browser*. *Website* yang menggunakan *bootstrap* akan menjadi *website* yang *fleksibel* dengan penataan *layout* standard dari bootstrap yang dinamakan *grid*, sehingga nyaman dan tentu saja cepat. (Muhammad Hidayatullah, 2013).

6.8 *Javascript*

Menurut Flanagan (2011:1) *Javascript* adalah bahasa pemrograman untuk *web*. *Javascript* merupakan *high-level, dynamic, untyped-intepreted programming language* yang cocok dengan *object-oriented* dan *functional programming style*. *Syntax* pada *Javascript* berasal dari Java namun sangat berbeda dengan *Java programming language*. Seiring dengan berjalannya waktu, *scripting language Javascript* menjadi lebih kuat, efisien dan banyak versi – versi baru yang bermunculan dari para pengembangnya.

6.9 *Model View Controller* (MVC)

Menurut Mackey (2010:295) MVC atau *Model View Controller* memiliki arti sebagai berikut:

- *Model* merupakan bagian yang berinteraksi dengan *database*.
- *View* merupakan *pages* yang tampil di halaman *web*.
- *Controller* akan mengatur interaksi antar *pages*/ mengkontrol hubungan *view* dan *model*.

Keuntungan MVC adalah sebagai berikut:

- *Division/testability*
  Cotrollers mengatur interaksi antara *User Interface* dan data (*model*).

- *Flexibility*

  Setiap individual *layer* mudah dibuang tanpa mempengaruhi *layer* lainnya karena dapat di *costumize* sehingga *user interface* dapat diganti atau menggunakan *database* yang berbeda.

- *Maintability*

  Walaupun dapat di *costumize*, tapi *projects* harus tetap memiliki code yang teratur. Oleh karena itu, *project structure* biasanya kaku dan *new developers* harus dapat memahami arsitekturnya dengan cepat.

6.10 MySQL

Menurut King et.al (2009:4) MySQL adalah *open source database product* yang mendukung *key subsets* SQL di Linux dan Unix *systems*. MySQL digunakan secara gratis dan hanya menghabiskan sedikit biaya bagi periklanannya namun sangat populer. Tidak seperti *commercial database*, MySQL sangat terjangkau dan mudah digunakan.

Fitur yang terdapat di MySQL antara lain:

- *Openess*

  MySQL sangat terbuka dalam setiap hubungan. SQL *dialect* yang digunakan adalah ANSI SQL2 sebagai pembangunnya. *Database engine* menjalankan *platform* yang tidak terhitung, termasuk Windows 2000, Mac OS X, Linux, FreeBSD, dan Solaris. Jika *binary* tidak tersedia untuk *platform*, user dapat mengaksesnya dari *source* untuk di *compile* ke *platform* tersebut.

- *Application Support*

  MySQL memiliki API untuk semua bahasa pemrograman. *User* bisa menulis aplikasi *database* yang mengakses MySQL di C, C++, Eiffel, Java, Perl, PHP, Phyton, dan Tcl.

- *Cross-database joins*

  User bisa mengkonstruksi MySQL queries yang bisa menggabungkan tabel dari database yang berbeda.

- *Outer join support*

MySQL mendukung kiri dan kanan outer joins menggunakan ANSI dan sintaks ODBC.

6.11 Metode *Longest Common Subsequence*

Dalam permasalahan matematika, sebuah *subsequence* adalah sebuah *sequence* yang diturunkan dari *sequence* lain dengan menghapus beberapa elemen tanpa mengganti urutan dari sisa element tersebut. Sebagai contoh, sequence "A B D" merupakan *subsequence* dari *sequence* "A B C D E". Secara singkat, *subsequence* merupakan bagian dari sebuah *sequence*.

Longest Common Subsequences merupakan sebuah metode untuk mencari *sequence* terpanjang yang sama, sebuah basis dari pemrograman untuk perbandingan data komputer. Pada kasus identifikasi tipe *file*, *Longest Common Subsequences* dapat digunakan untuk membandingkan *sequence hex number* beberapa *file*. Secara umum, *Longest Common Subsequence* adalah sebuah metode untuk mendapatkan *sequence* terpanjang dengan membandingkan beberapa *sequence*.

Pengaplikasian metode *Longest Common Subsequences* dapat digunakan untuk menyelesaikan berbagai masalah, contoh permasalahan yang dapat diselesaikan metode LCS mencakup :

1. *Sequence* dari DNA atau gen yang dapat direpresentasikan sebagai deretan 4 huruf ACGT. Pada saat seorang ahli biologi yang meneliti tentang gen, bila dia menemukan suatu *sequence* yang baru pada gen tersebut, biasanya mereka akan mencari tahu *sequence* lainnya yang mirip dengan *sequence* baru tersebut.
2. Pada program UNIX, „diff" digunakan untuk membandingkan dua file yang sama dengan versi yang berbeda, untuk menemukan perubahan yang terdapat pada file tersebut. Hal tersebut dilakukan dengan cara menemukan *Longest Common Subsequences* dari kedua *file* tersebut, dan menemukan *subsequence* yang tidak berubah, sehingga dapat ditemukan baris *subsequence* yang berbeda atau diganti.
3. Banyak *text* editor seperti "emacs" menampikan bagian dari sebuah *file* ke monitor, dan setiap kali *file* tersebut diganti, maka monitor akan ter-update. Program tersebut ingin mengirimkan teks sesedikit mungkin ke terminal

sehingga proses *update* tersebut dapat ditampilkan secara benar. Diperlukan metode *Longest Common Subsequences* sehingga karakter-karakter pada *file* yang tidak berubah tetap tinggal di terminal dan hanya mengirimkan karakter-karakter yang berubah saja (University of California, 1996).

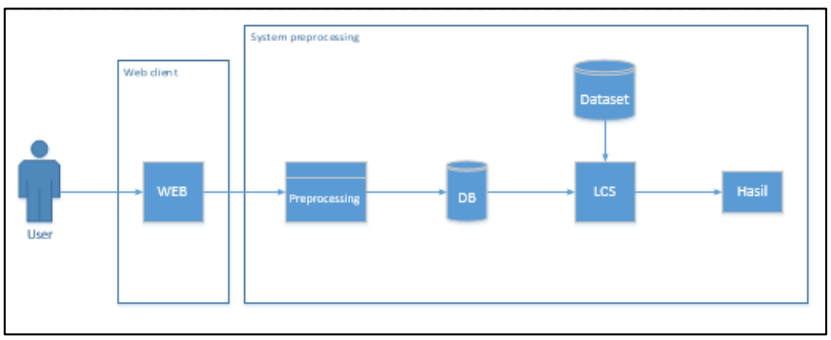
7. Metodologi Penelitian

7.1 Tahap Pertama

Tahapan awal yang penulis lakukan untuk melaksanakan penelitin ini yaitu mencari referensi berupa jurnal. Jurnal yang berjudul *Plagiarism detection using document similarity based on distributed representation* tersebut penulis gunakan sebagai referensi utama[4] yang kemudian penulis download dari website https://www.sciencedirect.com/ beserta jurnal – jurnal pendukung lainnya. Setelah referensi berhasil dipelajari kemudian mengumpulkan *dataset* dari dokumen – dokumen skripsi terdahulu, *dataset* tersebut berupa dokumen word skripsi mahasiswa yang berekstensi doc/docx, dari dokumen tersebut akan dilakukan proses konversi dengan mengambil teks dan menghilangkan semua elemen lain seperti gambar dan simbol – simbol, kemudian teks tersebut dilakukan *Case Folding* dan *Tokenezing* dan di simpan kedalam *database* sebagai *dataset*. Setelah mempelajari referensi diatas dan mendapatkan datasetnya maka penulis memutuskan untuk menggunakan metode *longest common subsequences* (LCS) sebagai metode untuk mencari kesamaan dari dokumen yang di uji.

7.2 Tahap Kedua

Tahapan selanjutnya akan digambarkan pada arsitektur sistem, seperti pada gambar 1.



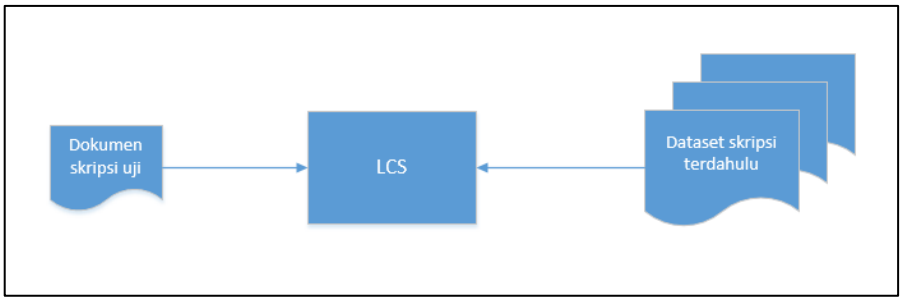
Gambar 1 Arsitektur sistem

Pada tahapan ini ada beberapa proses yang akan dilalui dokumen skripsi sebelum sampai pada proses LCS.

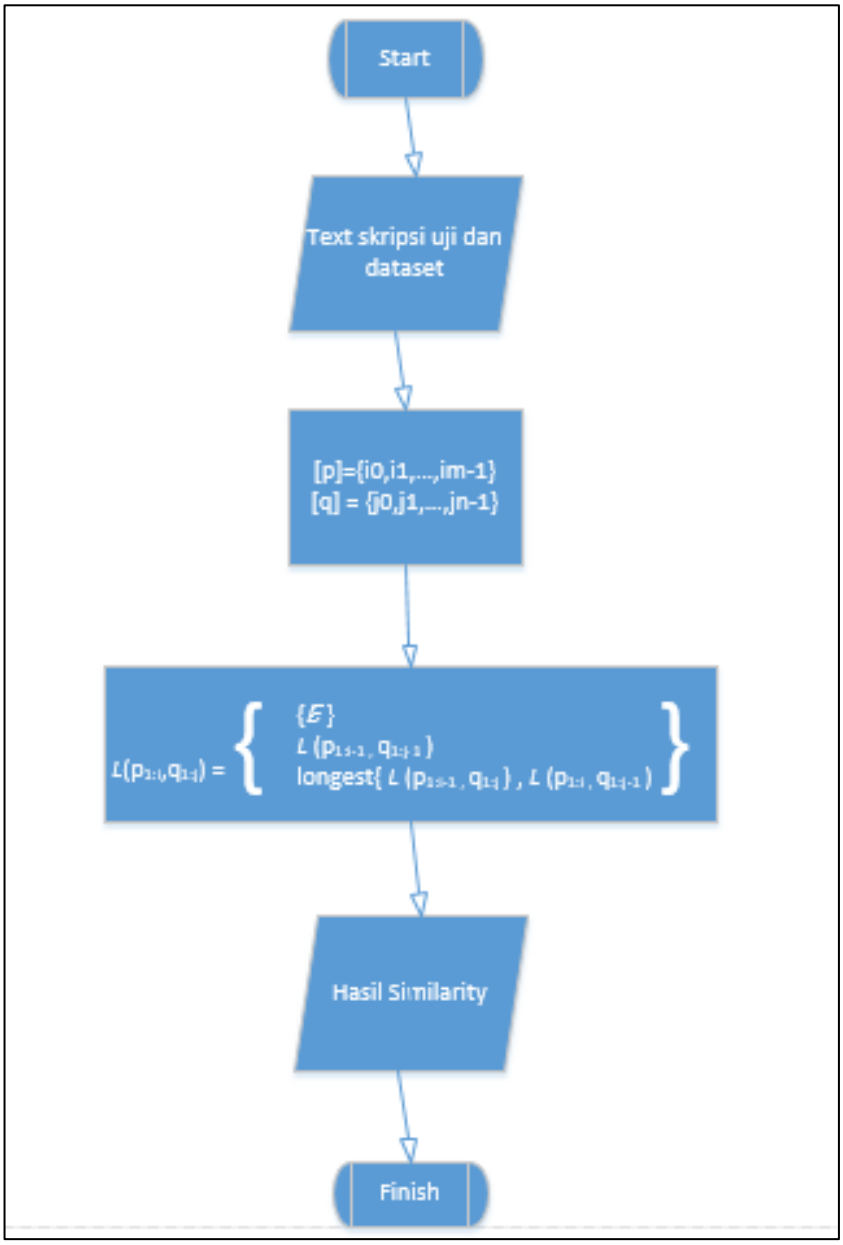
Tahap pertama mahasiswa akan *login* pada web *client*, jika mahasiswa belum terdaftar dapan melakukan *registrasi* terlebih dahulu. Setelah *login* mahasiswa akan diarahkan pada halaman *dashboard* dengan beberapa fitur yang sudah disediakan, seperti edit profil, ganti *password*, *upload* dokumen skripsi. Hal yang harus dilakukan untuk menjalankan aplikasi ini adalah mahasiswa mengupload dokumen skripsinya pada fitur *upload* dokumen. Setelah proses *upload* selesai mahasiswa diharapkan menunggu selama 12 jam hingga 1 hari. Pada pembuatan aplikasi web *client* penulis akan menggunakan *framework* codeigneter.

Tahap kedua dokumen skripsi yang di upload akan masuk ke tahap *preprocessing.* Pada tahap ini dokumen skripsi akan di *genetate* dan hanya di ambil teksnya saja. Kemudian teks tersebut di proses dengan *Case Folding* untuk mengkonversi semua teks menjadi huruf kecil (*lowercase*) secara keseluruhan, proses ini menangani apabila terdapat teks yang tidak konsisten seperti s*entence case*, *uppercase*, *capitalize each word*, dan *Tokenezing* digunakan untuk memecah sekumpulan karakter dalam suatu teks ke dalam satuan kata. Dari proses tersebut terbentuklah teks dengan huruf kecil secara menyeluruh, kemudian teks tersebut disimpan kedalam database.

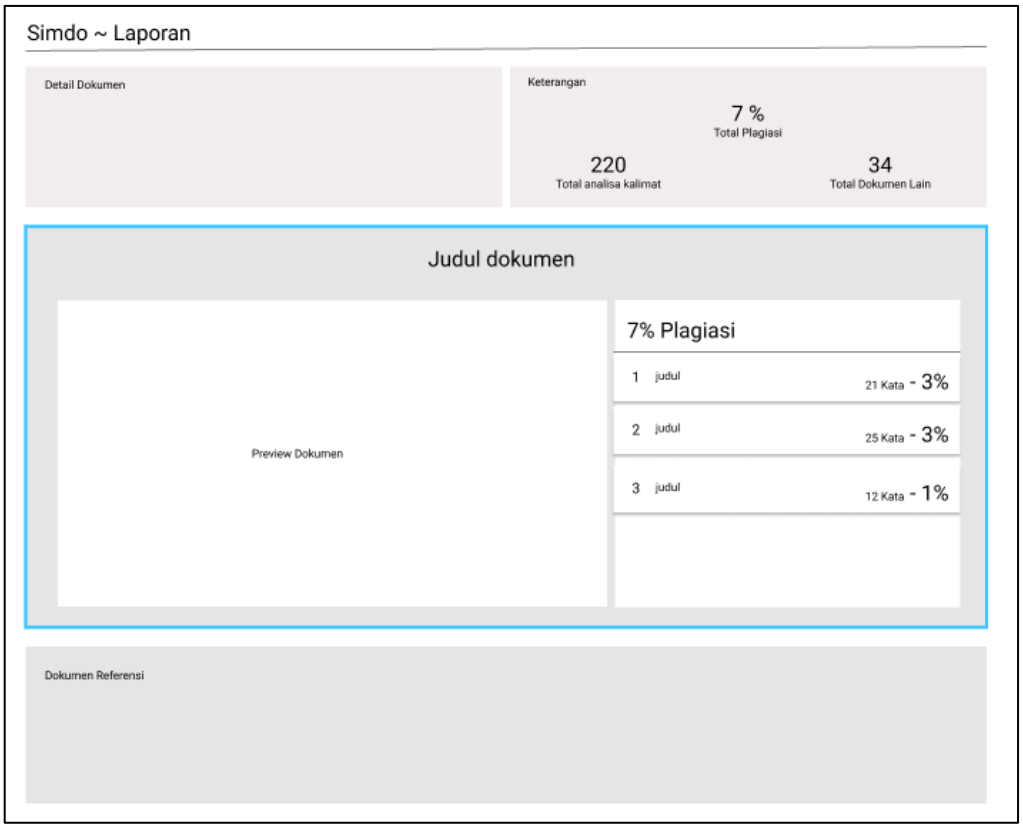
Tahap ketiga adalah proses LCS akan mengambil teks skripsi dari *database* yang sudah dilakukan beberapa proses, kemudian teks skripsi uji akan dilakukan perbandingan dengan *dataset* yang sudah ada pada *database dataset*. Skripsi uji akan dibandingkan dengan banyak teks skripsi terdahulu yang disimpan pada *database dataset* ,seperti pada gambar 2. Hasil yang dikeluarkan berupa persentase plagiarisme dari skripsi yang di uji dengan beberapa *dataset* skripsi – skripsi terdahulu dan akan dikirimkan kepada web *client* untuk ditampilkan kembali pada halaman web sebagai hasil *similarity* dokumen, seperti pada gambar 4 dan 5. Pada proses tahap dua dan tahap tiga akan dibangun menggunakan nodeJS.



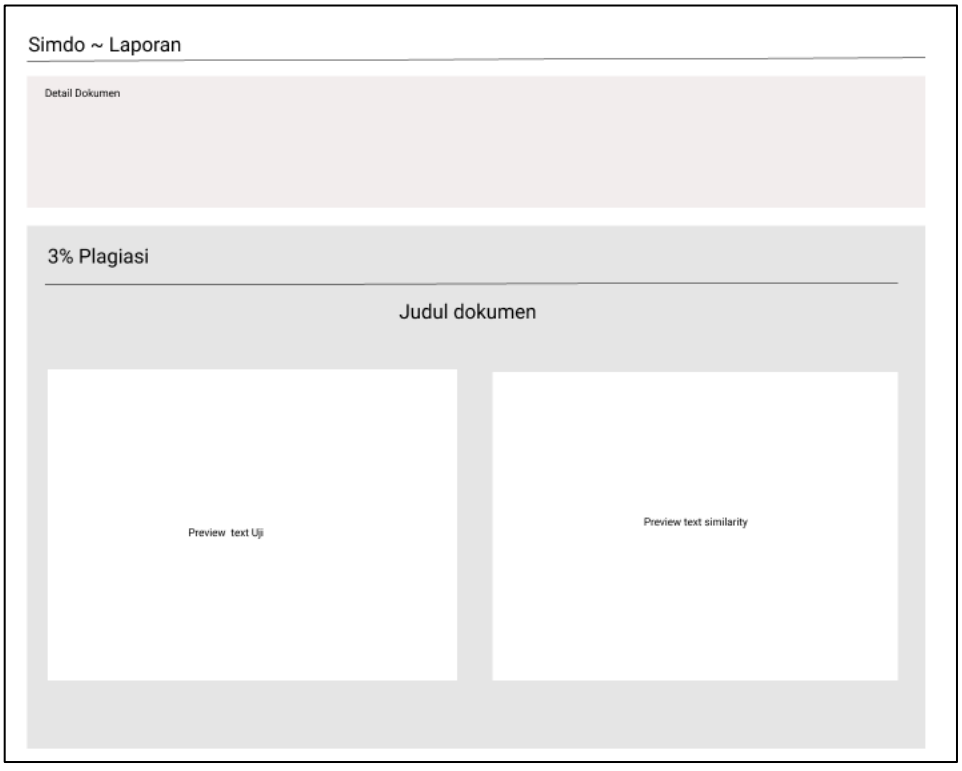
Gambar 2 Proses kerja Algoritma LCS.



Gambar 3 Flowchart LCS



Gambar 4 Hasil *Similarity*



Gambar 5 Menampilkan teks plagiasi

7.3 Tahap Ketiga

Menguji sistem merupakan tahapan yang dilakukan ketika sistem sudah menjadi perangkat lunak yang siap pakai. Pengujian yang dilakukan menggunakan du cara yaitu *White Box, Black Box*, pengujian arsitektur untuk memastikan perangkat lunak yang sudah jadi sesuai atau tidak dengan tujuan awal aplikasi ini dibuat (manual) serta pengujian tingkat keakurasian dari hasil perangkat lunak dengan menggunakan metode pengujian *recall precision*. Jika perangkat lunak sudah sesuai dengan tujuan awal maka sistem sudah dapat digunakan. Namun jika perangkat lunak belum sesuai dengan tujuan awal maka harus mengulang pada tahap kedua, kemudian dilakukan pengujian ulang.

8. Relevansi

No	Uraian	*Warsaw University of Technology*	*University of Malaya*	*Armed Forces Medical Collage*	*Fujitsu Laboratories*
1	Implementasi kontrol	*integrated-circuit layouts*	*integration of semantic relations between words and their syntactic composition*	*Author and editor in preventing/ avoiding plagiarism*	*Accurate methods for plagiarism detection from documents with longest common subsequence*
2	Tahun penelitian	2014	2015	2016	2017
3	Simulasi kontrol pada	*Layout* website	akademisi, penelitian ilmiah, jurnalisme	*Submitted manuscript*	*plagiarism detection with document data*

9. Sistematika Penulisan Laporan

Uraian dalam laporan Skripsi penulis menyusun dengan Sistematika penulisan sebagai berikut:

BAB I : Pendahuluan berisikan tentang latar belakang, rumusan masalah, batasan masalah, tujuan dan manfaat.

BAB II : Landasan teori berisikan tentang tinjauan pustaka dari aplikasi yang penulis buat.

BAB III : Berisi mengenai tahapan yang dilakukan untuk menyelesaikan masalah pada tugas akhir yang bersumber dari proses dalam perencanaa tugas akhir. Metode penelitian berisi urauian tentang metode pengmbilan data, metode pengembangan sistem, fase-fase pengembangan sistem.

BAB IV : Analisa dan Perancangan berisikan tentang analisa sistem aplikasi dan perancangannya.

BAB V : Implementasi berisikan penerapan/implementasi dari aplikasi yang telah penulis buat. Mulai dari implementasi proses dan implementasi data.

BAB VI : Pengujian dan Pembahasan berisikan tentang pengujian proses serta analisa dari hasil proses tersebut.

BAB VII : Kesimpulan berisikan tentang kesimpulan dan saran.

10. Jadwal Kegiatan

Tabel 2. Jadwal Kegiatan

Kegiatan	Desember				Januari				Februari				Maret				April			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1. Pengajuan Judul																				
1.1. Pengumpulan Usulan Judul																				
1.2. Seleksi Judul																				
2. Pengajuan Proposal																				
2.1. Pembuatan Proposal																				
2.2. Pengumpulan Proposal																				
2.3. Ujian Proposal																				
3. Studi Literatur																				
3.1. Mempelajari pustaka sesuai topik																				
3.2. Mempelajari konsep sistem																				
3.3. Mempelajari metode yang digunakan																				
3.4. Mengumpulan data file skripsi																				
4. Desain Sistem																				
4.1. Desain Usecase Diagram																				
4.2. Desain Activity Diagram																				
4.3. Desain Class Diagram																				
4.5. Pembuatan Mockup Sistem																				
5. Implementasi																				
5.1. Coding UI																				
5.2. Coding algoritma LCS																				

5.3. Coding proses deteksi plagiarisme																				
5.4. Coding input file skripsi																				
5.5. Coding Output deteksi plagiarisme																				
5.6. Sync hasil output dengan input																				
6. Testing																				
6.1. Pengujian fungsional sistem																				
6.2. Pengujian performa sistem																				
7. Analisa Hasil Laporan dan Kesimpulan																				
8. Penulisan Laporan Skripsi																				

DAFTAR PUSTAKA

[1] Aini, Bariroh Isriya Nur, Identifikasi Kemiripan Judul Tugas Akhir PSTI Dan PSMI Di Politeknik Negeri Malang, Malang : Politeknik Negeri Malang (2014).

[2] Widiastuti, Inta, Aplikasi Pendeteksi Kemiripan Dokumen Menggunakan Algoritma Rabin Karp, Malang : Politeknik Negeri Malang (2014).

[3] L.K.BurdineBAM.B.de Castro MaymoneMD, DScN.A.VashiMD "*Self-plagiarism in scientific writing!"* International Journal of Women's Dermatology (2018), doi: https://doi.org/10.1016/j.ijwd.2018.10.002.

[4] Asad Abdi, Norisma Idris, Rasim M. Alguliyev, Ramiz M. Aliguliyev "*Plagiarism detection using linguistic knowledge*" Expert Systems With Applications (2015), doi: https://doi.org/10.1016/j.eswa.2015.07.048

[5] Col Jyotindu Debnath "*Plagiarism: A silent epidemic in scientific writing – Reasons, recognition and remedies" Medical Journal Armed Forces India.* (2016), doi: https://doi.org/10.1016/j.mjafi.2016.03.010.

[6] Kensuke Babaa, Tetsuya Nakatoh, Toshiro Minami "*Plagiarism detection using document similarity based on distributed representation*" 8th International Conference on Advances in Information Technology, IAIT2016, 19-22 December 2016, Macau, China, doi: https://doi.org/10.1016/j.procs.2017.06.038.

[7] L.K. Burdine, BA, M.B. de Castro Maymone, MD, DSc, N.A. Vashi, MD "*Text recycling: Self-plagiarism in scientific writing" International Journal of Women's Dermatology* (2018), doi: https://doi.org/10.1016/j.ijwd.2018.10.002.

[8] Dominikus Damas Putranto "Pengkajian Masalah *Longest Common Subsequences"* (2008) http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2007-2008/Makalah2008/MakalahIF2251-2008-047.pdf , diakses pada (3 Desember 2018 16:15 WIB)

[9] Wagner RA,Fischer MJ. "*The string-to-string correction problem*". Journal of ACM(JACM). 1974; 21 (1):168–173

[10] Pasma Cadea Mikha, Pengembangan Sistem Pendeteksi Kemiripan Karya Pada Inaicta 2013, Malang : Politeknik Negeri Malang (2014).

[11] Kharismadita Paratisa, Implementasi Tokenizing Plus Pada Sistem Pendeteksi Kemiripan Jurnal Skripsi Di Program Studi Teknik Informatika Politeknik Negeri Malang, Malang : Politeknik Negeri Malang (2015).

[12] Roshinta Trisna Ali, Analisis Skema-Skema Kemiripan Vektor Pada Sistem Penilaian Essay Online, Malang : Politeknik Negeri Malang (2016).

[13] Nurhadi Riza A, Pengembangan Data Uji Sistem Komputasi Kemiripan Teks Secara Semantik Berbahasa Indonesia, Malang : Politeknik Negeri Malang (2016).