Seminar II

RANCANG BANGUN APLIKASI PENDETEKSI KESAMAAN DOKUMEN MENGGUNAKAN METODE *DICE SIMILARITY*

ARIS AKHYAR ABDILLAH

H071171505

Pembimbing Utama : **Dr. Hendra S.Si., M. Kom.**

Pembimbing Pertama : **Edy Saputra R, S. Si., M. Si.**

Penguji : **1. Dr. Muhammad Hasbi, M.Sc**

2. Ir. Eliyah Acantha Manapa Sampetoding, S.Kom., M.Kom.

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

MAKASSAR

2022

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Teknologi merupakan sarana untuk menyediakan barang-barang yang diperlukan untuk mempermudah kegiatan atau kehidupan manusia. Penggunaan teknologi oleh manusia dimulai dengan pengubahan sumber daya alam menjadi alat-alat sederhana. Penemuan di masa lampau tentang kemampuan mengendalikan api telah menaikkan ketersediaan sumber-sumber pangan, sedangkan penciptaan roda telah membantu manusia dalam bepergian dari satu tempat ke tempat lainnya. Perkembangan teknologi terbaru dimasa sekarang seperti mesin cetak, telepon, dan internet telah mengubah manusia untuk bekerja maupun berinteraksi dalam menjalankan kehidupannya.

Dengan adanya teknologi di saat ini, dapat dikatakan pekerjaan manusia semakin mudah. Seperti untuk akses informasi, sekarang sudah banyak sekali informasi yang tersebar di internet dan hampir semua yang dicari dapat ditemukan. Tidak seperti dahulu, untuk mengakses informasi, orang biasa pergi ke perpustakaan untuk mendapatkan informasi yang diinginkan. Hal itu memungkinkan karena adanya digitalisasi dokumen yang dulunya hanya berupa *hardcopy* (salinan cetak) menjadi *softcopy* (salinan digital). Dengan hal tersebut kita bisa mengakses semua dokumen kapan saja dan di mana saja. Dengan semua kemudahan yang diberikan teknologi seperti contoh di atas, timbul suatu masalah yaitu mudahnya dilakukan plagiarisme yang merupakan suatu tindakan yang tidak baik.

Plagiarisme berasal dari bahasa Latin yaitu "*plagiare*" yang berarti mencuri. Plagiarisme berasal dari kata plagiat yang berarti pengambilan karangan (pendapat dan sebagainya) orang lain dan menjadikannya seolah-olah karangan (pendapat dan sebagainya) sendiri, misalnya menerbitkan karya tulis orang lain atas nama dirinya sendiri. Sehingga dapat diartikan plagiarisme merupakan tindakan mencuri gagasan hasil penelitian orang lain, untuk kemudian disajikan seolah-olah milik sendiri (Ridhatillah, 2003). Plagiarisme yang marak dilakukan di kalangan mahasiswa membuat mahasiswa menjadi malas berpikir dan mengembangkan kemampuan

sebagai kaum intelektual. Moral mahasiswa akan luntur karena dengan melakukan plagiarisme pemikiran mereka tidak dapat berkembang dengan maksimal. Sebab mahasiswa cenderung mencari kemudahan dengan mengambil karya orang lain dan mengakui sebagai karya pribadi (Arista & Listyani, 2015). Tindakan plagiarisme merupakan salah satu tindakan yang melanggar hak cipta. Hak Cipta itu sendiri merupakan hak eksklusif untuk Pencipta ataupun penerima hak buat mengumumkan ataupun perbanyak Ciptaannya ataupun membagikan izin buat itu dengan tidak kurangi pembatasan- pembatasan bagi peraturan perundang-undangan yang berlaku. Jika terjadi pelanggaran tersebut, dapat dikenai pelanggaran hak cipta di Pasal 72 ayat UUHC dengan dipidana dengan pidana penjara pendek selama 1 bulan serta / ataupun denda sangat sedikitnya Rp1.000.000,00, ataupun pidana penjara lama 7 tahun serta / ataupun denda sebanyak Rp5.000.000.000,00 (Lopes, 2013)

Untuk mengurangi hal tersebut, dapat dilakukan beberapa metode untuk mencegah plagiarisme seperti menggunakan cara manual ataupun menggunakan teknologi yang ada. Jika menggunakan cara manual, dilakukan dengan membandingkan sebuah dokumen yang akan dicek dengan dokumen yang akan dijadikan perbandingan dengan melihat kata yang digunakan ataupun tata penulisan dokumen tersebut. Cara ini kurang efektif karena jumlah dokumen yang akan dijadikan perbandingan jumlah tidak sedikit bisa mencapai ratusan bahkan ribuan dokumen lebih. Sedangkan untuk cepat menggunakan mesin sebagai pembanding.

Ada beberapa metode yang dapat digunakan seperti *Cosine Similarity*, TF-IDF, *Jaccard Similarity*, *Dice Similarity*, *Word2Vec* dan sebagainya. Dengan menggunakan salah satu metode di atas, dapat temukan persentase kesamaan sebuah dokumen. Salah satu metode yang dapat digunakan yaitu dengan metode *Dice Similarity*. *Dice Similarity* merupakan salah satu metode perbandingan antara dokumen dengan membandingkan kata yang ada di antara dokumen tersebut. Melihat permasalahan di atas, maka disusunlah tugas akhir ini dengan judul "RANCANG BANGUN APLIKASI PENDETEKSI KESAMAAN DOKUMEN MENGGUNAKAN METODE DICE SIMILARITY". Dengan adanya penelitian ini, diharapkan dapat membangun sebuah aplikasi Web yang mampu memberikan persentase kesamaan dari sebuah dokumen terhadap dokumen lainnya yang

terdapat di dalam *database* guna mengurangi tindak plagiarisme yang sering terjadi saat ini.

1.2 Rumusan Masalah

Rumusan masalah dalam penelitian ini yaitu bagaimana cara merancang sebuah aplikasi pendeteksi kesamaan teks berbasis *website* dengan mengimplementasikan metode *Dice Similarity* serta bagaimana cara menguji aplikasi tersebut menggunakan sebuah dokumen ?

1.3 Tujuan Penelitian

Tujuan penelitian dalam penelitian yaitu merancang sebuah aplikasi pendeteksi kesamaan teks berbasis *website* dengan mengimplementasikan metode *Dice Similarity* serta menguji aplikasi tersebut menggunakan sebuah dokumen.

1.4 Manfaat Penelitian

Penelitian ini merupakan pengimplementasian metode *Dice Similarity* pada sebuah aplikasi berbasis web guna mengetahui persentase kesamaan sebuah dokumen dengan dokumen lainnya sehingga dapat digunakan untuk mengurangi tindak plagiarisme dengan mudah.

1.5 Batasan Masalah

Adapaun beberapa batasan masalah pada penelitian ini yaitu :

1. Sistem ini dibuat menggunakan bahasa pemrograman Javascript dengan *runtime environment* Nodejs, *framework* Express JS, dengan database MongoDB.
2. Menggunakan data berupa jurnal yang diinput oleh user dan memiliki ekstensi pdf (*Portable Document Format*).

BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Dalam Penelitian Mufti Ari Bianto, dkk (2018) dengan judul *Perancangan Sistem Pendeteksi Plagiarisme Terhadap Topik Penelitian Menggunakan Metode K-Means Clustering dan Model Bayesian*, hasil kemiripan diuji berdasarkan kesamaan pola kata dalam kalimat, dokumen dengan nilai *similarity* tertinggi diperoleh satu dokumen dengan persentase kemiripan sebanyak 100 %, dan jika berdasarkan kesamaan *term*, berpengaruh terhadap hasil dokumen mirip yang dihasilkan, sehingga diperoleh dua dokumen mirip. Meskipun uji berdasarkan kesamaan *term*, menghasilkan dokumen mirip yang lebih banyak namun belum cukup akurat menunjukkan adanya plagiasi, karena dalam menentukan plagiasi kesamaan rangkaian kalimat merupakan hal yang penting untuk diperhatikan (Bianto, Rahayu, Huda, & Kusrini, 2018).

Pada penelitian Joko Priambodo (2018) dengan judul *Pendeteksian Plagiarisme Menggunakan Algoritma Rabin - Karb dengan Meotde Rolling Hash*, pendeteksian plagiarisme menggunakan algoritma *Rabin-Karp* dengan metode *rolling hash* dari hasil pengujian 30 dokumen teks yang menghasilkan tingkat akurasi yang terbesar yaitu 47.58 %. Hasil persentase tersebut termasuk dalam kategori tingkat plagiat 15 - 50 %, berarti menandakan dokumen tersebut termasuk plagiat tingkat sedang. Sedangkan tingkat akurasi yang terkecil yaitu 19.28 %, berarti menandakan dokumen tersebut termasuk plagiat tingkat sedang. Selain itu, berdasarkan analisis proses pendeteksian tingkat plagiarisme menggunakan algoritma *Rabin-Karp* dengan metode *Rolling Hash* bisa membaca karakter berupa huruf, simbol seperti titik (.), koma (,), dan lain-lain (Priambodo, 2018).

Kemudian dipenelitian Pavel Stefanovic (2019) dengan judul *The N-Grams Based Text Similarity Detection Approach Using Self-Organizing Maps and Similarity Measures*, dengan pendekatan berdasarkan teks yang dipecah menjadi n-gram dan mengevaluasinya menggunakan *Self-Organizing Maps* (SOM) dan *Similarity Measure*. Deteksi teks serupa dilakukan dalam tiga langkah: (1) konversi kumpulan data teks ke numerik ekspresi menggunakan n-gram; (2) perhitungan

ukuran kesamaan; (3) visualisasi dataset teks menggunakan SOM dan representasi kesamaan di atasnya. Pada langkah pertama, fokus utamanya adalah membuat grup n-gram dari semua dataset. Berbagai jumlah kata dalam *n-gram* dianalisis. Selain itu, filter yang berbeda diterapkan seperti penghapusan angka dan tanda baca, frekuensi kata, transformasi huruf besar, *stemming* algoritma, dll. Analisis menunjukkan filter dan ukuran n-gram mempengaruhi hasil akhir. Untuk ini dataset, ukuran n-gram dipilih dan sama dengan tiga untuk penyelidikan eksperimental. Pada langkah kedua, empat ukuran kesamaan dihitung: *Cosinus*, *Dice*, *Extends Jaccard*, dan *Overlap*. Hasil akhir menunjukkan bahwa persentase kemiripan tertinggi diperoleh dengan menggunakan *overlap*. Tiga nilai ukuran lainnya selalu sama dan lebih kecil. Penggunaan SOM menunjukkan bahwa SOM membantu untuk melihat hasil ringkasan kesamaan semua teks dalam bentuk visual dengan cepat. Dia sangat mudah untuk memahami teks mana yang mirip satu sama lain atau tidak. Dalam kasus kumpulan data yang dianalisis, SOM membantu mendeteksi kesamaan, dan *cluster* yang terbentuk dikorelasikan dengan kategoris yang diberikan deskripsi kumpulan data (Stefanovi, Kurasova, & Štrimaitis, 2019).

Sedangkan, pada penelitian Uswatun Hasanah, dkk (2019) dengan judul *Perbandingan Metode Cosine Similarity dan Jaccard Similarity Untuk Penilaian Otomatis Jawaban Pendek*, dinilai belum mampu memberikan jawaban yang memuaskan i dikarenakan kedua metode hanya menilai kemiripan berdasarkan susunan leksikalnya. Sementara itu, jawaban mahasiswa juga sangat bervariasi dan menggunakan kata-kata yang jauh berbeda dari jawaban kunci, walaupun pada dasarnya memiliki makna semantik yang sama. Oleh karena itu, pada penelitian selanjutnya diperlukan metode lain yang mampu menangani makna semantik pada jawaban. Bagaimanapun, metode *Cosine Similarity* dan *Jaccard Similarity* masih dapat dipertimbangkan untuk menilai jawaban pendek secara otomatis, dengan batasan bahwa pertanyaan yang digunakan mengharuskan jawaban dalam format *keyword* sehingga tidak memunculkan kata-kata lain yang mampu menurunkan nilai kemiripan (Hasanah & Mutiara, 2019).

2.2 Landasan Teori

2.2.1 *Website*

Website adalah kumpulan halaman web yang saling terhubung dan seluruh file saling terkait. Web terdiri dari page atau halaman dan kumpulan halaman yang dinamakan *homepage*. *Homepage* berada pada posisi teratas dengan halaman-halaman terkait berada di bawahnya. Biasanya, setiap halaman di bawah *homepage* (*child page*) berisi *hyperlink* ke halaman lain dalam web (Agung, 2000).

Sejarah *website* dimulai oleh Tim Berners Lee, yang mengembangkan *World Wide Web* (WWW) pada tahun 1989. Pada bulan Oktober tahun 1990, Tim menggagas tiga teknologi dasar untuk membangun sebuah *website*, yaitu HTML, URL, dan HTTP. Awalnya, *website* dirancang hanya untuk berbagi informasi di kalangan para ilmuwan di *Centre Europeen pour la Recherchee Nucleaire* (CERN). Sampai akhirnya, dia melihat potensi *website* sebagai sarana yang bisa digunakan oleh siapa saja untuk berbagai tujuan. Kemudian, dia merilisnya secara resmi pada 6 Agustus 1991.

Website yang pertama dirilis saat itu baru sebatas menampilkan teks sederhana tanpa variasi *font*, video, maupun gambar. Saat ini, *website* sudah berkembang sangat pesat dan menjadikan *website* sebagai salah satu kebutuhan penting baik individu, organisasi, dan perusahaan di seluruh dunia hingga saat ini. Saat ini, website berkembang sangat pesat dengan muncul berbagai jenis *website* seperti pribadi, *e-commerce*, *blog*, dan media sosial.

2.2.2 *Hypertext Transfer Protocol Secure* (HTTPS)

Hypertext Transfer Protocol Secure (HTTPS) adalah sebuah *protocol* komunikasi dalam suatu jaringan internet dengan keamanan yang lebih terjamin. Disebut lebih aman karena suatu perintah atau data yang dikirim melalui HTTPS ini dilindungi dengan sistem enkripsi sehingga menyulitkan *hacker* untuk membobol atau mencurinya (Firmansyah, 2019). HTTPS merupakan tingkatan dari HTTP di mana yang membedakan di HTTPS terdapat *(Secure Socket Layer*) SSL dan (*Transport Layer Security*) TSL yang digunakan untuk mengamankan data yang disimpan atau yang akan dikirim. Menurut SSL Labs pada April 2018, 33.2% dari 1.000.000 situs web teratas Alexa menggunakan HTTPS sebagai *default*,

57.1% dari 137.971 situs web paling populer di Internet memiliki implementasi HTTPS yang aman, dan 70% dari pemuatan halaman (diukur oleh *Firefox Telemetry*) menggunakan HTTPS. Jadi saat ini *website* sekarang sudah mulai beralih dari sebelumnya HTTP ke HTTPS karena di HTTPS lebih aman daripada HTTP yang sudah lama.

2.2.3 Javascript

Javascript adalah sebuah bahasa pemrograman tingkat tinggi yang dinamis, *scripting*, *untyped*, dan *interpreter*. Javascript sendiri dibuat oleh Brendan Eich dari perusahaan Nestcape pada tahun 1994 yang diberi nama Mocha pada saat itu, kemudian berganti menjadi Livescript. Karena saat itu browser yang populer adalah Nestcape, Microsoft berusaha untuk mengalahkan popularitas browser tersebut dengan Internet Explorer dan melakukan *Reverse Engineering* terhadap Livescript dan terciptalah JScript pada tahun 1996. Karena terdapat dua browser yang besar yang berbeda, maka dibuatlah satu standar agar mempermudah pembuatan website saat itu dan dibuatlah ECMAScript. Menurut Douglas Rockford, "*Javascript, JScript, and ECMAScript 3 Silly Name for 1 Silly Language*" yang berarti bahwa ketiga nama tersebut adalah bahasa yang sama yaitu Javascript.

2.2.3.1 Karakteristik Javascript

Untuk Ekstensi dari file Javascriptmenggunakan ekstensi *.js. Contoh namaFile.js. Setiap akhir dari kode Javascriptdapat menggunakan “;” ataupun tidak. Sedangkan *block scope* di Javascript menggunakan { }.

Javascript merupakan salah satu bahasa pemrogram yang *untyped* / *dynamicly typed* yang berarti tidak mendefinisikan terlebih dahulu tipe variabel yang akan didefinisikan. Untuk penamaan variabel, terdapat beberapa *keyword* yaitu var, let, dan const. Untuk var, dan let, nilai variabel dapat berubah atau diisi ulang, sedangkan const nilai variabelnya tidak dapat diubah. Perbedaan var dan let sendiri terletak pada *hoisting*. Sedangkan untuk penamaan variabel, hampir sama dengan beberapa bahasa pemrograman lainnya seperti karakter pertama variabel tidak boleh angka, menggunakan penulisan *camelCase*, dan untuk *keyword* const yang merupakan variabel const biasanya menggunakan *snake_case* dan semuanya huruf kapital. Contoh :

```
var value = 10 // Contoh Penggunaan var
let nilai = 'Delapan' // Contoh Penggunaan let
const PI = 3.14 // Contoh Penggunaan const
```

Javascript memiliki beberapa tipe data seperti *String*, *Number* atau *Integer*, *Boolean*, *Array*, *Object*, dan *Undifined*. Untuk mengetahui tipe data dari sebuah variabel dapat menggunakan *keyword typeof*. Contoh :

```
let name = 'Aris Akhyar Abdillah' // String
let age = 22 // Number atau Integer
let male = true // Boolean
let family = ['Father', 'Mother', 'Son', 'Girl'] // Array
let detail = {
    fullName : 'Aris Akhyar Addillah',
    nim : 'H071171505'
} // Object
let girlfreind = undefined // Undefined
```

Synchronous dan *Asynchronous*, Secara sederhana, *Synchronous* dan *Asynchronous* merupakan tahapan dalam mengeksekusi sebuah kode di mana Synchronous mengeksekusi sebuah kode per baris sesuai urutan kode yang dituliskan. Sedangkan *Asynchronous*, tidak selalu seperti *Synchronous*, tapi melihat waktu proses dari kode tersebut. Penggunaan *Asynchronous* tidak akan menunggu suatu kode selesai dijalankan, tetapi berlanjut ke kode selanjutnya. Contoh :

```
// Menggunakan Synchronous
console.log('What Is Your Name ?')
console.log('Hello World')
console.log('My Name is Aris')
// Output :
// What Is Your Name ?
// Hello World
// My Name is Aris

// Menggunakan Asynchronous
```

```
setTimeout(function() {
  console.log('What Is Your Name ?')
}, 1500)
console.log('Hello World')
setTimeout(function() {
    console.log('My Name is Aris')
}, 1000)
// Output :
// Hello World
// My Name is Aris
// What Is Your Name ?
```

Jika melihat output dari dua program di atas antara menggunakan *Synchronous* dan *Asynchronous* terdapat perbedaan urutan dari hasil *output*, di mana *Synchronous* memiliki *output* sesuai dari urutan baris kode sedangkan *Asynchronous* berbeda di mana *output* yang dihasilkan berdasarkan jumlah waktu pengeksekusian dari kode tersebut yang disimulasikan menggunakan *setTimeout()* dari pengeksekusian paling cepat hingga yang paling lambat.

2.2.3.2 NodeJS

Node.js adalah *runtime environment* untuk Javascript yang bersifat *open-source* dan *cross-platform*. Dengan Node.js kita dapat menjalankan kode Javascript di mana pun, tidak hanya terbatas pada lingkungan *browser*. Seperti diketahui bahwa Javascript hanya dapat berjalan pada sebuah *web browser*, kemudian Ryan Dahl, membuat sebuah *runtime environment* dengan mengeluarkan *engine* Javasscript dari Chrome yaitu *V8 Javascript Engine* menggunakan bahaasa C agar Javascript dapat dijalankan diluar *browser*. Akhirnya tercipta NodeJS pada tahun 2009. Dengan Begitu, Javascript yang sebelumnya hanya bisa di *client side* dengan adanya NodeJS bisa juga di *server side*. Beberapa fitur yang terdapat di NodeJS seperti *Asynchronous & Event-driven, Single Threaded but Highly Scalable* (Anonymous, NodeJS, 2022).

2.2.3.3 ExpressJS

Menurut website resmi dari Express JS adalah "*Fast, unopinionated, minimalist web framework for Nodejs*" (Anonymous, Express JS, 2022). Express JS sendiri merupakan salah satu *web framework* khusus untuk NodeJS di mana kita dapat membuat sebuah *website* yang cepat, sederhana, dengan struktur yang tidak ditentukan atau tergantung dari pengguna ExpressJS sendiri.

2.2.4 *Database*

Database atau basis data adalah kumpulan informasi yang disimpan di dalam komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data (Andaru, 2018). Berdasarkan definisi *database, database* berfungsi untuk menyimpan catatan atau sebuah data pada sebuah penyimpanan yang kemudian akan digunakan pada waktu lainnya. Pada umumnya, database sudah pasti memiliki *key* dan *value*, walaupun istilah ini berbeda ditiap-tiap jenis database yang ada. Adapun beberapa jenis database seperti *Operational Database*, *Database Warehouse*, *Distributed Database*, *Relational Database*, dan *End User Database*.

2.2.4.1 SQL dan NoSQL

SQL dan NoSQL merupakan salah satu contoh dari *Relational Database* yang populer saat ini. SQL (*Structured Query Language*) merupakan bahasa yang digunakan untuk mengelola data secara *relational*. SQL sendiri memiliki ciri yaitu memiliki tabel yang terdiri dengan *row* atau *record* dan *field* yang bisa memiliki relasi dengan tabel lainnya. Untuk tiap data yang ada di dalam tabel, harus memiliki skema yang sama untuk tiap recordnya. Contoh dari SQL seperti MySQL, PostgreSQL, dan MariaDB. Berbeda dengan NoSQL (*Not Only Structured Query Language*) yang merupakan bahasa untuk mengelola data secara *Non Relational*. Berbeda dengan SQL yang menggunakan tabel untuk menyimpan data. NoSQL memilik banyak jenis tempat untuk menyimpan data seperti *Document Database*, *Key-Value Database*, dan *Graph Database*. NoSQL juga tidak memiliki skema sehingga untuk menyimpan data bisa secara flexibel. Contoh dari NoSQL seperti MongoDB, Redis, Neo4j, dan Cassandra.

2.2.4.2 MongoDB

MongoDB merupakan salah satu contoh dari NoSQL yang dikembangkan menggunakan bahasa pemrograman C++ yang rilis pertama kali pada tanggal 11 Februari 2009. MongoDB adalah salah satu contoh *Document Database* yang di mana tiap - tiap datanya merupakan sebuah JSON (*Javascript Object Notation*) atau dalam MongoDB disebut BSON (*Binary JSON*). Hingga saat ini, MongoDB sudah digunakan lebih dari 85 Juta pengguna di seluruh dunia dan sudah banyak perusahaan besar yang menggunakan database ini seperti EBay, Google, Adobe, dan EA (Anonymous, What Is MongoDB ?, 2022). Salah satu contoh data yang terdapat di MongoDB :

```
{
  _id : '91829jw1jooojo2',
  firstName : 'Aris',
  lastName : 'Akhyar',
  age : 20
  hobbies : ['eat', 'drink']
}
```

2.2.5 Jurnal

Jurnal merupakan bagian dari jenis terbitan berseri yang ada diperpustakaan, adapun pengertian jurnal menurut *High Beam* "*Journal is the collection and periodic publication or transmission of news and the result of research through media*", artinya bahwa jurnal merupakan suatu koleksi dan terbitan berkala atau transmisi mengenai berita dan hasil-hasil penelitian mengenai media. Jurnal sendiri terbagi atas dua format yaitu tercetak dan digital (*e-journal*). Untuk format digital jurnal dikemas dalam dua format , yaitu bentuk CD-ROM dan dalam bentuk akses secara *online* melalui *internet*. *E-Journal* dipahami sebagai publikasi ilmiah dalam format elektronik dan mempunyai ISSN (*International Standard Serial Number*) yang format dokumennya menggunakan pdf (Rusydi, 2014).

Penggunaan kata jurnal untuk berbagai bidang juga memberi arti yang bervariasi, misalnya jurnal dalam bidang ekonomi menunjukkan sistem pembukuan

rangkap. Jurnal dalam bidang pelayaran diartikan sebagai *logbook* berarti buku untuk mencatat semua kejadian selama pelayaran. Jurnal sebenarnya merupakan publikasi ilmiah yang memuat informasi tentang hasil kegiatan dalam bidang ilmu pengetahuan dan teknologi minimal harus mencakup kumpulan atau kumulasi pengetahuan baru, pengamatan empiris dan pengembangan gagasan atau usulan. Dengan demikian jurnal merupakan representasi dari pengetahuan baru tentang perkembangan ilmu pengetahuan yang dilaksanakan secara empiris dan biasanya merupakan gagasan yang terbaru.

2.2.6 Plagiarisme

Plagiarisme berasal dari bahasa Latin "*plagiare*" yang berarti mencuri. Plagiarisme berasal dari kata plagiat yang berarti pengambilan karangan (pendapat dan sebagainya) orang lain dan menjadikannya seolah-olah karangan (pendapat dan sebagainya) sendiri, misalnya menerbitkan karya tulis orang lain atas nama dirinya sendiri. Sehingga dapat diartikan plagiarisme merupakan tindakan mencuri gagasan hasil penelitian orang lain, untuk kemudian disajikan seolah-olah milik sendiri (Ridhatillah, 2003).

Tindakan plagiarisme merupakan salah satu tindakan yang melanggar hak cipta. Hak Cipta itu sendiri merupakan hak eksklusif untuk pencipta ataupun penerima hak buat mengumumkan ataupun perbanyak Ciptaannya ataupun membagikan izin buat itu dengan tidak kurangi pembatasan-pembatasan bagi peraturan perundang-undangan yang berlaku. Jika terjadi pelanggaran tersebut, dapat dikenai pelanggaran hak cipta di Pasal 72 ayat UUHC dengan dipidana dengan pidana penjara pendek selama 1 bulan serta / ataupun denda sangat sedikitnya Rp1.000.000,00, ataupun pidana penjara lama 7 tahun serta / ataupun denda sebanyak Rp5.000.000.000,00 (Lopes, 2013).

2.2.7 *Dice Similarity*

Dice Similarity atau *Sørensen–Dice coefficient* merupakan salah satu algoritma yang mengukur kesamaan antara dua set data (Khontoro, Andjarwirawan, & Yulia, 2021). Algoritma ini banyak digunakan dalam validasi algoritma segmentasi gambar yang dibuat dengan *Artificial Intelligence* atau AI, tetapi ini adalah konsep yang jauh lebih umum yang dapat diterapkan pada kumpulan data untuk berbagai

aplikasi termasuk *Natural Language Processing* (NLP). Untuk penggunaan rumus dari *Dice Similarity* menggunakan data *vektor*, dengan A dan B sebagai *vektor*, maka rumusnya yaitu :

$$DSC = \frac{2|A.B|}{|A|^2 + |B|^2} \quad (2.1)$$

2.2.7.1 Implementasi *Dice Similarity*

Untuk pengimplementasian dari metode *Dice Similarity*, diambil contoh dari definisi dari komputer menurut para ahli yaitu Menurut Robert H. Blissmer, "*komputer adalah suatu alat elektronik yang mampu melakukan beberapa tugas seperti menerima input, memproses input tadi sesuai dengan programnya, menyimpan perintah-perintah dan hasil pengolahan, serta menyediakan output dalam bentuk informasi"*. Sedangkan menurut Arief Susanto, "*komputer adalah sekelompok alat elektronik yang terdiri atas perintah input, alat yang mengolah input, dan peralatan output yang memberikan informasi serta bekerja secara otomatis"* (Susanto, 2009).

Setelah mendapatkan kedua data di atas, kemudian dilakukan pembagian. Untuk pengertian dari Robert H. Blissmer akan menjadi data dari *database* sebagai X, kemudian untuk pengertian dari Arief Susanto akan menjadi data yang akan dibandingkan dengan data dari *database* sebagai Y. Sebelum diolah, kedua data di atas akan dilakukan penghapusan *Stopword* dan karakter yang tidak berguna dalam perhitungan seperti pada Tabel 2.1.

Tabel 2.1 Perbandingan Data X dan Data Y

Data X	Data Y
alat elektronik tugas menerima input memproses input sesuai programnya menyimpan perintah - perintah hasil pengolahan menyediakan output bentuk informasi	sekelompok alat elektronik perintah input alat mengolah input peralatan output informasi otomatis

Setelah dilakukan penghapusan *Stopword* dan karakter yang tidak berguna, kemudian dilakukan pembuatan kamus kata lalu dilakukan perhitungan jumlah kata terhadap kamus data yang ada seperti pada Tabel 2.2.

Tabel 2.2 Kamus Data

No	Kamus Data	Data X	Data Y
1	alat	2	1
2	bentuk	0	1
3	elektronik	1	1
4	hasil	0	1
5	informasi	1	1
6	input	2	2
7	memproses	0	1
8	menerima	0	1
9	mengolah	1	0
10	menyediakan	0	1
11	menyimpan	0	1
12	otomatis	1	0
13	output	1	1
14	pengolahan	0	1
15	perintah	1	2
16	perlatan	1	0
17	programnya	0	1
18	sekelompok	1	0
19	sesuai	0	1
20	tugas	0	1

Setelah itu, data di atas akan diubah menjadi vektor satu dimensi sehingga menjadi :

$$X = \{2, 0, 1, 0, 1, 2, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0\} \quad (2.2)$$

$$Y = \{1, 1, 1, 1, 1, 2, 1, 1, 0, 1, 1, 0, 1, 1, 2, 0, 1, 0, 1, 1\} \quad (2.3)$$

Kemudian kedua vektor di atas dimasukkan ke dalam rumus *Dice Similarity* pada persamaan 2.3. Untuk mempermudah perhitungan, dilakukan beberapa penambahan *row* dan *field* sehingga menjadi :

Tabel 2.3 Perhitungan Dalam Kamus Data

No	Kamus Data	Data X	Data Y	X . Y	X^2	Y^2
1	alat	2	1	2	4	1
2	bentuk	0	1	0	0	1
3	elektronik	1	1	1	1	1
4	hasil	0	1	0	0	1
5	informasi	1	1	1	1	1
6	input	2	2	4	4	4

No	Kamus Data	Data X	Data Y	X . Y	X^2	Y^2
7	memproses	0	1	0	0	1
8	menerima	0	1	0	0	1
9	mengolah	1	0	0	1	0
10	menyediakan	0	1	0	0	1
11	menyimpan	0	1	0	0	1
12	otomatis	1	0	0	1	0
13	output	1	1	1	1	1
14	pengolahan	0	1	0	0	1
15	perintah	1	2	2	1	4
16	perlatan	1	0	0	1	0
17	programnya	0	1	0	0	1
18	sekelompok	1	0	0	1	0
19	sesuai	0	1	0	0	1
20	tugas	0	1	0	0	1
Total		**12**	**18**	**11**	**16**	**22**

Setelah memperoleh hasil seperti di Tabel 2.3, kemudian nilai tersebut dimasukkan ke dalam rumus *Dice Similarity* sehingga menjadi :

$$DSC = \frac{2|X.Y|}{|X|^2 + |Y|^2} \quad (2.4)$$

$$DSC = \frac{2|11|}{16 + 22} \quad (2.5)$$

$$DSC = \frac{22}{38} \quad (2.6)$$

$$DSC = 0.5789473684 \quad (2.7)$$

Sehingga diperoleh nilai *Dice Similarity* adalah 0.5789473684 atau jika dipersentasekan menjadi 57.8 %. Jadi, persentase kesamaan antara Data X dengan Data Y adalah sebesar 57.8 %

BAB III
METODE PENELITIAN

3.1 Prosedur Penelitian

Prosedur Penelitian merupakan serangkaian kegiatan yang dilaksanakan secara sistematis serta teratur agar mencapai tujuan dari penelitian ini. Adapun prosedur penelitian sebagai berikut :

1. Analisa Kebutuhan Sistem

   Dalam tahap ini, dilakukan analisa terhadap kebutuhan sebuah sistem seperti bagaimana tampilan sistem, bagaimana cara kerja sistem, ataupun penggunaan bahasa pemrograman yang tepat.
2. Studi Literatur

   Dalam tahap ini, dilakukan pengumpulan teori-teori ataupun landasan dasar dalam proses membangun sebuah sistem yang akan dibuat.
3. Perancangan Sistem

   Dalam tahap ini, dilakukan pembangunan Sistem Pendeteksi Kesamaan Teks Menggunakan Metode *Dice Similarity* dengan menggunakan metode, teori, ataupun bahasa pemrograman yang telah ditentukan sebelumnya.
4. Pengujian Sistem

   Dalam tahap ini, dilakukan *testing* atau pengujian terhadap sistem.

3.2 *Timeline* Penelitian

Timeline Penelitian adalah gambaran visual yang menggambarkan peristiwa atau tahapan yang akan terjadi dalam penelitian ini. Untuk Timeline penelitian dapat dilihat pada Tabel 3.1.

Tabel 3.1 *Timeline* Penelitian

No	Prosedur Penelitian	Waktu															
		Oktober				November				Desember				Januari			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Analisa Kebutuhan Sistem																
2	Studi Literatur																
3	Perancangan Sistem																
4	Pengujian Sistem																

Pada *timeline* penelitian ini, terdapat beberapa prosedur yang akan dilakukan dengan rentang waktu yang ada seperti analisa kebutuhan sistem, studi literatur, perancangan sistem, dan pengujian sistem dengan rentang waktu yang ada. Dengan adanya *timeline* penelitian ini, dapat mengendalikan pelaksanaan penelitian secara menyeluruh agar pelaksanaan penelitian tersebut berjalan dengan lancar.

3.3 Rancangan Sistem

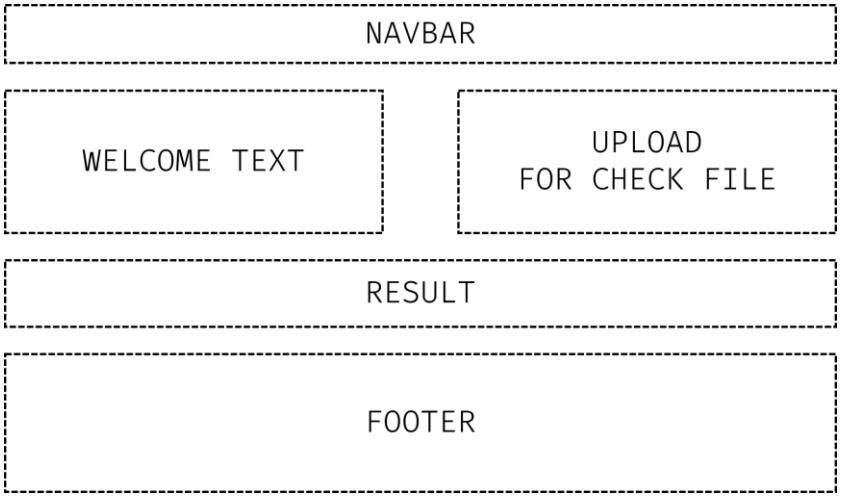
3.3.1 Rancangan Antar Muka (*Interface*)

Rancangan antar muka adalah suatu proses dalam menyusun dan menganalisis sistem secara fisik maupun non fisik untuk dapat dimanfaatkan ke depannya. Perancangan antarmuka terdiri dari beberapa proses mulai dari membuat fungsi yang ada pada sistem, informasi masalah, tahapan membuat *prototype*, melakukan implementasi terhadap perancangan yang dibuat dan evaluasi hasil (Sridevi, 2014).

Untuk aplikasi terdapat beberapa rancangan *interface* seperti berikut :

a. Halaman Utama

   Halaman utama merupakan halaman awal yang akan diakses pertama kali saat membuka *website* ini. Untuk rancangan antar muka pada halaman utama dapat dilihat pada Gambar 3.1.

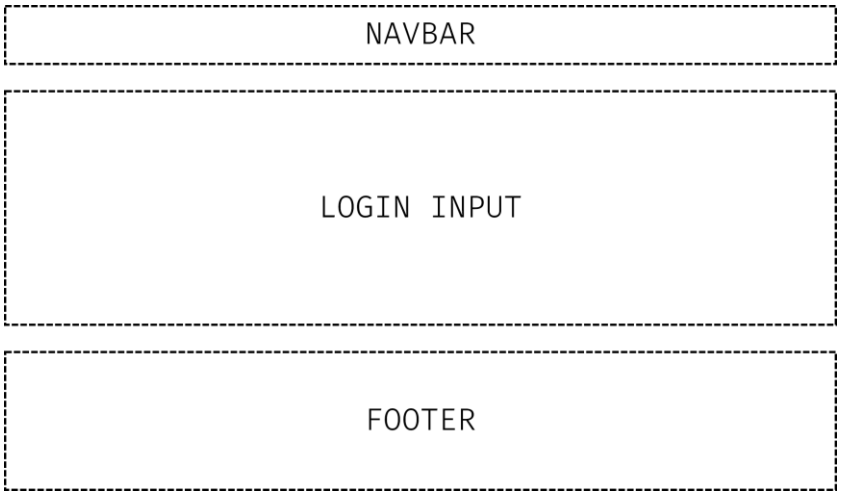


Gambar 3.1 Rancangan Antar Muka Halaman Utama

b. Halaman *Login*

   Halaman *login* merupakan halaman untuk masuk ke dalam *website*. Pada halaman ini terdapat input berupa input *username* dan *password*, serta

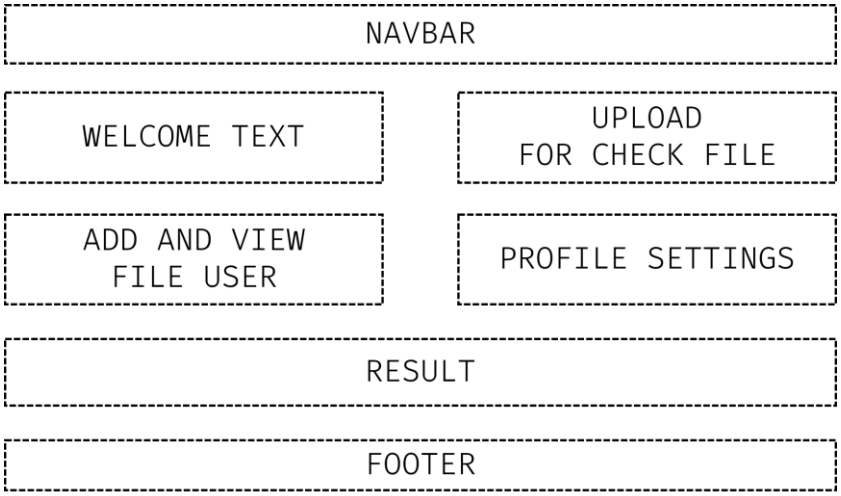
tombol *login*. Untuk rancangan antar muka pada halaman login dapat dilihat pada Gambar 3.2.



Gambar 3.2 Rancangan Antar Muka Halaman *Login*

c. Halaman Beranda

Halaman beranda merupakan halaman yang akan didapatkan setelah berhasil *login* ke dalam *website* sebagai user. Pada halaman ini, terdapat beberapa fitur seperti tambah *file* dan ubah pengaturan *profile*. Untuk rancangan antar muka pada halaman beranda dapat dilihat pada Gambar 3.3.

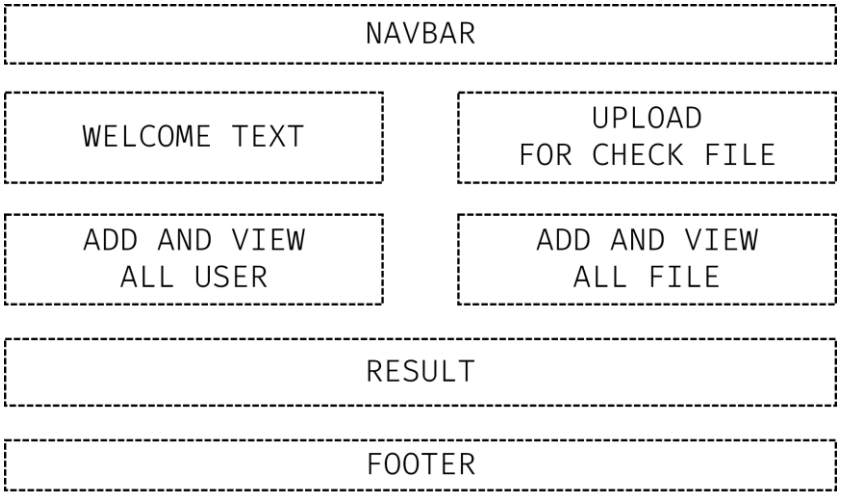


Gambar 3.3 Rancangan Antar Muka Halaman Beranda

d. Halaman *Admin*

Halaman *admin* merupakan halaman yang didapatkan setelah berhasil *login* sebagai *admin*. Untuk halaman ini, terdapat lebih banyak tambahan fitur

dibandingkan dengan halaman beranda seperti dapat menambah atau menghapus *user* dan *files*. Untuk rancangan antar muka pada halaman admin dapat dilihat pada Gambar 3.4.



Gambar 3.4 Rancangan Antar Muka Halaman *Admin*

3.4 Metode Penelitian

Dalam sistem ini, metode perhitungan yang digunakan dalam metode *Dice Similarity* dengan membandingkan data teks yang diinput pengguna dengan teks yang terdapat di dalam *database* atau repositori teks.

3.4.1 Metode Prototype

Metode *Prototype* adalah sebuah metode pengembangan *software* yang cukup banyak digunakan karena Teknik ini mengembangkan sistem yang menggunakan *prototype* untuk menggambarkan sistem sehingga klien atau pemilik sistem mempunyai gambaran jelas pada sistem yang akan dibangun oleh tim pengembang. Dengan metode ini, pengembang dan pengguna bisa saling berinteraksi selama proses pengembangan *software*. Hal ini tentu sangat menguntungkan dan semakin memudahkan dalam pembuatan perangkat lunak.

Saat menggunakan model jenis ini, kesalahan biasanya dapat dideteksi lebih cepat dan umpan balik pengguna yang lebih cepat tersedia untuk menghasilkan solusi yang lebih baik. Dalam metodologi ini model kerja dari sistem disediakan, pengguna mendapatkan pemahaman yang lebih baik tentang sistem yang sedang dikembangkan. Developer bisa bekerja menentukan kebutuhan klien dengan baik,

Efisiensi waktu tinggi dalam pengembangan sistem serta Lebih mudah dalam penerapannya karena klien mengetahui apa yang dibutuhkan.

Dalam penelitian ini dilakukan sesuai tahapan yang ada pada model *prototype* untuk mempermudah memahami proses dari awal sampai sistem selesai dan implementasi.

3.5 Instrumen Penelitian

Instrumen yang digunakan dalam penelitian ini adalah terbagi dua yaitu *hardware* dan *software*. Untuk *hardware,* menggunakan laptop dengan *processor* AMD Athlon 300U with Radeon Vega Mobile Gfx 2.40 GHz dan RAM sebanyak 8 GB serta sistem operasi Windows 10 Home 64 bit. Sedangkan *software* yang digunakan yaitu *Visual Studio Code* dan *Microsoft Edge.*

BAB IV
HASIL DAN PEMBAHASAN

4.1. *Requirements Gatehering And Analysis* (Analisis Kebutuhan)

Dalam tahap ini kebutuhan sistem didefinisikan dengan rinci. Dalam prosesnya, peneliti akan menganalisa kebutuhan apa saja yang diperlukan demi pembuatan sistem yang sesuai dengan keinginan agar aplikasi dapat berjalan sebagaimana mestinya. Kebutuhan yang akan dianalisis mulai dari sistem yang akan dibangun, bahasa yang digunakan, *database* yang sesuai dengan aplikasi serta infrastruktur yang digunakan dalam membangun aplikasi tersebut. Analisis kebutuhan fungsional sistem terdiri dari beberapa fungsi utama yang saling berkaitan dan saling mendukung satu sama lain. Sedangkan kebutuhan non fungsional sistem memiliki fungsi sebagai sarana pendukung agar kelancaran dari fungsi utama beroperasi sesuai dengan harapan.

Analisis kebutuhan fungsional pada sistem yaitu terdapat beberapa jenis pengguna yaitu *guest*, *user*, dan admin di mana semua jenis pengguna ini memiliki interaksi yang berbeda-beda sehingga memiliki ruang lingkup yang berbeda juga. Aktivitas yang akan terjadi pada pengguna berbeda-beda namun berhubungan. Untuk *guest* atau pengguna biasa hanya dapat melakukan pengecekan dokumen. Pada *user*, selain bisa melakukan apa yang bisa dilakukan oleh *guest*, pengguna juga bisa melakukan manajemen dokumen yang akan dijadikan sebagai pembanding dalam sistem seperti menambah, mengubah, atau menghapus dokumen tersebut. Sedangkan untuk Admin, selain pengguna dapat melakukan apa yang dapat dilakukan oleh *guest* dan *user*, pengguna juga dapat melakukan manajemen *user* seperti menambah, mengubah, atau menghapus *user*.

Analisis kebutuhan non fungsional yaitu meliputi kebutuhan akan spesifikasi seperti *hardware* dan *software* yang mendukung jalannya sistem. Sistem ini membutuhkan pemrosesan data yang cepat karena akan mengolah banyak data berupa dokumen. Selain itu, *database* yang digunakan harus bisa menangani data yang besar. Maka berdasarkan hal tersebut, maka kebutuhan non fungsional yang akan dibutuhkan yaitu :

A. Hardware

Spesifikasi perangkat keras (*hardware*) yang digunakan agar sistem dapat berjalan yaitu :

Processor	:	AMD Athlon 300U *with* Radeon Vega Mobile Gfx 2.40 GHz
*Memory (*RAM)	:	8 Gb
Hardisk	:	*Solid State Drive* (SSD) 2400 Gb

B. Software

Spesifikasi perangkat lunak (*software*) yang digunakan agar sistem dapat berjalan yaitu :

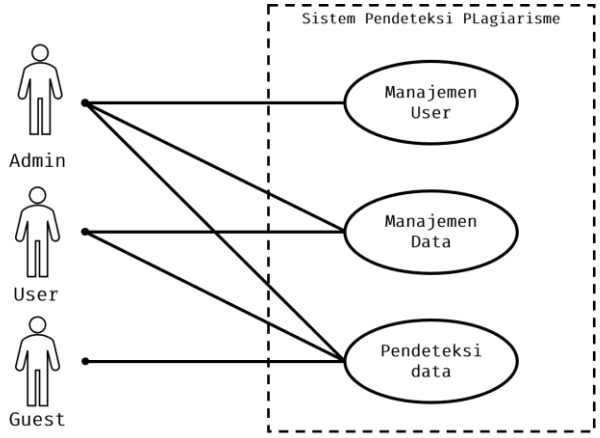
Sistem Operasi	:	Windows 10 *Home Single Language*
Server	:	*NodeJs*
Bahasa Pemrograman	:	*Javascript*
Framework	:	ExpressJs, Tailwindcss 3
Database	:	MongoDb
Text Editor	:	*Visual Studio Code*
Web Browser	:	Microsoft Edge

4.2 *Quick Design* (Desain Cepat)

Tahap selanjutnya adalah pembuatan desain sederhana yang akan memberi gambaran singkat tentang sistem yang ingin dibuat. Desain yang dibuat pada tahap ini berdasarkan pada tahap sebelumnya.

4.2.1 *Use Case Diagram*

Use Case Diagram adalah suatu jenis diagram yang menggambarkan hubungan atau interaksi dari sebuah sistem dengan pengguna. Dengan adanya *Use Case Diagram*, kita dapat mendeskripsikan tipe atau jenis yang dilakukan pengguna terhadap sebuah sistem. Untuk *use case* diagram dari Sistem Pendeteksi Kesamaan Teks Menggunakan Metode *Dice Similarity* dapat dilihat pada Gambar 4.1.

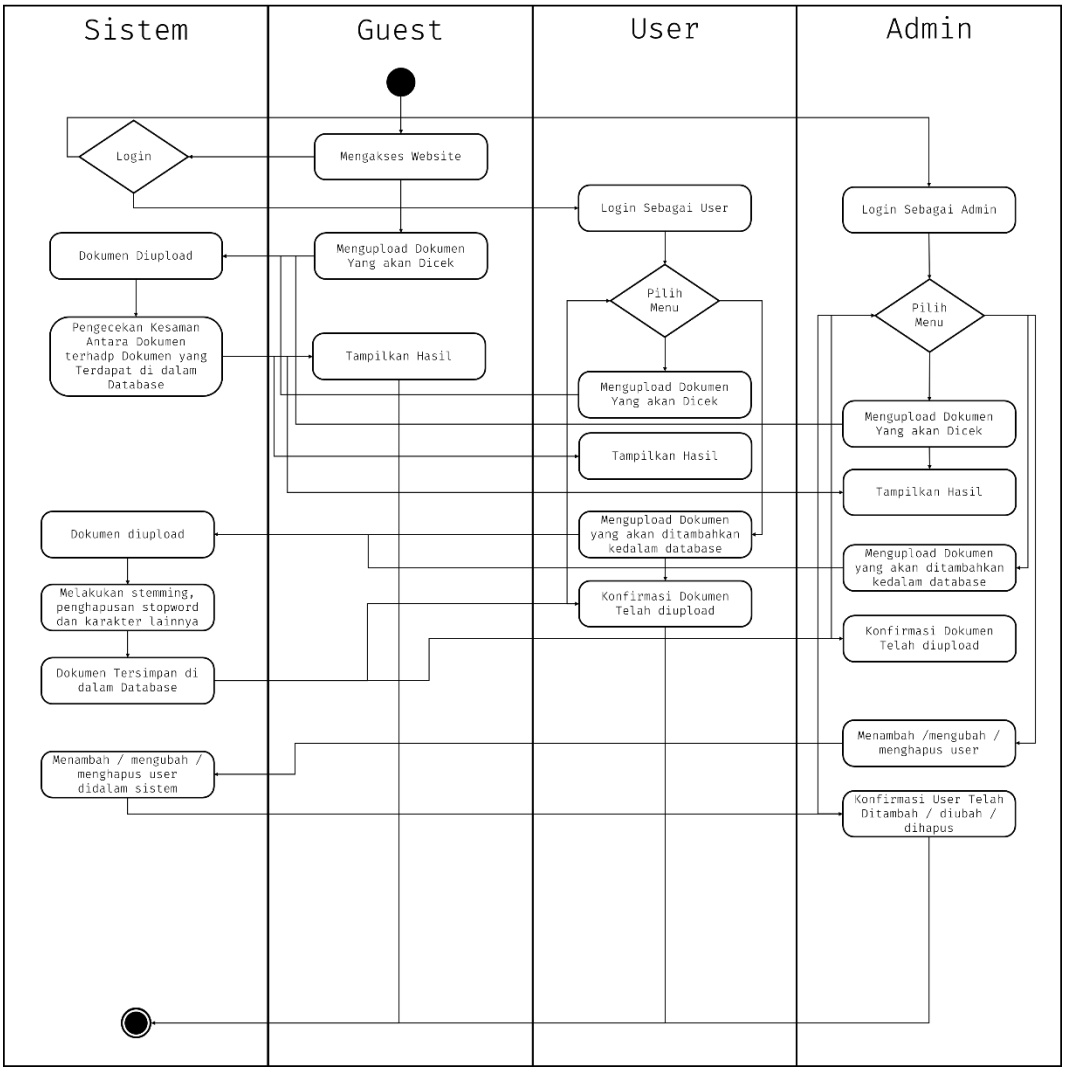


Gambar 4.1 *Use Case Diagram*

Dalam *use case diagram* pada Gambar 3.1, terdapat 3 jenis pengakses yang akan menggunakan sistem ini yaitu *Admin*, *User*, dan *Guest*. Admin memiliki akses terhadap semua sistem baik manajemen dari *user*, dan juga manajemen data yang disimpan dalam *database*. Untuk user, hanya memiliki akses terhadap data yang disimpan ke dalam *database* nantinya berdasarkan yang dimiliki oleh *user*. Selain itu, *user* juga dapat menggunakan sistem pendeteksi kesamaan dokumen yang ada dalam sistem ini. Sedangkan *guest* atau tamu hanya bisa melakukan pendeteksi kesamaan dokumen dan tidak bisa melakukan manajemen data yang ada di dalam *database*.

4.2.2 *Activity Diagram*

,*Acitvity diagram* adalah diagram yang dapat memodelkan proses-proses yang terjadi pada sebuah sistem. Runtutan proses dari suatu sistem digambarkan secara vertikal. *Activity diagram* merupakan pengembangan dari *Use Case* yang memiliki alur aktivitas. Alur atau aktivitas berupa bisa berupa runtutan menu-menu atau proses bisnis yang terdapat di dalam sistem tersebut. Untuk *activity diagram* pada Sistem Pendeteksi Kesamaan Teks Menggunakan Metode *Dice Similarity* dapat dilihat pada Gambar 4.2.



Gambar 4.2 *Activity Diagram*

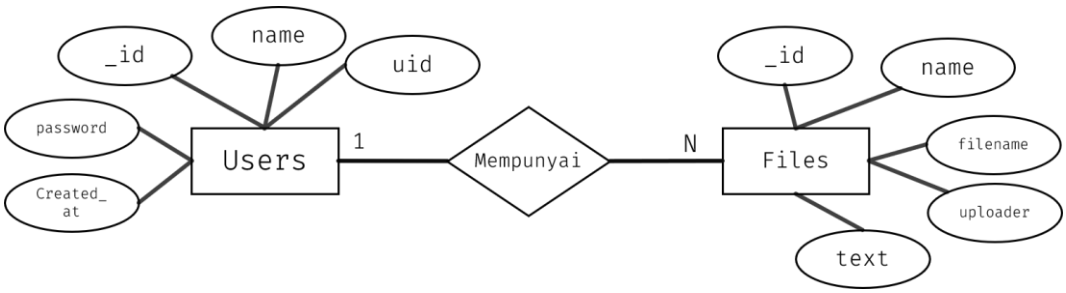
Pada Gambar 4.2 dapat dilihat ada beberapa aktivitas yang dapat dilakukan pada sistem ini berdasarkan *role* atau jenis pengguna pada sistem ini seperti *guest* yang hanya terbatas hanya pada pengecekan kesamaan dokumen saja. Untuk pengguna user, selain bisa melakukan pengecekan dokumen, juga bisa menambah dokumen yang akan dijadikan perbandingan pada saat pengecekan nantinya. Selain itu, terdapat admin yang bisa melakukan semua yang dapat dilakukan oleh user dan *guest*. Selain itu admin dapat melakukan manajemen user seperti menambah, mengubah, dan menghapus user yang terdapat dalam sistem.

4.2.3 Entity Relationship Diagram (ERD)

Entity Relationship Diagram (ERD) adalah suatu jenis diagram yang digunakan untuk merancang suatu basis data (*database*). Dalam ERD, terdapat beberapa

komponen seperti entitas, relasi, atribut, dan garis penghubung di mana digunakan untuk memperlihatkan hubungan atau relasi antar entitas atau objek yang terlihat beserta atributnya. Untuk ERD pada Sistem Pendeteksi Kesamaan Teks Menggunakan Metode *Dice Similarity* dapat dilihat pada Gambar 4.3.

.



Gambar 4.3 *Entity Relation Diagram*

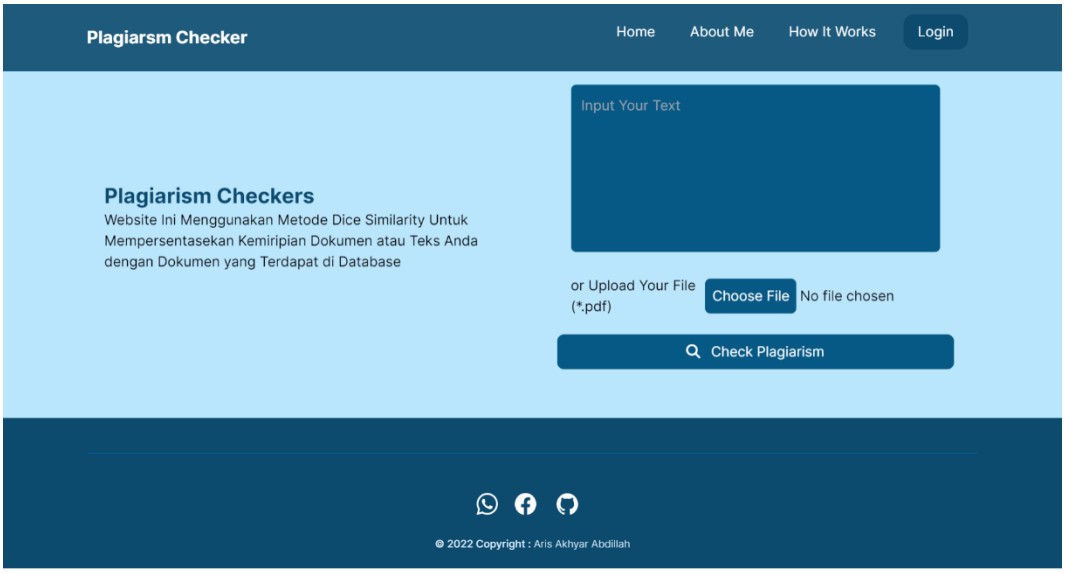
Berdasarkan ERD pada Gambar 3.2, terdapat dua entitas yaitu Users dan Files, di mana Users memiliki beberapa atribut yaitu _id, *name*, *password*, *uid*, *password*, dan *created_at*. Sedangkan Files memiliki beberapa atribut juga seperti *_id*, *name*, *filename*, *uploader*, dan *text*. *Collections* Users mempunya relasi *One to Many* terhadap Tabel Files, sedangakan sebaliknya yaitu *Collections Files* memiliki relasi *One to One* terhadap *Collections* Users.

4.2.4 Tampilan Antar Muka (*Interface*)

Tampilan antar muka merupakan tampilan yang akan muncul saat pengguna mengakses *website* yang akan dibuat. Tampilan antarmuka ini dibuat berdasarkan rancangan antar muka yang telah rancang sebelumnya. Untuk tampilan antar muka dapat dilihat sebagai berikut :

a. Halaman Utama

   Halaman utama merupakan halaman yang diakses pertama kali oleh user. Berdasarkan rancangan antar muka sebelumnya, maka tampilan antar muka halaman utama dapat dilihat pada Gambar 4.4.

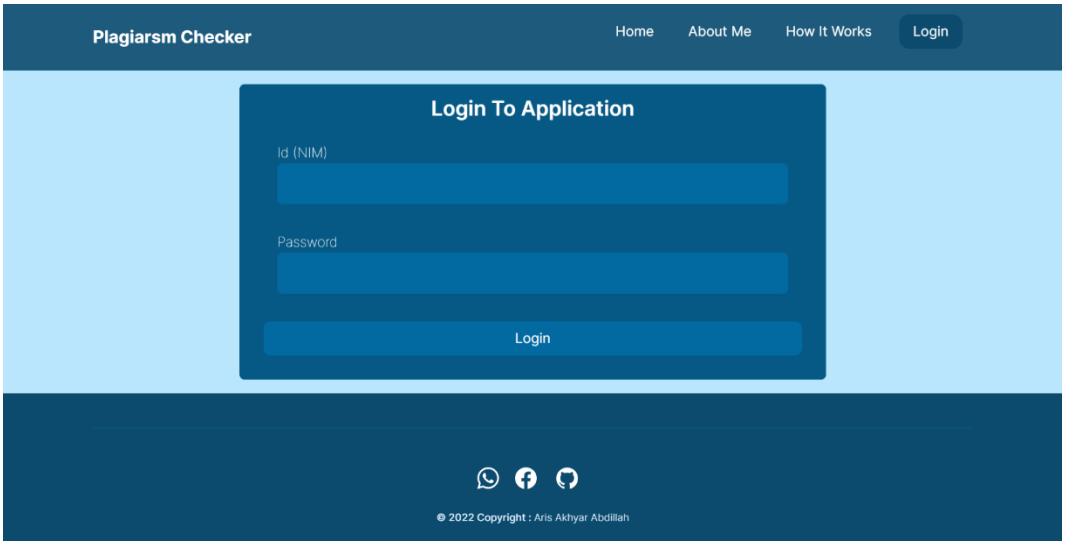


Gambar 4.4 Tampilan Halaman Utama

Dapat dilihat pada Gambar 4.4, tampilan *website* ini berdasarkan rancangan antar muka terdapat beberapa bagian seperti *navbar*, *welcome text*, *upload for check file*, serta *footer*.

b. Halaman Login

Halaman login merupakan halaman yang diakses ketika pengguna ingin mengakses *website* baik sebagai user, ataupun sebagai admin. Berdasarkan rancangan antar muka sebelumnya, maka tampilan antar muka halaman utama dapat dilihat pada Gambar 4.5.

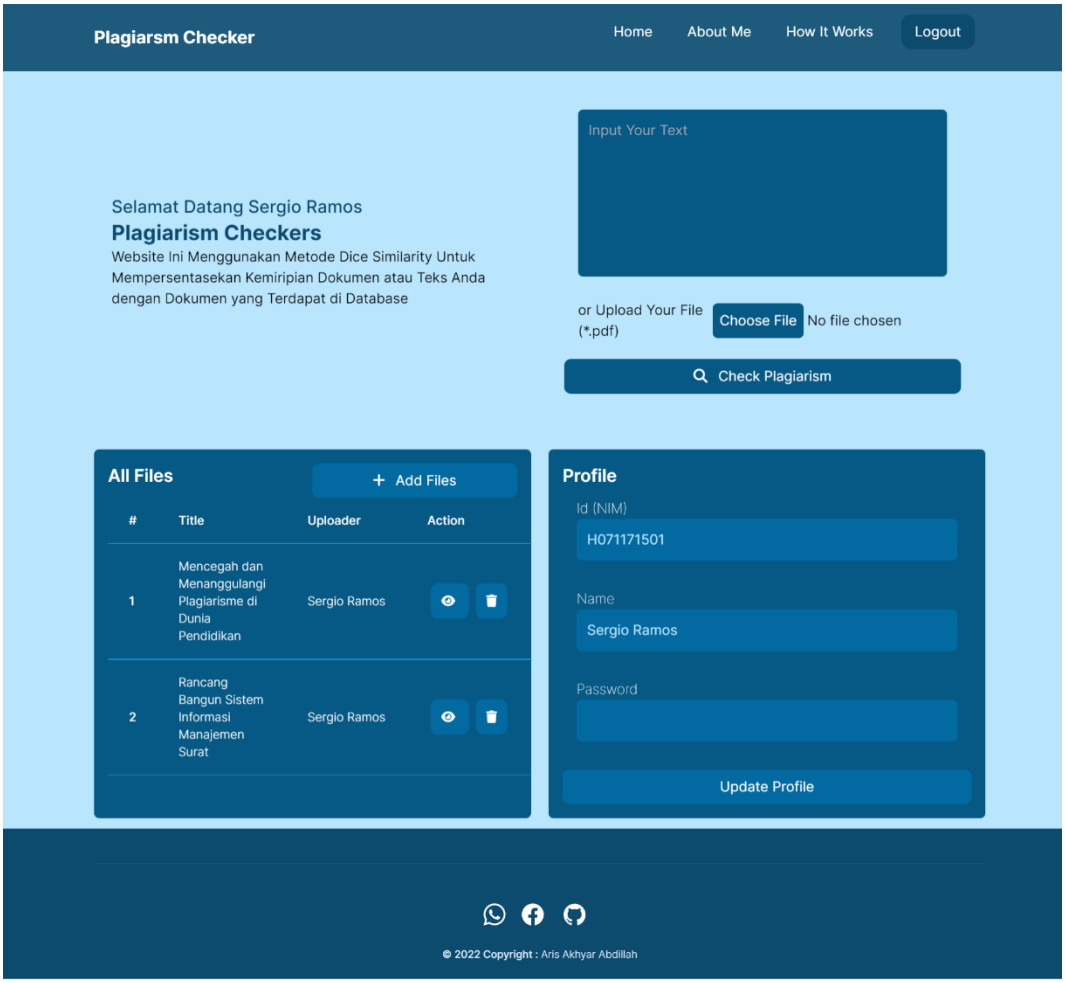


Gambar 4.5 Tampilan Halaman *Login*

Pada Gambar 4.5, tampilan pada halaman login ini berdasarkan rancangan antar muka yang memiliki beberapa bagian seperti *navbar*, *login input*, dan *footer*.

c. Halaman Beranda

Halaman beranda merupakan halaman yang didapat pengguna ketika berhasil *login* ke dalam sistem sebagai user. Berdasarkan rancangan antar muka sebelumnya, maka tampilan antar muka halaman utama dapat dilihat pada Gambar 4.6.

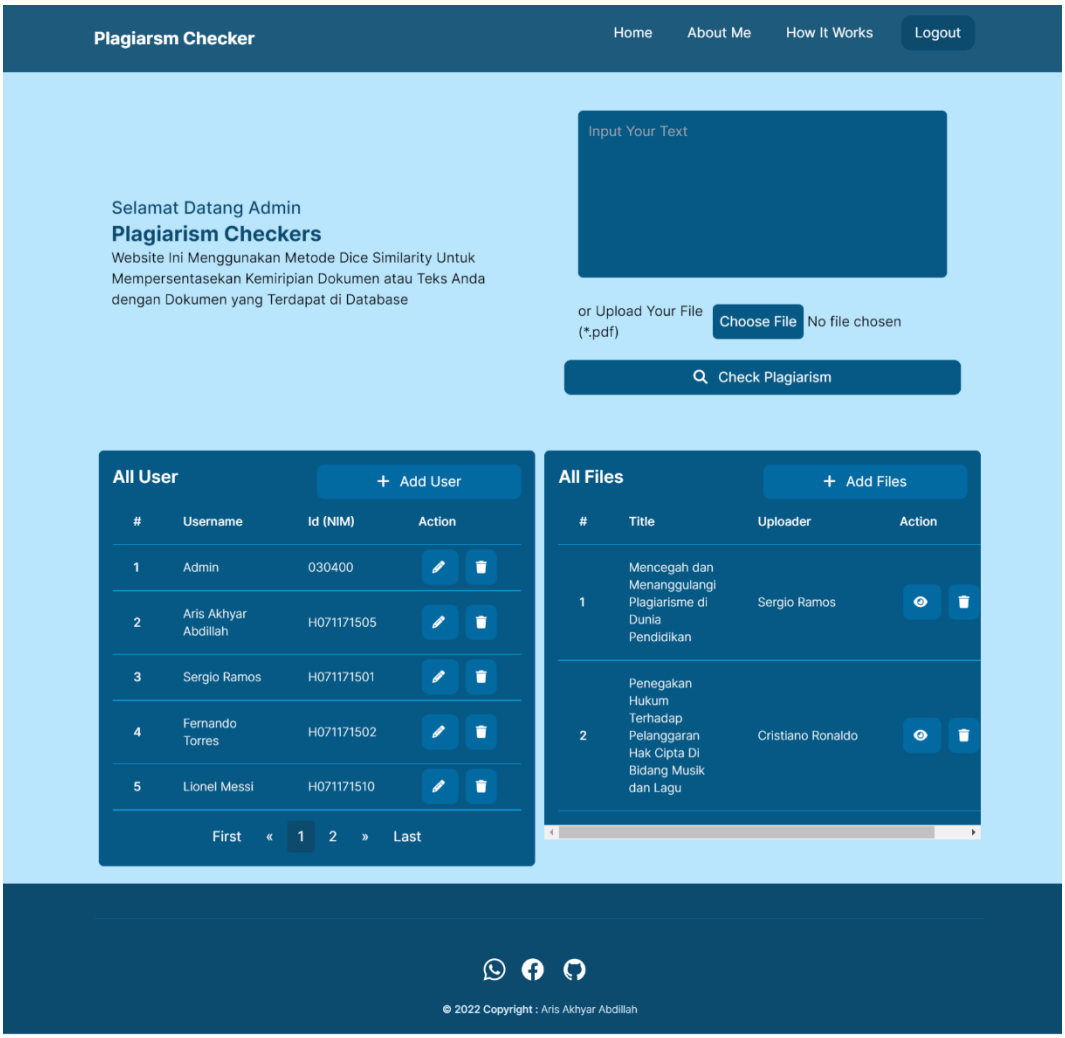


Gambar 4.6 Tampilan Halaman Beranda

Pada Gambar 4.6, tampilan pada halaman beranda ini berdasarkan rancangan antar muka yang memiliki beberapa bagian seperti *navbar*, *welcome text*, *upload for check file* , *add and view file user*, *profile settings*, dan *footer*.

d. Halaman Admin

Halaman admin merupakan halaman yang didapat pengguna ketika login sebagai admin ke dalam sistem. Berdasarkan rancangan antar muka sebelumnya, maka tampilan antar muka halaman utama dapat dilihat pada Gambar 4.7.



Gambar 4.7 Tampilan Halaman Admin

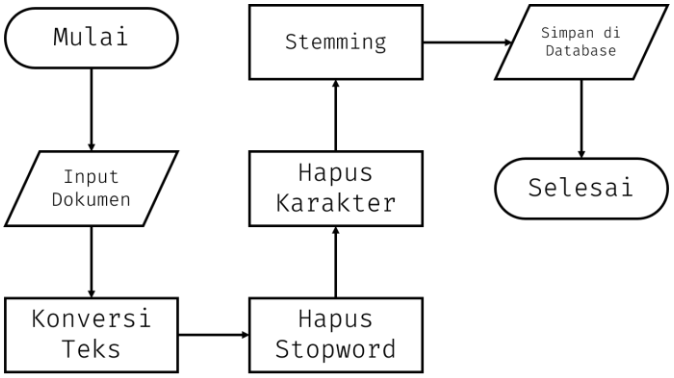
Pada Gambar 4.7, tampilan pada halaman beranda ini berdasarkan rancangan antar muka yang memiliki beberapa bagian seperti *navbar*, *welcome text*, *upload for check file*, *add and view file all user*, *add and view file all files*, *profile settings*, dan *footer*.

4.2.5 Desain Sistem

Pada bagian ini, akan menjelaskan bagaimana desain suatu sistem yang diimplementasikan. Dengan adanya desain sistem ini, dapat memberikan gambaran rancang bangun yang lengkap terhadap pengguna dari sistem ini.

4.2.5.1 Algoritma Input Dokumen ke dalam *Database*

Algoritma input dokumen ke dalam *database* merupakan salah satu tahapan dalam sistem ini di mana dokumen akan dimasukkan ke dalam *database*. *Flowchart* untuk tahapan ini dapat dilihat pada Gambar 4.8.



Gambar 4.8 Algoritma Input Dokumen ke dalam *Database*

Keterangan :

1. Input Dokumen.

   Sebelum membandingkan teks yang akan ditentukan kesamaannya, terlebih dahulu dibuat sebuah repositori atau database teks yang akan digunakan sebagai pembanding nantinya. Teks yang diinput pada penelitian ini berupa *file* yang berformat pdf (*Portable Document Format*).
2. Konversi Teks.

   Teks yang telah diinput, kemudian diubah ke dalam bentuk tipe data *String* agar dapat diolah nantinya di dalam sistem.
3. Hapus *Stopword.*

   Stopword merupakan kata yang diabaikan karena memiliki frekuensi kemunculan yang sangat tinggi dan tidak mempunyai arti. Beberapa contoh *Stopword* seperti, "atau", "dan", dan "tapi". Dengan menghapus *Stopword*,

sistem dapat bekerja lebih cepat karena *Stopword* akan menghapus kata-kata yang tidak relevan atau yang abaikan (Rahutomo & Ririd, 2018).

4. Hapus Karakter.

   Selain *Stopword*, karakter seperti ",", ".", "?", dll yang hanya berupa karakter yang tidak memiliki arti dan fungsi karena sistem hanya memeriksa kata.

5. *Stemming*

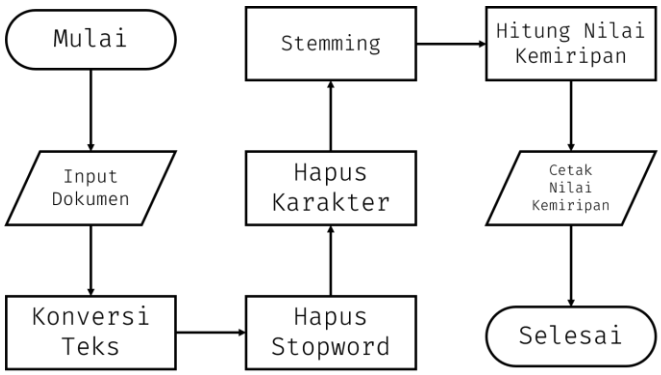
   Stemming adalah proses yang dilakukan untuk mengembalikan kata berimbuhan ke bentuk dasarnya dengan cara menghilangkan imbuhan. Imbuhan tersebut berupa awalan, akhiran, awalan dan akhiran, dan sisipan (Winarti, Titin, Jati, & Sunny, 2017). Dengan adanya *stemming*, sistem dapat menghilangkan kata imbuhan menjadi kata dasar sehingga dapat mempermudah perhitungan.

6. Simpan di *Database.*

   Setelah teks telah melewati tahap sebelumnya, maka teks akan dimasukkan ke dalam *database* yang nantinya akan digunakan saat ingin mendapatkan persentase kemiripan dari teks yang akan dibandingkan nantinya.

4.2.5.2 Algoritma Perbandingan Dokumen

Algoritma perbandingan dokumen merupakan salah satu tahapan dalam sistem ini di mana dokumen akan dibandingkan dengan dokumen lainnya yang sudah berada di dalam *database*. *Flowchart* untuk tahapan ini dapat dilihat pada Gambar 4.9.



Gambar 4.9 Algoritma Perbandingan Dokumen

Keterangan :

1. Input Dokumen.

   Input teks yang akan dibandingkan ke dalam sistem yang nantinya akan dibandingkan terhadap teks yang terdapat dalam *database* atau repositori. Teks yang di input berupa teks yang berformat pdf (*Portable Document Format*).

2. Konversi Teks.

   Teks yang telah diinput, kemudian diubah ke dalam bentuk tipe data *String* agar dapat diolah nantinya di dalam sistem.

3. Hapus *Stopword.*

   Stopword merupakan kata yang diabaikan karena memiliki frekuensi kemunculan yang sangat tinggi dan tidak mempunyai arti. Beberapa contoh *Stopword* seperti, "atau", "dan", dan "tapi". Dengan menghapus *Stopword*, sistem dapat bekerja lebih cepat karena *Stopword* akan menghapus kata-kata yang tidak relevan atau yang abaikan (Rahutomo & Ririd, 2018). Dengan adanya *stemming*, sistem dapat menghilangkan kata imbuhan menjadi kata dasar sehingga dapat mempermudah perhitungan.

4. Hapus Karakter.

   Selain *Stopword*, karakter seperti ",", ".", "?", dll yang hanya berupa karakter yang tidak memiliki arti dan fungsi karena sistem hanya memeriksa kata.

5. *Stemming*

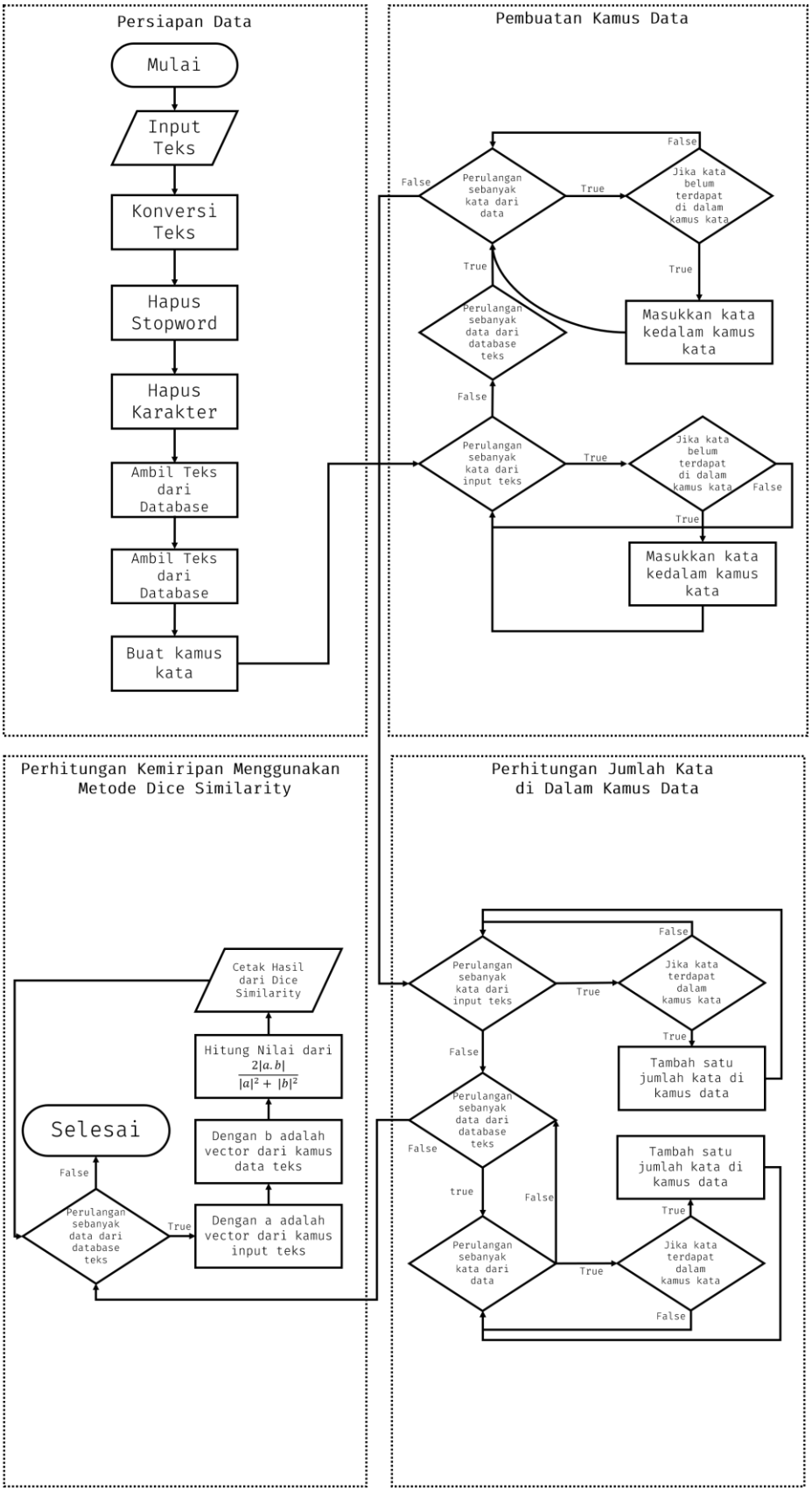
   Stemming adalah proses yang dilakukan untuk mengembalikan kata berimbuhan ke bentuk dasarnya dengan cara menghilangkan imbuhan. Imbuhan tersebut berupa awalan, akhiran, awalan dan akhiran, dan sisipan (Winarti, Titin, Jati, & Sunny, 2017).

6. Hitung Nilai Kemiripan.

   Setelah teks melewati tahap sebelumnya, maka teks akan dibandingkan dengan semua teks yang terdapat di dalam *database* atau repositori. Teks akan dibandingkan dengan semua data teks dengan menggunakan metode *Dice Similarity*. Kemudian hasil kemiripan antar teks akan muncul sebanyak data yang terdapat dalam *database* atau repositori teks.

4.2.5.3 Algoritma *Dice Similarity*

Algoritma perhitungan persentase kemiripan antar teks dapat dilihat pada Gambar 4.10.



Gambar 4.10 Algoritma Dice Similarity

1. Persiapan Data.

   Sebelum dilakukan perhitungan, data disiapkan terlebih dahulu, di mana data input teks merupakan data yang akan dibandingkan dengan data teks yang terdapat di dalam *database* atau repositori.

2. Pembuatan Kamus Data.

   Untuk mempermudah perhitungan *Dice Similarity*, dilakukan pembuatan kamus data yang berisi kata yang terdapat di dalam input teks maupun data teks yang berasal dari *database* atau repositori. Pada awal tahap ini, dilakukan perulangan sebanyak kata yang terdapat di dalam input teks kemudian dilakukan pengkondisian jika di dalam kamus kata tidak terdapat indeks yang sama dengan kata, maka kata tersebut ditambahkan ke dalam kamus. Setelah itu, dilakukan perulangan terhadap data dalam *database* atau repositori kemudian dilakukan perulangan kembali sebanyak kata yang terdapat di dalam teks dan dilakukan hal yang sama dengan input teks di awal.

3. Perhitungan Jumlah Kata di Dalam Kamus Data.

   Sebelum dilakukan perhitungan, terlebih dahulu ditentukan jumlah kata yang terdapat di dalam teks untuk data teks input dan data teks dari *database*. Pada awal tahap ini, dilakukan perulangan sebanyak kata yang terdapat di dalam input teks kemudian dilakukan pengkondisian jika di dalam kamus kata terdapat kata yang sama dengan kata yang terdapat di kamus kata, maka kata untuk indeks input teks akan bertambah satu. dilakukan perulangan terhadap data dalam *database* atau repositori kemudian dilakukan perulangan kembali sebanyak kata yang terdapat di dalam teks dan dilakukan hal yang sama dengan input teks di awal.

4. Perhitungan Kemiripan Menggunakan Metode *Dice Similarity*.

   Setelah memperoleh kamus data, maka akan dilakukan perhitungan untuk memberikan persentase kemiripan antara input teks dan data teks yang berasal dari *database*. Ditahap ini dilakukan perulangan sebanyak data dalam *database*. Kemudian dilakukan perhitungan sesuai dengan rumus dari *Dice Similarity* kemudian hasilnya akan ditampilkan.

4.3 *Build Prototype* (Bangun Prototipe)

Setelah melalui tahapan sebelumnya, maka selanjutnya adalah pembangunan prototipe untuk pembuatan program atau aplikasi berdasarkan *quick desain* sebelumnya. Untuk pembuatan *website* ini menggunakan metode *Model View Controller* (MVC).

4.3.1 Model View Controller (MVC)

Model View Controller (MVC) adalah sebuah konsep yang diperkenalkan oleh penemu *Smalltalk* (Trygve Reenskaug) untuk membuat satu jenis paket data jaringan menjadi jenis data lainya bersama dengan pemrosesan (*model*), dari proses manipulasi (*controller*) dan tampilan (*view*) untuk dipresentasikan pada sebuah *user interface* (Deacon, 2009). Untuk pembuatan aplikasi menggunakan metode MVC membagi menjadi beberapa bagian yaitu *model*, *view*, dan *controller*.

Model merupakan *code* atau sintaks yang menghubungkan antara aplikasi ke dalam *database*. *Model* biasanya berisi deklarasi variabel yang mempunyai hubungan dengan *database* seperti deklarasi tabel, koneksi, dan semacamnya. *View* merupakan *code* atau sintaks yang menampilkan bagian depan dari aplikasi. Dalam *view* terdapat stintaks yang merupakan hasil dari rancangan antar muka yang telah dibuat sebelumnya yang kemudian diimplementasikan dalam bentuk sintaks atau *code*. Sedangkan *controller* merupakan *code* atau sintaks yang berisi fungsi yang akan mengatur sistem serta sebagai penghubung antara *model* dan *view* nantinya. Dalam *controller* biasanya terdapat logika yang mengatur sistem ataupun logika yang akan menampilkan *view* dari aplikasi.

4.3.2 Implementasi Pembuatan Sistem

Implementasi pembuatan sistem ini akan menggunakan *arsitektur Model View Controller* (MVC) dengan beberapa tambahan penyesuaian untuk membuat sistem. Pembagian sistem akan dibagi menjadi beberapa bagian seperti pada konsep *Model View Controller* (MVC) dengan tambahan beberapa *folder* tambahan seperti *config*, *middleware*, *routes*, *utils*, dan sebagainya. Untuk pembagiannya dibagi sebagai berikut :

1. *Model*

   Model merupakan *code* atau sintaks yang menghubungkan antara aplikasi ke dalam *database*. Contoh *code* atau sintaks salah satu model dalam sistem sebagai dapat dilihat pada Gambar 4.11.

```
const mongoose = require('mongoose')
const mongoosePaginate = require('mongoose-paginate-v2');
const bcrypt = require('bcryptjs')

require('../utils/db')

const userSchema = new mongoose.Schema( {
    name : {
        type : String,
        required : true,
    },
    uid : {
        type : String,
        required : true,
    },
    password : {
        type : String,
        required : true,
    },
})
userSchema.plugin(mongoosePaginate)
const User = mongoose.model('User', userSchema )
const salt = bcrypt.genSaltSync(10)

const verifyPassword = (password, h) => {
    return bcrypt.compare(password, h)
}

const createPassword = (password) => {
    return bcrypt.hashSync(password, salt)
}


module.exports = {
    User,
    userSchema,
    verifyPassword,
    createPassword
}
```

Gambar 4.11 Contoh Sintaks *Model*

Pada Gambar 4.5 merupakan *code* atau sintaks dari salah satu contoh model yaitu model User. Pada baris awal merupakan inisialisasi *library* yang digunakan yaitu *mongoose*. *Mongoose* merupakan *library* yang digunakan untuk membuat sebuah model untuk *database* MongoDb. Kemudian pada baris selanjutnya merupakan pembuatan dari model User di mana terdapat

beberapa *field* yaitu *name, uid,* dan *password*. Setelah mendefinisikan *field* pada model nantinya, kemudian di inisialisasi pada variabel User yang akan dipakai pada *controller*. Kemudian selanjutnya merupakan modul tambahan untuk mengenkripsi *password* yang akan disimpan pada *database*. Untuk baris akhir adalah kode untuk *mengexport code* agar dapat digunakan pada *code* atau *file* lain.

2. *View*

   View merupakan *code* atau sintaks yang menampilkan bagian depan dari aplikasi. Contoh *code* atau sintaks salah satu model dalam sistem dapat dilihat pada Gambar 4.12.

```
  <tr class="border-b border-sky-500">
      <td class="px-6 py-4 whitespace-nowrap text-sm font-
medium"><%= i + 1 %> </td>
      <td class="text-sm font-light px-6 py-4">
          <%= u.name %>
      </td>
      <td class="text-sm font-light px-6 py-4 whitespace-nowrap">
          <%= u.uid %>
      </td>a
      <td class="text-sm font-light px-6 py-4 whitespace-nowrap">
          <a href="" data-id="<%= u._id %> " data-uid="<%= u.uid
%>" data-name="<%= u.name %>"
              class="w-full mx-1 bg-sky-700 p-3 rounded-lg text-
white hover:bg-yellow-500 user-change"
              data-bs-toggle="modal" data-bs-
target="#addUserModal">
              <i class="fa-solid fa-pencil"></i>
          </a>
          <a href="" class="w-full mx-1 bg-sky-700 p-3 rounded-lg
text-white hover:bg-red-500 delete-user"
              data-id="<%= u._id %>" data-name="<%= u.name %> "
data-bs-toggle="modal" data-bs-target="#deleteUserModal">
              <i class="fa-solid fa-trash"></i>
          </a>
      </td>
  </tr>
```

Gambar 4.12 Contoh Sintaks *View*

Pada Gambar 4.12, merupakan *code* atau sintaks dari salah satu contoh *view* yaitu *view* dari *table* User pada Halaman Admin. Dalam salah satu contoh

view diatas merupakan sintaks dari *file* yang memiliki format ejs, yaitu salah satu *templating engine* di mana dalam filenya terdapat *markup* HTML dan Javascript. Pada baris awal merupakan inisialisasi pada table kemudian dibuat *tag row* <tr>. Selanjutnya untuk kolom menggunakan *tag coloumn* <td>. Untuk class yang digunakan berasal dari *utility class* TailwindCss. Karena memiliki format .ejs, dalam HTML ini dapat memasukkan kode dari Javascript menggunakan tag <%= %> didalamnya.

3. *Controller*

   Controller merupakan *code* atau sintaks yang berisi fungsi yang akan mengatur sistem serta sebagai penghubung antara *model* dan *view*. Contoh *code* atau sintaks salah satu *controller* dalam sistem dapat dilihat pada Gambar 4.13.

```
require('dotenv').config()
const {User, verifyPassword, createPassword} = require('../models/user')
const jwt = require('jsonwebtoken')
require('../utils/db')

const loginView = (req, res) => {
    res.render('../views/login-page', {
        layout : 'main',
        errors : req.flash('errors'),
    })
}

const loginAuth = async(req, res) => {
    const nim = req.body.nim
    const password = req.body.password

    if (nim === '030400' && password == '1234') {
        let admin = await User.findOne({uid : '030400'})
        req.session.userName = 'Admin'
        req.session._id = admin._id
        const token = jwt.sign({data : admin}, process.env.SECRET_KEY)
        res.cookie('x-access-token', token)
        return res.redirect('/admin')
    }
    const userLog = await User.findOne({
        uid : nim
    })

    if (!userLog) {
        req.flash('errors', 'errors')
         return res.redirect('/login')
    }

    let passwordVerify = await verifyPassword(password, userLog.password)

    if (passwordVerify) {
        const token = jwt.sign({data : userLog}, process.env.SECRET_KEY)
        res.cookie('x-access-token', token)
        return res.redirect('/')
    }
    req.flash('errors', 'errors')
    return res.redirect('/login')
}

const logout = (req, res) => {
    req.session.destroy(error => {
        res.clearCookie('x-access-token')
        res.redirect('/login')
    })
}

module.exports = {
    loginView,
    loginAuth,
    logout,
}
```

Gambar 4.13 Contoh Sintaks *Controller*

Pada Gambar 4.13 merupakan *code* atau sintaks dari salah satu contoh *controller* yaitu *controller login*. Untuk awal merupakan insialisasi *library* yang digunakan dalam *controller* ini yaitu *JSON Web Token* (JWT) dan *model* dari User. Kemudian baris selanjutnya merupakan *controller* untuk *login view*, dimana fungsi ini menerima dua parameter yaitu req (*request*)

dan res (*result*). Untuk baris selanjutnya merupakan sintaks untuk menampilkan *view* dari *login-page*.

4. *Middleware*

   Middleware merupakan perangkat lunak yang digunakan oleh aplikasi yang berbeda untuk berkomunikasi dengan satu sama lain. *Middleware* menyediakan fungsionalitas untuk menghubungkan aplikasi secara cerdas dan efisien sehingga. Contoh *code* atau *sintaks* dari *middleware* dalam sistem dapat dilihat pada Gambar 4.14.

```
require('dotenv').config()
const jwt = require('jsonwebtoken')

const adminVerify = (req, res, next) => {
    const token = req.cookies['x-access-token']
    if (!token) return res.redirect('/')
    const data = jwt.verify(token, process.env.SECRET_KEY)
    if (data.data.uid === '030400') return next()
    return res.redirect('/')
}


module.exports = {
    adminVerify
}
```

Gambar 4.14 Contoh Sintaks *Middleware*

Pada Gambar 4.14 merupakan *code* atau sintaks dari salah satu contoh *middleware* yaitu *middleware* admin. Untuk baris awal merupakan fungsi untuk melakukan pengecekan terhadap user apakah dia admin atau tidak. Jika dia adalah admin maka proses akan berlanjut dan jika pengguna bukan admin maka akan dikembalikan ke halaman utama atau halaman *login*.

5. *Routes*

   Routes merupakan arah atau tempat yang akan dituju pada sebuah *website* yang akan diakses. Dalam *routes* ini akan memiliki beberapa rute yang nantinya akan mengatur aplikasi sesuai dengan. Contoh *code* atau sintaks dari *routes* dalam sistem dapat dilihat pada Gambar 4.15.

```
const express = require('express')
const multer = require('multer')
const fileUpload = require('express-fileupload')

const {home} = require('../controllers/homeController')
const {loginView, loginAuth, logout} = require('../controllers/loginController')
const {admin, addUser, deleteUser, updateUser, updateUserByUser, pagination,
paginationUser, paginationFile} = require('../controllers/adminController')
const {uploadPdf, deletePdf, result, resultNone} =
require('../controllers/fileController')

const {toLogin, toAddUser, toUpdateUser, toUpdateUserByUser, verifyToken} =
require('../middleware/user')
const {adminVerify} = require('../middleware/admin')
const { toAddFile } = require('../middleware/file')
const { fileStorage, fileFilter, fileStorageRes } =
require('../config/fileConfig')

const uploadFile = multer({
    storage : fileStorage,
    fileFilter : fileFilter
}).single('upload')

const uploadFileRes = multer({
    storage : fileStorageRes,
    fileFilter : fileFilter
}).single('upload-res')

const router = express.Router()

router.get('/', home)
router.post('/result', uploadFileRes, result)
router.post('/result-none', fileUpload(), resultNone)
router.put('/user', toUpdateUserByUser, updateUserByUser)

router.get('/login', loginView)
router.post('/login', toLogin,  loginAuth)
router.get('/logout', logout)

router.post('/pdf', uploadFile, toAddFile, uploadPdf)
router.delete('/pdf', deletePdf)

router.get('/user', paginationUser)
router.get('/file', paginationFile)

router.use(adminVerify)
router.get('/admin', admin)
router.post('/admin-user', toAddUser, addUser)
router.put('/admin-user', toUpdateUser, updateUser)
router.delete('/admin-user', deleteUser)

module.exports = router
```

Gambar 4.15 Contoh Sintaks *Routes*

Pada Gambar 4.15 merupakan *code* atau sintaks dari salah satu contoh routes. Pada baris awal merupakan proses *import controller* dari folder controller yang ada dalam sistem. Untuk baris selanjutnya merupakan proses pembagian *routes* berdasarkan pada *controller* yang telah dibuat sebelumnya sesuai dengan *route* yang akan dihubungkan dengan *controller*.

6. *Utils*

Utils merupakan komponen tambahan yang nantinya akan dibutuhkan dalam sebuah aplikasi. Contoh *code* atau sintaks dari *utils* pada sistem dalam dilihat pada Gambar 4.16.

```
require('dotenv').config()
const mongoose = require('mongoose')
const URI = process.env.MONGODB_URL
const DB = 'latihan'

mongoose.connect(`${URI}/${DB}`, {
    useNewUrlParser : true,
    useUnifiedTopology : true,
})
```

Gambar 4.16 Contoh Sintaks *Utils*

Pada Gambar 4.16 merupakan *code* atau sintaks dari salah satu contoh utils yaitu *utils* db. Pada sintaks ini merupakan sintaks untuk menghubungkan *website* dengan *database*.

4.3.3 Sintaks *Dice Similarity*

Dice Similarity merupakan algoritma yang akan digunakan untuk membandingkan dokumen untuk mengetahui tingkat kesamaannya antar dokumen yang terdapat di dalam *database*. Untuk sintaks dari *Dice Similarity* dapat dilihat pada Gambar 4.17.

```
const data = res.data
const trainData = res.file
const user = res.user

let allData = {}
let all = []

addDict(data, all)
trainData.forEach(data => {
    addDict(data.text, all)
})
all = [...new Set(all)]
all.forEach(e => allData[e] = {})

data.split(' ').forEach(e => {
    allData[e]['s'] = (allData[e]['s'] || 0) + 1
})

trainData.forEach((str, i) => {
    str.text.split(' ').forEach(f => {
        allData[f]['s' + (i + 1)] = (allData[f]['s' + (i + 1)] || 0) + 1
    })
})

let resultAll = []
let averageSimilarity = 0
const keys = Object.keys(allData)

for (let i = 0; i < trainData.length; i++) {
    let e = 's' + (i + 1)
    let aTimesB = 0
    let aa = 0
    let bb = 0

    keys.forEach(f => {
        const a = allData[f]['s'] || 0
        const b = allData[f][e] || 0
        aTimesB += a * b
        aa += a * a
        bb += b * b
    })

    const result = (2 * aTimesB) / (aa + bb)
    resultAll.push({
        file: trainData[i],
        result: Math.round(result * 100)
    })
    averageSimilarity += Math.round(result * 100)
}
averageSimilarity /= trainData.length
averageSimilarity = Math.round(averageSimilarity)
```

Gambar 4.17 Sintaks *Dice Similarity*

Pada Gambar 4.17 merupakan sintaks untuk *Dice Similarity*. Untuk baris awal merupakan data yang akan digunakan sebagai pembanding dan data dari user yang telah diupload. Untuk variabel data merupakan *data* dari user yang telah diupload untuk variabel *trainData* merupakan data yang diambil dari *database* sistem. Pada baris selanjutnya merupakan variabel yang digunakan untuk menyimpan data yang akan dihitung kemiripannya di mana variabel *allData* akan menyimpan semua data

yang akan dibandingkan sedangkan variabel *all* untuk menyimpan semua kata yang akan dihitung terhadap semua data yang terdapat dalam *database*.

Untuk fungsi *addDict* merupakan fungsi untuk memasukkan kata ke dalam variabel *all*. Setelah itu merupakan baris untuk menghapus kata yang berulang dengan membuatnya ke dalam *Set* sehingga tidak ada kata yang terulang dalam variabel *all*. Setelah itu, variabel *all* akan menjadi *key* pada variabel *allData*. Tahap selanjutnya yaitu menghitung jumlah kata yang terdapat di dalam variabel *data* dan *trainData*. Setelah telah kata dari variabel data dan *trainData* telah dihitung maka akan dilakukan perhitungan sesuai dengan metode *Dice Similarity* pada Persamaan 2.1.

Untuk perhitungan, terlebih dahulu dilakukan perulangan sebanyak jumlah data yang terdapat pada variabel *trainData*. Kemudian dilakukan perbandingan terhadap data yang diupload oleh user dan data yang berasal dari *database*. Untuk mempermudah menghitung nilai Dice Similarity, dibuat terlebih dahulu beberapa variabel yaitu variabel *e*, *aTimesB*, *aa*, dan *bb*. Untuk variabel e merupakan urutan dari data dari variabel *dataTrain* yang akan bertambah sesuai dengan perulangannya. Selanjutnya, variabel *aTimesB* merupakan variabel untuk menyimpan nilai dari hasil perkalian vektor a dan b. Sedangkan untuk variabel *aa* dan *bb* untuk menyimpan hasil kali antara *a* dan *a*, dan juga untuk *b* dan *b*.

Setelah membuat variabel untuk menyimpan nilainya, dilakukan perhitungan *Dice Similarity* dengan cara melakukan perulangan terhadap *key* yang terdapat pada *trainData*. Untuk nilai *a* merupakan jumlah kata yang terdapat pada variabel data yang di awal, kemudian untuk *b* adalah variabel yang mempunyai nilai berupa jumlah kata yang sama dengan *a*. Misalkan untuk kata "rumah", maka variabel *a* akan mempunyai nilai berupa jumlah kata rumah pada variabel *data*, sedangkan variabel *b* akan mempunyai nilai berupa jumlah kata rumah yang terdapat pada sebuah data dalam *trainData*. Selanjutnya hasil dari perhitungan akan disimpan pada variabel *resultAll* berupa hasil nilai *Dice Similarity* kemudian dikalikan dengan 100 lalu dibulatkan untuk mendapatkan nilai kemiripan antara dua data tersebut. Untuk mendapatkan nilai rata-ratanya yaitu dengan cara menjumlah semua nilai *Dice Similarity* tiap *trainData* kemudian dibagikan dengan jumlah

trainData. Hasil *Dice Similarity* tiap-tiap dokumen nantinya akan ditampilkan ke dalam website.

4.4 *User Evaluation* (Evaluasi Pengguna)

Di tahap ini, sistem yang telah dibuat dalam bentuk prototipe akan dilakukan pengujian terhadap sistem yang telah dibuat. Untuk pengujian yang digunakan yaitu menggunakan metode pengujian *Black Box*. *Black Box* adalah pengujian perangkat lunak dari segi spesifikasi fungsional tanpa menguji kode atau sisi internal programnya. Artinya, hanya sisi fungsi, antarmuka, dan alurnya saja yang diuji tanpa menyentuh kode atau *script* dari perangkat lunak.

4.4.1 Rencana Pengujian

Tahap pengujian menggunakan metode Black Box akan menguji *website* yang akan dibuat dengan beberapa pengujian berdasarkan jenis pengguna yang akan mengakses *website* tersebut. Rencana pengujian secara spesifik dapat dilihat pada Tabel 4.1, Tabel 4.2, dan Tabel 4.3.

Tabel 4.1 Pengujian Pengguna *Guest*

No	*Requirement* Yang Diuji	Indikator	Pengujian
1	Akses Halaman Utama	Pengguna dapat mengakses halaman utama	Pengguna dapat mengakses *website*
2	*Check* Plagiarisme	Pengguna dapat melakukan pengecekan plagiarisme dalam *website*	Pengguna dapat melakukan pengecekan plagiarisme dengan *file* yang memiliki format *pdf*
			Pengguna tidak dapat melakukan pengecekan plagiarisme dengan *file* yang memiliki format selain *pdf*

Tabel 4.2 Pengujian Pengguna User

No	***Requirement* Yang Diuji**	Indikator	Pengujian
1	Akses Halaman Utama	Pengguna (*User*) dapat mengakses halaman utama	Pengguna (User) dapat mengakses *website*
2	Melakukan *Login*	Pengguna (*User*) dapat melakukan *login* ke dalam *website*	Pengguna (User) dapat melakukan proses *login* dengan NIM dan *Password* yang benar
			Pengguna (User) tidak dapat melakukan proses *login* dengan NIM dan *Password* yang salah
3	*Check* Plagiarisme	Pengguna (*User*) dapat melakukan pengecekan plagiarisme dalam *website*	Pengguna (User) dapat melakukan pengecekan plagiarisme dengan *file* yang memiliki format *pdf*
			Pengguna (User) tidak dapat melakukan pengecekan plagiarisme dengan *file* yang memiliki format selain *pdf*
4	Manajemen Dokumen	Pengguna (*User*) dapat melakukan manajemen dokumen yang akan digunakan sebagai perbandingan	Pengguna (User) dapat menambahkan dokumen yang memiliki format *pdf*
			Pengguna (User) tidak dapat menambahkan dokumen yang memiliki format selain *pdf*
			Pengguna (User) dapat menghapus

No	*Requirement* Yang Diuji	Indikator	Pengujian
			dokumen yang telah ditambahkan sebelumnya

Tabel 4.3 Pengujian Pengguna Admin

No	*Requirement* Yang Diuji	Indikator	Pengujian
1	Akses Halaman Utama	Pengguna (*Admin*) dapat mengakses halaman utama	Pengguna (Admin) dapat mengakses *website*
2	Melakukan *Login*	Pengguna (*Admin*) dapat melakukan *login* ke dalam *website*	Pengguna (Admin) dapat melakukan proses *login* dengan NIM dan *Password* yang benar
			Pengguna (Admin) tidak dapat melakukan proses *login* dengan NIM dan *Password* yang salah
3	*Check* Plagiarisme	Pengguna (*Admin*) dapat melakukan pengecekan plagiarisme dalam *website*	Pengguna (Admin) dapat melakukan pengecekan plagiarisme dengan *file* yang memiliki format *pdf*
			Pengguna (Admin) tidak dapat melakukan pengecekan plagiarisme dengan *file* yang memiliki format selain *pdf*
4	Manajemen Dokumen	Pengguna (*Admin*) dapat melakukan manajemen dokumen	Pengguna (Admin) dapat menambahkan dokumen yang

No	*Requirement* Yang Diuji	Indikator	Pengujian
		yang akan digunakan sebagai perbandingan	memiliki format *pdf*
			Pengguna (Admin) tidak dapat menambahkan dokumen yang memiliki format selain *pdf*
			Pengguna (Admin) dapat menghapus dokumen yang telah ditambahkan sebelumnya
5	Manajemen User	Pengguna (*Admin*) dapat melakukan manajemen *user*	Pengguna (Admin) dapat membuat *user* baru
			Pengguna (Admin) tidak dapat membuat *user* baru sesuai ketentuan
			Pengguna (Admin) dapat mengubah *user* yang sudah ada
			Pengguna (Admin) dapat menghapus *user* yang sudah ada

4.4.2 Tahap Pengujian

Setelah membuat rencana pengujian, selanjutnya dilakukan pengujian berdasarkan rencana pengujian yang telah dibuat. Untuk pengujian *website* dilakukan secara bertahap sesuai dengan rencana pengujian yang sudah ada. Dalam pengujian, nantinya akan didapat hasil berdasarkan pengujian apakah sesuai dengan pengujian yang dilakukan atau tidak. Jika tidak sesuai, nantinya akan dilakukan evaluasi sesuai rencana pengujiannya dan akan dilakukan perbaikan agar hasil pengujian yang dilakukan sesuai dengan yang diharapkan.

Untuk pengujian pertama yaitu pengujian pada pengguna *guest* yang sesuai pada Tabel 4.1. Berikut hasil pengujian untuk pengguna *guest* pada Tabel 4.4

Tabel 4.4 Hasil Pengujian Pengguna *Guest*

No	***Requirement* Yang Diuji**	Indikator	Pengujian	Hasil Pengujian
1	Akses Halaman Utama	Pengguna dapat mengakses halaman utama	Pengguna dapat mengakses *website*	Sesuai
2	*Check* Plagiarisme	Pengguna dapat melakukan pengecekan plagiarisme dalam *website*	Pengguna dapat melakukan pengecekan plagiarisme dengan *file* yang memiliki format *pdf*	Sesuai
			Pengguna tidak dapat melakukan pengecekan plagiarisme dengan *file* yang memiliki format selain *pdf*	Sesuai

Pengujian selanjutnya yaitu pengujian pada pengguna *User* yang sesuai dengan pada Tabel 4.2. Berikut hasil pengujian untuk pengguna *User* pada Tabel 4.5.

Tabel 4.5 Hasil Pengujian Pengguna *User*

No	***Requirement* Yang Diuji**	Indikator	Pengujian	Hasil Pengujian
1	Akses Halaman Utama	Pengguna (*User*) dapat mengakses halaman utama	Pengguna (User) dapat mengakses *website*	Sesuai
2	Melakukan *Login*	Pengguna (*User*) dapat	Pengguna (User) dapat	Sesuai

No	*Requirement* Yang Diuji	Indikator	Pengujian	Hasil Pengujian
		melakukan *login* ke dalam *website*	melakukan proses *login* dengan NIM dan *Password* yang benar	
			Pengguna (User) tidak dapat melakukan proses *login* dengan NIM dan *Password* yang salah	Sesuai
3	*Check* Plagiarisme	Pengguna (*User*) dapat melakukan pengecekan plagiarisme dalam *website*	Pengguna (User) dapat melakukan pengecekan plagiarisme dengan *file* yang memiliki format *pdf*	Sesuai
			Pengguna (User) tidak dapat melakukan pengecekan plagiarisme dengan *file* yang memiliki format selain *pdf*	Sesuai
4	Manajemen Dokumen	Pengguna (*User*) dapat melakukan manajemen dokumen yang akan digunakan sebagai perbandingan	Pengguna (User) dapat menambahkan dokumen yang memiliki format *pdf*	Sesuai
			Pengguna (User) tidak dapat menambahkan dokumen yang	Sesuai

No	*Requirement* Yang Diuji	Indikator	Pengujian	Hasil Pengujian
			memiliki format selain *pdf*	
			Pengguna (User) dapat menghapus dokumen yang telah ditambahkan sebelumnya	Sesuai

Pengujian terakhir yaitu pengujian pada pengguna *Admin* yang sesuai dengan pada Tabel 4.3. Berikut hasil pengujian untuk pengguna *Admin* pada Tabel 4.6.

Tabel 4.6 Hasil Pengujian Pengguna *Admin*

No	*Requirement* Yang Diuji	Indikator	Pengujian	Hasil Pengujian
1	Akses Halaman Utama	Pengguna (*Admin*) dapat mengakses halaman utama	Pengguna (Admin) dapat mengakses *website*	Sesuai
2	Melakukan *Login*	Pengguna (*Admin*) dapat melakukan *login* ke dalam *website*	Pengguna (Admin) dapat melakukan proses *login* dengan NIM dan *Password* yang benar	Sesuai
			Pengguna (Admin) tidak dapat melakukan proses *login* dengan NIM dan *Password* yang salah	Sesuai
3	*Check* Plagiarisme	Pengguna (*Admin*) dapat melakukan pengecekan	Pengguna (Admin) dapat melakukan pengecekan plagiarisme	Sesuai

No	*Requirement* Yang Diuji	Indikator	Pengujian	Hasil Pengujian
		plagiarisme dalam *website*	dengan *file* yang memiliki format *pdf*	
			Pengguna (Admin) tidak dapat melakukan pengecekan plagiarisme dengan *file* yang memiliki format selain *pdf*	Sesuai
4	Manajemen Dokumen	Pengguna (*Admin*) dapat melakukan manajemen dokumen yang akan digunakan sebagai perbandingan	Pengguna (Admin) dapat menambahkan dokumen yang memiliki format *pdf*	Sesuai
			Pengguna (Admin) tidak dapat menambahkan dokumen yang memiliki format selain *pdf*	Sesuai
			Pengguna (Admin) dapat menghapus dokumen yang telah ditambahkan sebelumnya	Sesuai
5	Manajemen User	Pengguna (*Admin*) dapat melakukan manajemen *user*	Pengguna (Admin) dapat membuat *user* baru	Sesuai

No	*Requirement* Yang Diuji	Indikator	Pengujian	Hasil Pengujian
			Pengguna (Admin) tidak dapat membuat *user* baru sesuai ketentuan	Sesuai
			Pengguna (Admin) dapat mengubah *user* yang sudah ada	Sesuai
			Pengguna (Admin) dapat menghapus *user* yang sudah ada	Sesuai

4.5 *Refining Prototype* (Memperbaiki Prototipe)

Setelah melakukan tahap sebelumnya, maka selanjutnya akan dilakukan perbaikan prototipe jika masih terdapat pengujian yang tidak sesuai dengan apa yang diharapkan. Proses ini akan terus dilakukan sampai sudah tidak ada lagi pengujian yang tidak sesuai dengan yang diharapkan.

Berdasarkan hasil pengujian pada Tabel 4.4, Tabel 4.5, dan Tabel 4.6, hasil pengujian yang diperoleh sesuai dengan rencana pengujian yang dibuat sehingga untuk memperbaiki prototipe ini tidak dilakukan perbaikan. Sehingga pada tahap ini tidak dilakukan.

4.6 *Implement Product and Maintain* (Implementasi dan Pemeliharaan)

Pada fase akhir ini, aplikasi akan segera diselesaikan berdasarkan prototipe akhir. Selanjutnya adalah fase pemeliharaan agar sistem berjalan lancar tanpa kendala pada waktu ke depannya.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan dan pembahasan sebelumnya, dapat diambil kesimpulan yaitu Aplikasi Pendeteksi Kesamaan Teks Berbasis *Website* dengan Mengimplementasikan Metode *Dice Similarity* telah selesai dirancang kemudian telah diuji menggunakan sebuah dokumen dengan hasil yang sesuai.

5.2 Saran

Berdasarkan penelitian yang dilakukan, maka untuk penelitian selanjutnya agar dapat dilakukan pengecekan dokumen berdasarkan jenis jurnal yang terdapat di dalam sistem untuk memperoleh kesamaan yang lebih akurat serta waktu perhitungan yang digunakan lebih sedikit.

DAFTAR PUSTAKA

Agung, G. (2000). *Microsoft Frontpage 2000 Webbot*. Jakarta: PT. Elex Media Komputindo.

Andaru, A. (2018). Pengertian Database Secara Umum. *Fakultas Komputer Umitra*.

Anonymous. (2022, July 15). *Express JS*. Diambil kembali dari Express - Node.js web application framework: https://expressjs.com/

Anonymous. (2022, July 15). *NodeJS*. Diambil kembali dari https://nodejs.dev/

Anonymous. (2022, July 15). *What Is MongoDB ?* Diambil kembali dari MongoDb: https://www.mongodb.com/what-is-mongodb

Arista, R. F., & Listyani, R. H. (2015). Plagiarisme di Kalangan Mahasiswa. *Paradigma.Volume 03 Nomor 02*.

Bianto, M. A., Rahayu, S., Huda, M., & Kusrini. (2018). Perancangan SIstem Pendeteksi Plagiarisme Terhadap Topik Penelitian Mengganakan Metode K-Means Clustering dan Model Bayesian. *Seminar Nasional Teknologi Informasi dan Multimedia*.

Deacon, j. (2009). Model-View-Controller (MVC) Architecture. *JOHN DEACON Computer Sistems Development, Consulting & Training*.

Dhamayanti, & Sari, L. P. (2019). Aplikasi Pendeteksi Plagiasi pada Universitas Indo Global Mandiri Berbasis Web. *Jurnal Ilmiah Informatika Global Volume 10 No. 02*.

Fikri, A. D. (2019). Perbandingan Metode Dice Similarity Dengan Cosine Similarity Menggunakan Query Expansion Pada Pencarian Ayatul Ahkam Dalam Terjemah Al Quran Berbahasa Indonesia .

Firmansyah, F. A. (2019, September 22). *NESABAMEDIA*. Diambil kembali dari Pengertian HTTPS Beserta Fungsi dan Perbedaannya dengan HTTP: https://www.nesabamedia.com/pengertian-https/

Hasanah, U., & Mutiara, D. A. (2019). Perbandingan Metode Cosine Similarity dan Jaccard Similarity Untuk Penilaian Otomatis Jawaban Pendek. *Seminar Nasional Sistem Informasi dan Teknik Informatika SENSITif*.

Khontoro, C., Andjarwirawan, J., & Yulia. (2021). Penerapan Algoritma TextRank dan Dice Similarity Untuk Verifikasi Berita Hoax. *Jurnal Infra Vol 9, No 1*.

Lopes, F. M. (2013). Penegakan Hukum Pelanggaran Terhadap Hak Cipta Dibidang Musik dan Lagu. *Lex Privatum, Vol.I/No.2/Apr-Jun/2013*.

Pratama, R. P. (2018). Aplikasi Pendeteksi Plagiarisme Menggunakan Cosine Similarity.

Pratama, R. P., Faisal, M., & Hanani, A. (2019). Deteksi Plagiarisme Pada Artikel Jurnal Menggunakan Metode Cosine Similarity . *SMARTICS Journal, Vol.5 No. 1* .

Priambodo, J. (2018). Pendeteksian Plagiarisme Menggunakan Algoritma Rabin - Karp dengan Meotde Rolling Hash. *Jurnal Informatika Universitas Pamulang Vol. 3, No. 1*.

Rahutomo, F., & Ririd, A. R. (2018). Evaluasi Daftar Stopword Bahasa Indonesia. *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK)*.

Ridhatillah, A. (2003). Dealing with Plagiarism in the Information Sistem Research Community: A Look at Factors That Drive Plagiarism and Ways to Address Them. *MIS Quarterly, Vol.27, No. 4, p. 511-532*.

Rusydi, I. (2014). Pemanfaatan E-Journal Sebagai Media Informasi Digital. *Jurnal Iqra' 8(2)*.

Sasongko, J., & Diartono, D. A. (2009). Rancang Bangun Sistem Informasi Manajemen Surat. *Jurnal Teknologi Informasi Dinamik*.

Sridevi, S. (2014). User Interface Design. *International Journal of Computer Science and Information Technology Research*.

Stefanovi, P., Kurasova, O., & Štrimaitis, R. (2019). The N-Grams Based Text Similarity Detection Approach Using Self-Organizing Maps and Similarity Measures. *Applied Science*.

Sunardi, Yudhana, A., & Mukaromah, I. A. (2018). Implementasi Deteksi Plagiarisme Menggunakan Metode N-Gram dan Jaccard Similarity Terhadap Algoritma Winnowing. *TRANSMISI, 20, (3)*.

Susanto, A. (2009). *Pengenalan Komputer.* Bekasi: IlmuKomputer.com.

Tala, F. Z. (2003). A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia. *Master of Logic Project Institute for Logic, Language and Computation Universiteit van Amsterdam The Netherlands*.

Wibowo, A. (2012). Mencegah dan Menaggulangi Plagiarisme di Dunia Pendidikan. *Jurnal Kesehatan Masyarakat Nasional Vol. 6, No. 5*.

Winarti, Titin, Jati, K., & Sunny, A. (2017). Determining Term on Text Document Determining Term on Text Document. *International Journal of Computer Application 157 (9):6*.