BAB III. METODOLOGI PENELITIAN

Dalam metodologi penelitian ini akan dijelaskan proses tahap-tahapan yang dilakukan untuk membangun aplikasi Deteksi Plagiarisme Dokumen Skripsi Menggunakan Metode *Longest Common Subsequence* dijabarkan sebagai berikut :

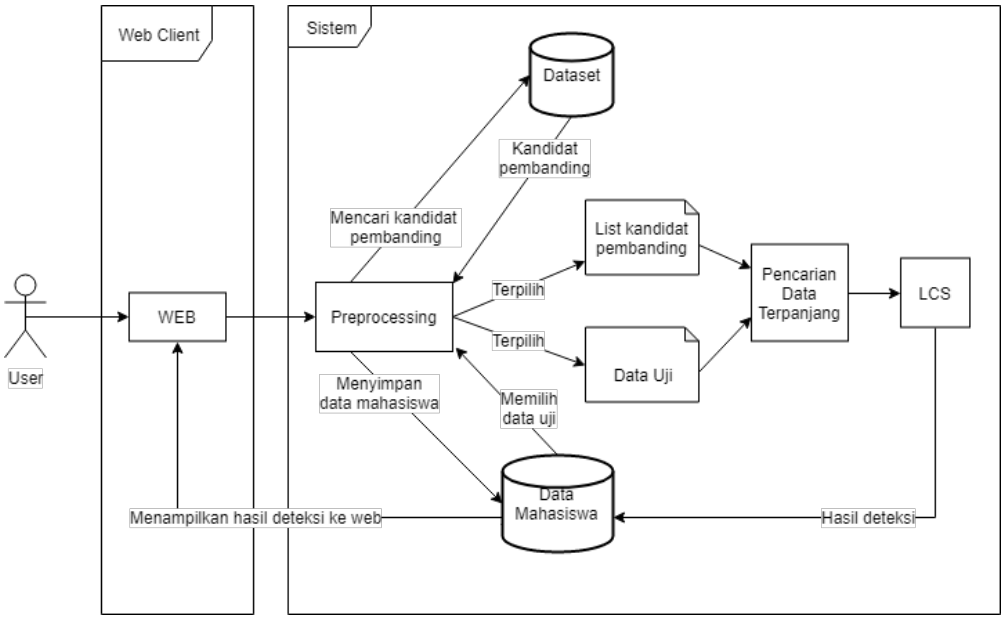
3.1 Metode Pengambilan Data

Metode pengambilan data yang dilakukan untuk melaksanakan penelitin ini yaitu mencari referensi berupa penelitian terdahulu dan jurnal. penelitian yang berjudul Aplikasi Pendeteksi Kemiripan Dokumen Menggunakan Algoritma Rabin Karp tersebut penulis gunakan sebagai referensi utama. Setelah referensi berhasil dipelajari kemudian mengumpulkan *dataset* dari dokumen – dokumen tugas akhir dan skripsi terdahulu, *dataset* tersebut berupa dokumen word atau pdf skripsi mahasiswa, dari dokumen tersebut akan dilakukan proses konversi dengan mengambil teks dan menghilangkan semua elemen lain seperti gambar dan simbol – simbol, kemudian teks tersebut di simpan kedalam *database* sebagai *dataset*, sehingga penulis memutuskan untuk menggunakan metode *Longest Common Subsequences* (LCS) sebagai metode untuk mencari kesamaan dari dokumen yang di uji. Metode tersebut diambil dari referensi jurnal yang berjudul *Plagiarism detection using document similarity based on distributed representation*.

3.2 Metode Pengolahan Data

Data yang didapat dari data tugas akhir dan skripsi pada Jurusan Teknologi Informasi merupakan data yang diambil secara manual dari website digilib.jti.polinema.ac.id. data tersebut akan dijadikan sebagai *dataset* sistem, data tersebut perlu dilakukan pengolahan agar mempermudah dalam proses deteksi oleh sistem, pengolahan yang dilakukan yaitu ketika proses *upload* data tugas akhir dan skripsi akan dilakukan *convert* dari dokumen pdf/docx/doc menjadi teks, hal ini dilakukan untuk mengambil teks didalam dokumen, sehingga gambar dan objek-objek lain tidak diambil, setelah berhasil mengambil keseluruhan teks kemudian data disimpan ke *database,* proses *convert* ini juga berlaku ketika proses *upload* dokumen yang akan diuji, proses ini disebut dengan *preprocessing*. Ada juga yang terdapat dalam proses *preprocessing* selain *convert* dokumen adalah *filter* data-data

pada dataset berdasarkan kata kunci yang akan menjadi kandidat pembanding. Tahapan tersebut akan digambarkan pada arsitektur sistem, seperti pada gambar 3.1.



Gambar 3. 1 Arsitektur Sistem

3.2.1 *Preprocessing*

Tahapan dalam *preprocessing* sebagai berikut :

1. *Convert* Dokumen Menjadi Teks

   Convert dokumen bertujuan untuk mempermudah dalam penyimpanan ke *database* dan meminimalisir penggunaan logika yang terlalu banyak, sehingga proses ini dapat mempersingkat waktu karna ketika proses deteksi dijalankan tidak perlu *convert* dokumen berulang-ulang.

2. *Filter* Kandidat Pembanding

   Filter Kandidat Pembanding merupakan sebuah fungsi yang terdapat didalam sistem yang bertujuan untuk mencari data-data didalam *dataset,* pencarian dilakukan berdasarkan kata kunci yang sesuai dengan data uji. Sehingga data uji nantinya dilakukan proses perbandingan dengan kandidat pembanding saja, karna data yang tidak memiliki kata kunci yang sama tidak akan diambil menjadi kandidat pembanding. Proses ini memiliki dampak yang besar karna sistem tidak perlu melakukan

perbandingan data uji dengan data-data pada *dataset,* serta dapat mempersingkat waktu yang dibutuhkan dalam proses deteksi menggunakan LCS.

3.2.2 Pencarian Dokumen Terpanjang

Pencarian dokumen terpanjang merupakan proses yang dilakukan sebelum menjalankan LCS, karena data parameter yang dibutuhkan LCS, parameter 1 harus lebih panjang dari parameter 2, sehingga dibuat sebuah fungsi pencarian data terpanjang untuk mempersiapkan variabel yang akan dimasukkan pada parameter LCS. Sehingga sistem ini benar-benar berjalan secara dinamis.

Jika X > Y		Jika Y > X
Maka	Atau	Maka
S1 = X		S1 = Y
S2 = Y		S2 = X

Keterangan :

S1 = variabel yang menyimpan yang panjang

S2 = variabel yang menyimpan lebih pendek

X = data uji

Y = kandidat pembanding

3.2.3 *Longest Common Subsequences* (LCS)

Longest Common Subsequences merupakan cara yang digunakan untuk menghitung dan mencari setiap rangkaian kata dalam deteksi plagiarisme. Pencarian ini dilakukan dengan cara membuat tabel matriks yang menyimpan hasil penghitungan LCS, matrik tersebut akan dibuat dengan rumus :

$$\text{tab}[i, j] = \begin{cases} \text{tab}[i-1, j-1] + 1 , S1[i] = s2[j] \\ \max(\text{tab}[i-1, j], \text{tab}[i, j-1]) \end{cases}$$

Keterangan :

tab[*i, j*] = panjang LCS yang ditemukan

S1 = data uji

S2 = kandidat pembanding

Karena pada dasar metode LCS hanya mengambil 1 hasil LCS dengan rangkaian kata terpanjang didalam 1 dokumen, maka dibuat sebuah kondisi perulangan untuk menemukan lebih dari 1 rangkaian kata yang plagiat. Kondisi pada perulangan ini dibatasi dengan jumlah kata hasil LCS lebih dari 2 kata, dalam kondisi ini diambil berdasarkan struktur kalimat terbaik adalah kalimat yang memiliki lebih dari 2 rangkaian kata. Dengan adanya kondisi ini akan membuat sistem dapat melakukan pencarian secara berulang-ulang dalam 1 dokumen pembanding, sehingga dalam 1 dokumen pembanding sistem dapat mencari dan mendapatkan lebih dari 1 hasil LCS. Selain itu metode juga dibuat secara dinamis dengan dibuat sebuah kondisi untuk mengetahui jumlah data kandidat pembanding, sehingga sistem dapat berjalan secara berulang-ulang sebanyak data kandidat pembanding.

3.3 Metode Pengujian

Menguji sistem merupakan tahapan yang dilakukan ketika sistem sudah menjadi perangkat lunak yang siap pakai. Pengujian yang dilakukan menggunakan beberapa cara yaitu *White Box, Black Box*, pengujian arsitektur untuk memastikan perangkat lunak yang sudah jadi sesuai atau tidak dengan perencanaan aplikasi (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 dimana letak kesalahan pada aplikasi, kemudian dilakukan pengujian ulang.