**Jurnal Indonesia : Manajemen Informatika dan Komunikasi**
https://journal.stmiki.ac.id

E - ISSN: 2723-7079 | P - ISSN: 2776-8074
Vol. 6 No. 1 (2025) | Januari

OPEN ACCESS

https://doi.org/10.35870/jimik.v6i1.1191

# Implementasi Algoritma Rabin-Karp dalam Pendeteksian *Plagiarisme* pada Dokumen Makalah Mahasiswa

Muhammad Rohyan Zidan [1], Kiki Setiawan [2*]

[1,2*] Program Studi Ilmu Komputer, Sekolah Tinggi Ilmu Komputer Cipta Karya Informatika, Kota Jakarta Timur, Daerah Khusus Ibukota Jakarta, Indonesia.

*Email*: rohyanzidan@gmail.com [1], kikisetiawan@gmail.com [2*]

**Histori Artikel:**
*Dikirim* 23 Oktober 2024; *Diterima dalam bentuk revisi* 13 November 2024; *Diterima* 20 Desember 2024; *Diterbitkan* 10 Januari 2025. 

## Abstrak

Tugas makalah merupakan karya ilmiah yang disusun oleh mahasiswa sebagai bagian dari penilaian dalam proses pembelajaran sesuai dengan jenjang pendidikan yang dijalani. Kekhawatiran terhadap meningkatnya kasus plagiarisme dalam dokumen akademik semakin relevan. Untuk menilai tingkat plagiarisme dalam dokumen, penggunaan sistem deteksi berbasis algoritma Rabin-Karp menjadi sangat penting. Algoritma ini digunakan untuk membandingkan kemiripan antara dokumen dalam format PDF dengan sumber yang telah ada, sehingga dapat mengidentifikasi pola yang menyerupai referensi aslinya. Sistem pendeteksi plagiarisme ini memberikan data yang relevan untuk mengukur tingkat keaslian suatu karya dan menentukan langkah-langkah pencegahan atau perbaikan yang diperlukan. Metode pengumpulan data yang diterapkan adalah studi literatur dan wawancara, sementara pengembangan sistem menggunakan pendekatan prototyping, yang memiliki tahapan-tahapan yang jelas, terstruktur, dan praktis.

**Kata Kunci:** Deteksi; Plagiarisme; Rabin-Karp.

## Abstract

Term papers are scholarly works prepared by students as part of their assessment in the educational process according to their academic level. Concerns about the increasing incidence of plagiarism in academic documents are becoming more significant. To assess the extent of plagiarism in a document, the use of a plagiarism detection system based on the Rabin-Karp algorithm is essential. This algorithm is employed to compare the similarities between a PDF document and existing sources, helping to identify patterns resembling the original references. This plagiarism detection system provides relevant data to measure the originality of a work and determine the necessary preventive or corrective actions. The data collection methods applied include literature review and interviews, while the system development follows the prototyping approach, which features clear, structured, and practical stages.

**Keyword:** Detection; Plagiarism; Rabin-Karp.

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E - ISSN: 2723-7079 | P - ISSN: 2776-8074
Vol. 6 No. 1 (2025) | Januari
OPEN ACCESS
https://doi.org/10.35870/jimik.v6i1.1191

## 1. Pendahuluan

Tugas makalah merupakan karya ilmiah yang disusun oleh mahasiswa sebagai bagian dari penilaian dalam proses pembelajaran sesuai dengan jenjang pendidikan yang dijalani. Makalah berfungsi untuk mengukur pemahaman mahasiswa terhadap materi yang telah diajarkan dan mengembangkan kemampuan berpikir kritis serta menulis ilmiah. Namun, pengelolaan data tugas makalah saat ini terbatas dan belum terintegrasi secara optimal, dengan dokumen sering disimpan dalam format PDF yang tidak mudah diproses untuk pengecekan lebih lanjut. Hal ini menyulitkan pengelolaan data baik bagi pengajar maupun mahasiswa. Masalah lain yang tidak kalah penting adalah meningkatnya *plagiarisme* di kalangan mahasiswa. Dengan kemajuan teknologi informasi yang pesat, akses ke berbagai sumber informasi semakin mudah, memungkinkan mahasiswa untuk menyalin atau *copy-paste* karya orang lain dengan cepat. Tindakan tersebut mengancam keaslian karya ilmiah dan merusak integritas akademik di perguruan tinggi. Oleh karena itu, pendeteksian *plagiarisme* menjadi sangat penting dalam menjaga kualitas akademik. Untuk mendeteksi *plagiarisme* dalam dokumen, diperlukan sistem yang efektif. Salah satu pendekatan yang dapat digunakan adalah algoritma Rabin-Karp. Algoritma ini bekerja dengan membandingkan kesamaan antara dokumen yang diperiksa dengan sumber-sumber lain yang tersedia. Dengan cara ini, algoritma dapat mengidentifikasi pola yang mirip dengan referensi asli dan memberikan informasi yang akurat mengenai tingkat kesamaan antara karya yang diuji dan sumber yang ada. Sistem deteksi berbasis algoritma Rabin-Karp memiliki keunggulan dalam memeriksa dokumen dalam jumlah besar, sehingga sangat cocok untuk digunakan dalam sistem pendeteksian *plagiarisme* pada skripsi atau tugas akademik lainnya (Rabin & Karp, 1987). Sistem pendeteksian *plagiarisme* yang memanfaatkan algoritma Rabin-Karp memungkinkan pengajar untuk mengevaluasi keaslian dokumen yang diserahkan mahasiswa. Sistem ini dapat memberikan informasi yang jelas mengenai tingkat kemiripan dokumen dengan karya yang telah dipublikasikan sebelumnya, sehingga mempermudah pengambil keputusan untuk memastikan bahwa dokumen tersebut benar-benar merupakan karya asli mahasiswa. Dengan demikian, algoritma ini dapat membantu mengidentifikasi *plagiarisme* dengan lebih cepat dan akurat.

Sistem ini memberikan umpan balik kepada mahasiswa mengenai keaslian karya mereka. Hal ini mendorong mahasiswa untuk lebih berhati-hati dalam mengutip sumber dan meningkatkan kesadaran mereka akan pentingnya menulis secara orisinal. Dalam jangka panjang, penggunaan sistem deteksi *plagiarisme* dapat mengurangi kasus *plagiarisme* dan meningkatkan kualitas karya ilmiah yang dihasilkan oleh mahasiswa. Penelitian ini menggunakan metode pengumpulan data melalui studi literatur yang mencakup referensi mengenai algoritma Rabin-Karp dan deteksi *plagiarisme*, serta wawancara dengan ahli di bidang teknologi informasi untuk memperoleh pandangan tentang penerapan sistem pendeteksian *plagiarisme* di lingkungan pendidikan. Pengembangan sistem dilakukan dengan pendekatan prototyping, yang memungkinkan evaluasi dan perbaikan sistem secara bertahap. Pendekatan ini memiliki tahapan yang jelas dan praktis, sehingga dapat menghasilkan sistem yang efektif dan sesuai kebutuhan pengguna (Pressman, 2014). Pengembangan sistem pendeteksian *plagiarisme* berbasis algoritma Rabin-Karp diharapkan dapat membantu meningkatkan efektivitas manajemen penulisan tugas di perguruan tinggi. Sistem ini bertujuan untuk mengurangi *plagiarisme* dan meningkatkan kualitas tugas akademik yang diserahkan oleh mahasiswa.

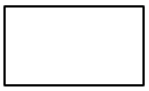
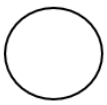
## 2. Metode Penelitian

Metode yang digunakan dalam penelitian ini melibatkan beberapa tahapan dan komponen yang saling berkaitan. Rancang bangun merupakan serangkaian prosedur yang digunakan untuk mentransformasikan hasil analisis suatu sistem ke dalam bahasa pemrograman, dengan tujuan menjelaskan secara rinci bagaimana komponen-komponen sistem diterapkan. Rancang bangun juga mencakup kegiatan untuk menerjemahkan hasil analisis ke dalam bentuk perangkat lunak dan mengembangkan atau memperbaiki sistem yang telah ada. Sistem, menurut Hidayah *et al.* (2020),

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 6 No. 1 (2025) | Januari
OPEN ACCESS
https://doi.org/10.35870/jimik.v6i1.1191

adalah sekumpulan komponen yang saling terhubung dan berinteraksi satu sama lain untuk mencapai tujuan tertentu. Sistem ini seringkali dibagi menjadi subsistem yang lebih kecil yang mendukung sistem utama. Penelitian ini juga mengaplikasikan konsep *website*, yang menurut Nurjani & Inda Aditya (2023) adalah sebuah media yang terdiri dari banyak halaman yang saling terhubung melalui *hyperlink*. *Website* memiliki fungsi untuk menyampaikan informasi dalam berbagai format, seperti teks, gambar, video, suara, dan animasi, atau gabungan dari semuanya. Terkait dengan pengembangan perangkat lunak, sistem yang dikembangkan menggunakan bahasa pemrograman yang terdiri dari sekumpulan instruksi standar untuk mengendalikan komputer, yang memiliki aturan sintaksis dan semantik untuk mendefinisikan program komputer (Syam *et al.*, 2024).

Pengembangan sistem ini juga menggunakan *Visual Studio Code*, editor kode sumber yang ringan namun kuat, yang dapat dijalankan di desktop dan tersedia untuk Windows, macOS, dan Linux. Program ini mendukung berbagai bahasa pemrograman seperti JavaScript, TypeScript, dan Node.js, serta memiliki ekosistem ekstensi yang luas untuk bahasa lain seperti C++, C#, Java, Python, PHP, Go, dan runtime seperti .NET dan Unity (Sofi & Dharmawan, 2022). Untuk menggambarkan alur proses dalam pengembangan sistem, digunakan *flowchart*, yang menurut Pemrograman (2020) adalah diagram yang menggambarkan langkah-langkah penyelesaian masalah secara logis dan berguna sebagai pedoman dalam pengembangan sistem. Diagram ini menggambarkan proses dengan orientasi dari atas ke bawah dan kiri ke kanan, memastikan bahwa setiap langkah dalam proses tersebut dinyatakan secara jelas dan tanpa ambigu. Selain itu, untuk menggambarkan alur data dalam sistem yang dikembangkan, digunakan *Data Flow Diagram* (DFD), yang menurut Satyaninggrat *et al.* (2023) adalah diagram yang menggunakan notasi-notasi untuk menggambarkan alur data. *DFD* sangat berguna untuk mempermudah pemahaman tentang bagaimana sistem bekerja secara logis dan terstruktur, serta membantu dalam mengidentifikasi masalah atau kekurangan dalam proses bisnis yang ada, sehingga dapat dilakukan perbaikan yang diperlukan.

| Notasi | Nama | Keterangan |
|---|---|---|
|  | Entitas Eksternal | Sistem yang menghasilkan informasi bagi transformasi oleh perangkat lunak atau menerima informasi uang dihasilkan oleh perangkat lunak. |
|  | Proses | Menunjukkan transformasi yang diaplikasikan ke data (kontrol) dan mengubahnya dengan berbagai macam cara. |
|  | Data Store | Tempat penyimpanan data dalam sistem |

Gambar 1. Simbol DFD

## 2.1 *Sequence Diagram*

*Sequence Diagram* menggambarkan interaksi antar objek dalam suatu *use case* dengan mendeskripsikan waktu hidup objek serta pesan yang dikirimkan dan diterima antar objek tersebut. Untuk menggambarkan *sequence diagram*, perlu diketahui objek-objek yang terlibat dalam *use case* beserta metode-metode yang dimiliki oleh kelas yang diinstansiasi menjadi objek tersebut. Dengan pemahaman yang jelas tentang objek dan metode ini, diagram sekuen dapat menggambarkan secara rinci alur komunikasi antar komponen dalam sistem yang sedang dianalisis. Diagram sekuen ini sangat berguna dalam menggambarkan urutan proses dan interaksi antar berbagai elemen dalam sistem, sehingga memberikan gambaran yang lebih jelas tentang cara kerja sistem secara keseluruhan. Gambar 2 menunjukkan contoh *sequence diagram* yang digunakan dalam penelitian ini.

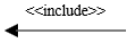
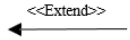
Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E - ISSN: 2723-7079 | P - ISSN: 2776-8074
Vol. 6 No. 1 (2025) | Januari
OPEN ACCESS
https://doi.org/10.35870/jimik.v6i1.1191

| Gambar | Nama | Keterangan |
|---|---|---|
|  | Entity Class | Gambaran sistem sebagai landasan dalam menyusun basis data |
|  | Boundary Class | Menangani komunikasi antar lingkungan sistem |
|  | Control Class | Bertanggung jawab terhadap kelas-kelas terhadap objek yang berisi logika |
|  | Recursive | Pesan untuk dirinya |
|  | Activation | Mewakili proses durasi aktivasi sebuah operasi |
|  | Life Line | Komponen yang digambarkan garis putus terhubung dengan objek |

Gambar 2. Simbol-simbol *Sequence Diagram*

## 2.2 *Use case*

*Use case* adalah diagram yang menggambarkan interaksi antara *use case* dan aktor. Aktor dapat berupa individu, perangkat, atau sistem lain yang berinteraksi dengan sistem yang sedang dikembangkan. Menurut Rudianto & Achyani (2022), *use case* menggambarkan fungsionalitas sistem atau persyaratan yang harus dipenuhi oleh sistem dari sudut pandang pengguna atau aktor. Diagram ini sangat berguna untuk mendefinisikan tujuan yang ingin dicapai oleh sistem, serta untuk mengidentifikasi dan menggambarkan berbagai skenario yang dapat terjadi dalam sistem. Gambar 3 menunjukkan contoh diagram *use case* yang digunakan dalam penelitian ini.

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

Gambar 3. Simbol *Usecase Diagram*

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 6 No. 1 (2025) | Januari
OPEN ACCESS
https://doi.org/10.35870/jimik.v6i1.1191

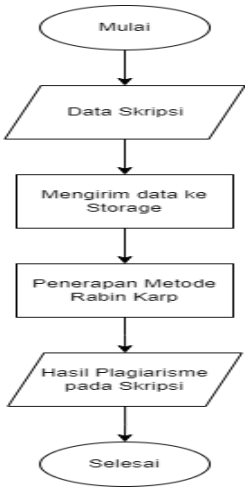
### 2.3 Algoritma Rabin-Karp

Algoritma Rabin-Karp adalah algoritma pencarian string yang ditemukan oleh Michael Rabin dan Richard Karp. Algoritma ini menggunakan *hashing* untuk menemukan sebuah substring dalam sebuah teks. *Hashing* adalah metode yang menggunakan fungsi hash untuk mengubah suatu jenis data menjadi beberapa bilangan bulat sederhana. Meskipun disebut sebagai algoritma "pencarian string", berbeda dengan algoritma "pencocokan string" seperti Knuth-Morris-Pratt atau Boyer-Moore, karena algoritma Rabin-Karp tidak bertujuan untuk menemukan string yang cocok dengan string masukan. Sebaliknya, algoritma ini digunakan untuk menemukan pola (*pattern*) yang sekiranya sesuai dengan teks masukan (Priambodo, 2018).


Gambar 4. Model Proses Rabin Karp

### 2.4 Model *Prototype*

*Prototyping* adalah metode pengembangan perangkat lunak yang berupa model fisik kerja sistem dan berfungsi sebagai versi awal dari sistem yang akan dibangun. Dengan metode *prototyping* ini, sebuah prototype sistem dihasilkan sebagai perantara antara pengembang dan pengguna, memungkinkan keduanya untuk berinteraksi dalam proses pengembangan sistem informasi. Agar proses pembuatan prototype ini berhasil dengan baik, sangat penting untuk mendefinisikan aturan-aturan sejak tahap awal, yaitu dengan memastikan bahwa pengembang dan pengguna memiliki pemahaman yang sama bahwa prototype dibangun untuk mendefinisikan kebutuhan awal sistem.




Gambar 5. Langkah metode pengembangan sistem berbasis *Prototype*

### 2.5 *Black Box testing*

*Black Box testing* adalah pengujian perangkat lunak yang berfokus pada spesifikasi fungsional tanpa menguji desain dan kode program, dengan tujuan untuk memastikan bahwa fungsi, masukan, dan keluaran perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Metode *Black Box Testing* merupakan salah satu metode yang mudah digunakan karena hanya memerlukan batas bawah dan batas atas dari data yang diharapkan. Estimasi banyaknya data uji dapat dihitung berdasarkan jumlah field data entri yang akan diuji, aturan entri yang harus dipenuhi, serta kasus batas atas dan batas bawah yang memenuhi kriteria (Cholifah *et al.*, 2018).

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E - ISSN: 2723-7079 | P - ISSN: 2776-8074
Vol. 6 No. 1 (2025) | Januari
OPEN ACCESS
https://doi.org/10.35870/jimik.v6i1.1191

### 2.6 *Beta Testing*

*Beta Testing* adalah pengujian yang dilakukan secara objektif dengan melibatkan pengguna secara langsung. Pengujian beta biasanya dilakukan dengan menggunakan kuesioner untuk mengumpulkan tanggapan pengguna terkait perangkat lunak yang telah dibangun. Metode penilaian pengujian yang digunakan dalam *Beta Testing* adalah metode kuantitatif yang didasarkan pada data yang diperoleh dari pengguna.

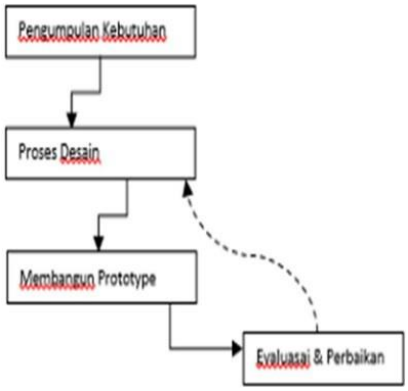
$$Y = \frac{P}{Q} x\ 100\%$$

Keterangan:
Y = Nilai Persentase
P = Banyaknya Jawaban Responden Tiap Soal
Q = Jumlah Responden

## 3. Hasil dan Pembahasan

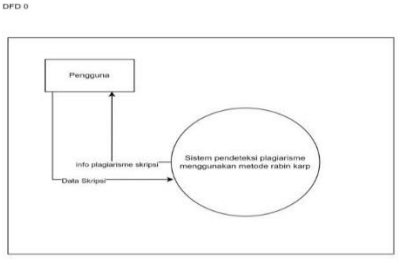
### 3.1 Hasil

Pada tahap rancangan sistem/*website* ini bertujuan untuk memberikan suatu gambaran umum tentang alur sistematis dari sistem tersebut.




Gambar 6. *Flowchart* Sistem Pendeteksi*Plagiarisme*

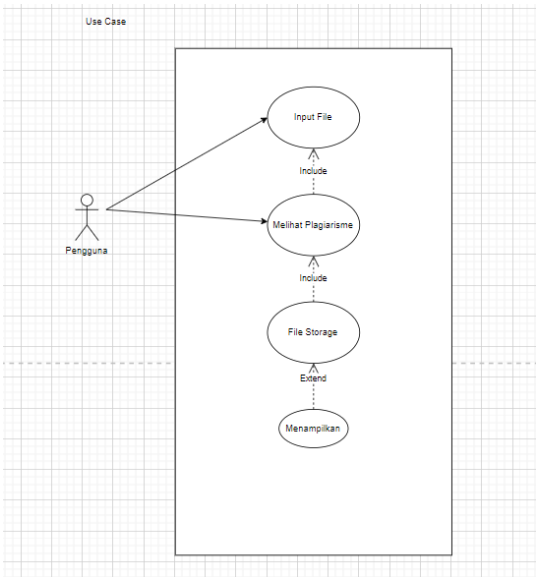
Pada gambar 6 terdapat *flowchart* sistem pendeteksi *plagiarisme* yang menggambarkan proses penginputan data skripsi yang diperoleh dari internet. Data tersebut kemudian akan disimpan di penyimpanan (storage) laptop yang digunakan. Selanjutnya, metode Rabin-Karp akan diterapkan untuk memproses file skripsi tersebut. Setelah proses selesai, hasil deteksi *plagiarisme* akan ditampilkan dalam bentuk persentase.




Gambar 7. DFD Sistem Pendeteksi *Plagiarisme*

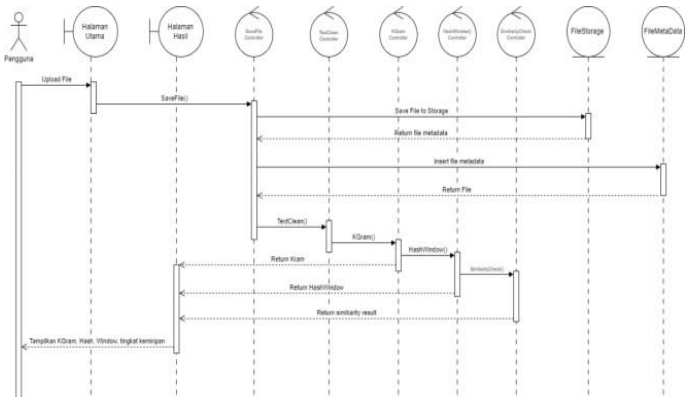
Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E - ISSN: 2723-7079 | P - ISSN: 2776-8074
Vol. 6 No. 1 (2025) | Januari
OPEN ACCESS
https://doi.org/10.35870/jimik.v6i1.1191

Pada gambar 7 terdapat *Data Flow Diagram* (DFD) 0 dari sistem pendeteksi *plagiarisme*. Pertama, pengguna akan menginputkan data skripsi ke dalam sistem, yang kemudian akan diproses menggunakan metode Rabin-Karp. Setelah proses selesai, sistem akan menampilkan informasi mengenai *plagiarisme* skripsi yang terdeteksi untuk pengguna.




Gambar 8. *Use Case* Sistem Pendeteksi *Plagiarisme*

Pada gambar 8 terdapat *use case diagram* dari sistem pendeteksi *plagiarisme*. Alur kerjanya dimulai dengan adanya *Input File*, yang kemudian diproses melalui sistem pendeteksi *plagiarisme*. Proses ini bertujuan untuk menganalisis isi file dan mendeteksi kemungkinan adanya *plagiarisme*. Setelah proses analisis selesai, file akan disimpan dalam komponen *File Storage*. Selanjutnya, hasil analisis *plagiarisme* akan ditampilkan melalui komponen tampilan.



Gambar 9. *Sequence Diagram* Sistem Pendeteksi *Plagiarisme*

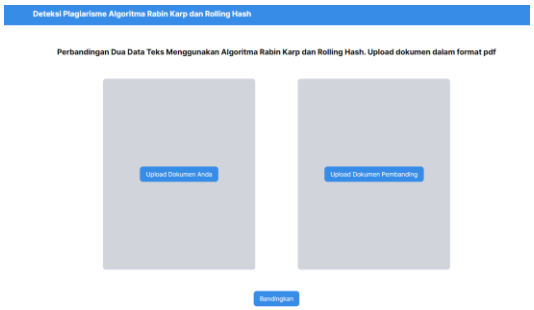
Pada gambar 9 terdapat *sequence diagram* yang menggambarkan alur proses di mana pengguna akan mengunggah file, yang kemudian disimpan di penyimpanan. Setelah file disimpan, file tersebut akan diambil untuk dilakukan pembersihan teks dari karakter atau elemen yang tidak diperlukan. Setelah teks dibersihkan, teks tersebut akan diubah menjadi hash, yaitu representasi unik dari teks asli dalam bentuk kode. Selanjutnya, hash tersebut akan dibandingkan dengan hash lainnya untuk mencari kesamaan. Hasil dari proses pencarian kesamaan ini akan dikembalikan kepada pengguna, memberikan informasi mengenai persentase *plagiarisme* yang terdeteksi.

**Jurnal Indonesia : Manajemen Informatika dan Komunikasi**
https://journal.stmiki.ac.id
E - ISSN: 2723-7079 | P - ISSN: 2776-8074
Vol. 6 No. 1 (2025) | Januari
OPEN ACCESS
https://doi.org/10.35870/jimik.v6i1.1191

3.1.1 Proses Algoritma Rabin-Karp

Pada proses algoritma Rabin-Karp dalam sistem pendeteksi *plagiarisme* ini terdapat beberapa tahapan, yaitu sebagai berikut:

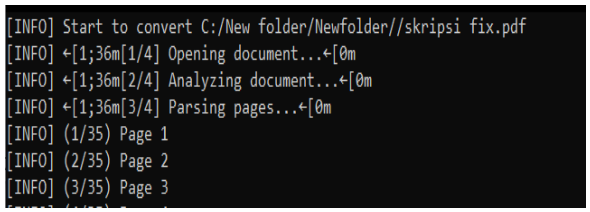
1) *Upload*



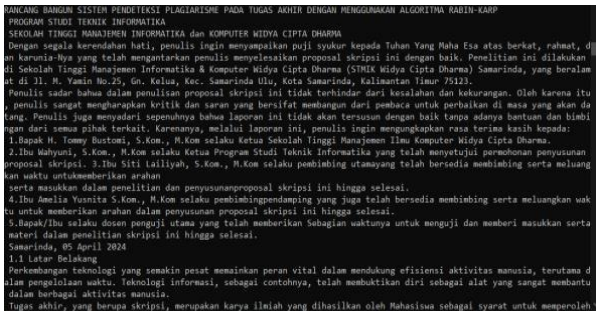

Gambar 10. Tahapan *Upload*

Pada gambar 10 merupakan tampilan awal dalam tahapan ini yang dimana pengguna mengupload dua file pdf dokumen skripsi yang ingin dilihat apakah terdapat *plagiarisme* dari kedua dokumen tersebut.

2) *Text Pre-Processing*




Gambar 11. Pengubahan dokumen pdf menjadi microsoft word


RANCANG BANGUN SISTEM PENDETEKSI PLAGIARISME PADA TUGAS AKHIR DENGAN MENGGUNAKAN ALGORITMA RABIN-KARP
PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TINGGI MANAJEMEN INFORMATIKA dan KOMPUTER WIDYA CIPTA DHARMA
Dengan segala kerendahan hati, penulis ingin menyampaikan puji syukur kepada Tuhan Yang Maha Esa atas berkat, rahmat, dj
an karunia-Nya yang telah mengantarkan penulis menyelesaikan proposal skripsi ini dengan baik. Penelitian ini dilakukan
di Sekolah Tinggi Manajemen Informatika & Komputer Widya Cipta Dharma (STMIK Widya Cipta Dharma) Samarinda, yang beralam
at di Jl. M. Yamin No.25, Gn. Kelua, Kec. Samarinda Ulu, Kota Samarinda, Kalimantan Timur 75123.
Penulis sadar bahwa dalam penulisan proposal skripsi ini tidak terhindar dari kesalahan dan kekurangan. Oleh karena itu
, penulis sangat mengharapkan kritik dan saran yang bersifat membangun dari pembaca untuk perbaikan di masa yang akan da
tang. Penulis juga menyadari sepenuhnya bahwa laporan ini tidak akan tersusun dengan baik tanpa adanya bantuan dan bimbi
ngan dari semua pihak terkait. Karenanya, melalui laporan ini, penulis ingin mengungkapkan rasa terima kasih kepada:
1.Bapak H. Tommy Bustomi, S.Kom., M.Kom selaku Ketua Sekolah Tinggi Manajemen Ilmu Komputer Widya Cipta Dharma.
2.Ibu Wahyuni, S.Kom., M.Kom selaku Ketua Program Studi Teknik Informatika yang telah menyetujui permohonan penyusunan
proposal skripsi. 3.Ibu Siti Lailiyah, S.Kom., M.Kom selaku pembimbing utamayang telah bersedia membimbing serta meluang
kan waktu untukmemberikan arahan
serta masukkan dalam penelitian dan penyusunanproposal skripsi ini hingga selesai.
4.Ibu Amelia Yusnita S.Kom., M.Kom selaku pembimbingpendamping yang juga telah bersedia membimbing serta meluangkan wak
tu untuk memberikan arahan dalam penyusunan proposal skripsi ini hingga selesai.
5.Bapak/Ibu selaku dosen penguji utama yang telah memberikan Sebagian waktunya untuk menguji dan memberi masukkan serta
materi dalam penelitian skripsi ini hingga selesai.
Samarinda, 05 April 2024
1.1 Latar Belakang
Perkembangan teknologi yang semakin pesat memainkan peran vital dalam mendukung efisiensi aktivitas manusia, terutama d
alam pengelolaan waktu. Teknologi informasi, sebagai contohnya, telah membuktikan diri sebagai alat yang sangat membantu
dalam berbagai aktivitas manusia.
Tugas akhir, yang berupa skripsi, merupakan karya ilmiah yang dihasilkan oleh Mahasiswa sebagai syarat untuk memperoleh

Gambar 12. Pengubahan file microsoft word menjadi text

```
def semua(data1, data2):

    data1 =data1.replace(' ', '')
    data2 =data2.replace(' ', '')
```

Gambar 13. Menghapus Semua Spasi

**Jurnal Indonesia : Manajemen Informatika dan Komunikasi**
https://journal.stmiki.ac.id
E - ISSN: 2723-7079 | P - ISSN: 2776-8074
Vol. 6 No. 1 (2025) | Januari
OPEN ACCESS
https://doi.org/10.35870/jimik.v6i1.1191

```python
def rolling_hash(s: str, panjang_huruf_diuji: int, angkaHash: int = 26, pembatas: int = 10**9 + 7):
    n = len(s)
    pangkat = [1]
    hash_values = []

    for i in range(1, n + 1):
        pangkat.append((pangkat[i - 1] * angkaHash) % pembatas)
    hash_baru = 0
    for i in range(panjang_huruf_diuji):
        hash_baru = (hash_baru * angkaHash + ord(s[i])) % pembatas

    hash_values.append(hash_baru)
    for i in range(1, n - panjang_huruf_diuji + 1):
        hash_baru = (
            hash_baru - pangkat[panjang_huruf_diuji - 1] * ord(s[i - 1])) % pembatas
        hash_baru = (hash_baru * angkaHash + ord(s[i + panjang_huruf_diuji - 1])) % pembatas

        hash_values.append(hash_baru)

    return hash_values
```

Gambar 14. Rumus *Rolling Hash*

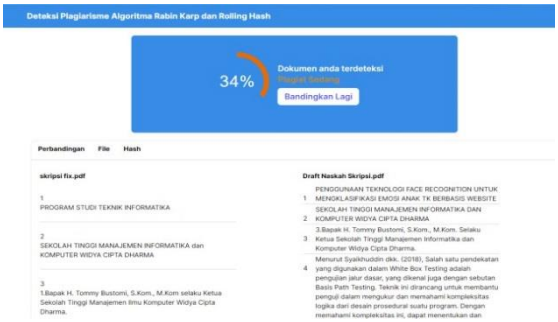
```
sudah ketemu plagiat sebanyak 0 hash dari 65 total hash
sudah ketemu plagiat sebanyak 37 hash dari 112 total hash
sudah ketemu plagiat sebanyak 53 hash dari 135 total hash
sudah ketemu plagiat sebanyak 53 hash dari 198 total hash
sudah ketemu plagiat sebanyak 53 hash dari 237 total hash
sudah ketemu plagiat sebanyak 53 hash dari 329 total hash
sudah ketemu plagiat sebanyak 53 hash dari 383 total hash
sudah ketemu plagiat sebanyak 53 hash dari 1784 total hash
sudah ketemu plagiat sebanyak 53 hash dari 1857 total hash
sudah ketemu plagiat sebanyak 68 hash dari 1885 total hash
sudah ketemu plagiat sebanyak 92 hash dari 1921 total hash
sudah ketemu plagiat sebanyak 119 hash dari 1955 total hash
sudah ketemu plagiat sebanyak 141 hash dari 1991 total hash
sudah ketemu plagiat sebanyak 160 hash dari 2024 total hash
sudah ketemu plagiat sebanyak 177 hash dari 2056 total hash
sudah ketemu plagiat sebanyak 200 hash dari 2092 total hash
sudah ketemu plagiat sebanyak 229 hash dari 2127 total hash
sudah ketemu plagiat sebanyak 250 hash dari 2157 total hash
sudah ketemu plagiat sebanyak 250 hash dari 2188 total hash
sudah ketemu plagiat sebanyak 269 hash dari 2225 total hash
sudah ketemu plagiat sebanyak 290 hash dari 2259 total hash
sudah ketemu plagiat sebanyak 290 hash dari 2299 total hash
sudah ketemu plagiat sebanyak 315 hash dari 2335 total hash
sudah ketemu plagiat sebanyak 340 hash dari 2373 total hash
sudah ketemu plagiat sebanyak 373 hash dari 2428 total hash
```

Gambar 15. Pencocokan Hash

```
sudah ketemu plagiat sebanyak 3929 hash dari 23661 total hash
sudah ketemu plagiat sebanyak 3929 hash dari 23695 total hash
sudah ketemu plagiat sebanyak 3929 hash dari 23727 total hash
sudah ketemu plagiat sebanyak 3929 hash dari 23762 total hash
sudah ketemu plagiat sebanyak 3949 hash dari 23794 total hash
sudah ketemu plagiat sebanyak 3967 hash dari 23824 total hash
sudah ketemu plagiat sebanyak 3967 hash dari 23859 total hash
sudah ketemu plagiat sebanyak 3967 hash dari 23896 total hash
sudah ketemu plagiat sebanyak 3967 hash dari 23918 total hash
sudah ketemu plagiat sebanyak 3967 hash dari 23952 total hash
sudah ketemu plagiat sebanyak 3967 hash dari 24073 total hash
sudah ketemu plagiat sebanyak 3967 hash dari 24250 total hash
sudah ketemu plagiat sebanyak 4008 hash dari 24326 total hash
sudah ketemu plagiat sebanyak 4008 hash dari 24342 total hash
kemiripannya adalah 16.465368498890808%
```

Gambar 16. Perhitungan Kemiripan Kata

Pada gambar 15 dilakukan proses pencocokan hash dari kedua dokumen yang bertujuan untuk mengukur kesamaan teks antara dua dokumen tersebut. Pada gambar 16 dilakukan perhitungan kemiripan kata dari semua hash yang didapatkan dari kedua dokumen tersebut. Semakin banyak nilai hash yang sama, semakin tinggi tingkat kemiripan kata. Algoritma ini akan menampilkan persentase kemiripan antara dua dokumen. Persentase ini menunjukkan seberapa mirip isi kedua dokumen. Semakin tinggi persentasenya, semakin mirip isi kedua dokumen.




Gambar 17. Hasil akhir Proses Sistem

Pada gambar 17 menampilkan text dalam bentuk paragraf-paragraf dari dokumen yang di proses sistem yang nilai hashnya terdeteksi plagiat dengan dokumen pembandingnya. Hasil akhir dari proses sistem yang dimana dapat disimpulkan bahwa *plagiarisme* yang terdeteksi dari kedua dokumen yang telah diupload dan diproses oleh sistem adalah 34%.

Jurnal Indonesia : Manajemen Informatika dan Komunikasi
https://journal.stmiki.ac.id
E-ISSN: 2723-7079 | P-ISSN: 2776-8074
Vol. 6 No. 1 (2025) | Januari
OPEN ACCESS
https://doi.org/10.35870/jimik.v6i1.1191

### 3.2 Pembahasan

Sistem pendeteksi *plagiarisme* yang dibangun menggunakan algoritma Rabin-Karp memiliki alur yang sistematis dan efisien dalam menganalisis kesamaan teks. Dalam prosesnya, sistem mengunggah file, membersihkan teks, kemudian mengubahnya menjadi *hash* untuk memudahkan perbandingan dengan dokumen lainnya. Algoritma Rabin-Karp dipilih karena kemampuannya yang efisien dalam memproses string dalam jumlah besar, serta sifatnya yang berbasis *rolling hash*, yang memungkinkan deteksi pola secara cepat dan akurat. Pada dasarnya, algoritma Rabin-Karp menggunakan teknik hashing untuk membandingkan bagian-bagian teks dalam waktu yang lebih cepat dibandingkan algoritma pencocokan pola lainnya, seperti Knuth-Morris-Pratt (Park & George, 1999). Metode ini mengubah teks menjadi *hash*, yang kemudian dibandingkan dengan *hash* dari dokumen lainnya. Proses ini memungkinkan sistem untuk mendeteksi kemiripan meskipun teks yang dibandingkan memiliki panjang yang besar atau terdapat perbedaan dalam struktur kalimatnya. Dalam penelitian yang dilakukan oleh Priambodo (2018), algoritma Rabin-Karp terbukti efektif dalam mendeteksi *plagiarisme*, terutama dalam analisis teks panjang, karena kecepatannya yang tinggi dalam pencocokan *hash*. Selain itu, sistem ini menggunakan berbagai komponen penting seperti *Data Flow Diagram* (DFD) dan *Use Case Diagram* untuk menggambarkan alur interaksi antara pengguna dan sistem. Dalam *DFD*, terlihat bahwa file yang diunggah oleh pengguna akan melalui proses pembersihan teks, yang bertujuan untuk menghilangkan elemen-elemen yang tidak relevan, seperti tanda baca atau karakter khusus, yang dapat memengaruhi hasil perbandingan. Hal ini sejalan dengan yang diungkapkan oleh Rahma & Taufiq (2023), yang menyatakan bahwa pembersihan teks merupakan langkah penting dalam memastikan akurasi deteksi *plagiarisme*, karena elemen-elemen yang tidak perlu dapat memperburuk hasil analisis.

Proses berikutnya adalah konversi teks menjadi *hash* yang unik. Penggunaan *hashing* untuk mendeteksi *plagiarisme* tidak hanya meningkatkan kecepatan proses, tetapi juga memungkinkan perbandingan yang lebih efisien, terutama ketika dibandingkan dengan metode lain seperti Jaro-Winkler Distance yang lebih rumit dalam pengimplementasiannya (Tinaliah & Elizabeth, 2018). Hasil analisis berupa *hash* yang dibandingkan dengan dokumen lain kemudian digunakan untuk menentukan tingkat kemiripan, yang ditampilkan sebagai persentase. Hal ini memberikan pengguna informasi yang jelas mengenai kemungkinan *plagiarisme* dalam dokumen yang diunggah. Selain itu, dalam penerapan *Black Box Testing*, sistem diuji berdasarkan fungsionalitasnya tanpa melihat implementasi internal (Cholifah *et al.*, 2018). Proses ini memastikan bahwa input yang diberikan oleh pengguna akan diproses sesuai dengan harapan dan menghasilkan output yang valid, yakni tingkat *plagiarisme* yang terdeteksi. Keakuratan deteksi *plagiarisme* ini sangat bergantung pada algoritma yang digunakan. Berdasarkan penelitian sebelumnya, seperti yang dilakukan oleh Alamsyah (2017) dan Nangi *et al.* (2020), algoritma Rabin-Karp memiliki keunggulan dalam mendeteksi kemiripan pada teks dengan efisiensi yang tinggi. Dibandingkan dengan algoritma lain seperti Winnowing, Rabin-Karp lebih unggul dalam menangani teks dengan panjang yang besar, yang sering ditemui dalam dokumen seperti skripsi atau tesis (Arnawa, 2021). Sistem pendeteksi *plagiarisme* yang menggunakan algoritma Rabin-Karp ini mampu memberikan hasil yang cepat dan akurat dalam mendeteksi kemiripan teks, yang merupakan langkah penting dalam menjaga integritas akademik. Penggunaan metode ini dapat diandalkan untuk membantu mahasiswa dan pengajar dalam memastikan keaslian karya ilmiah, sehingga mendorong terciptanya karya yang lebih orisinal dan bebas dari *plagiarisme*.

## 4. Kesimpulan

Sistem ini mampu mendeteksi *plagiarisme* secara efektif dengan memproses seluruh dokumen pada makalah mahasiswa. Proses yang dilakukan mencakup pengunggahan file, pembersihan teks, pengubahan teks menjadi *hash*, hingga perhitungan kesamaan teks berdasarkan *hash*. Dengan metode ini, sistem dapat memberikan hasil yang akurat dalam bentuk persentase *plagiarisme* yang terdeteksi, sehingga dapat digunakan sebagai alat bantu untuk menjaga keaslian karya ilmiah mahasiswa.

E - ISSN: 2723-7079 | P - ISSN: 2776-8074
Vol. 6 No. 1 (2025) | Januari
OPEN ACCESS
https://doi.org/10.35870/jimik.v6i1.1191

## 5. Daftar Pustaka

Alamsyah, N. (2017). Perbandingan Algoritma Winnowing Dengan Algoritma Rabin Karp Untuk Mendeteksi *Plagiarisme* Pada Kemiripan Teks Judul Skripsi. *Technologia: Jurnal Ilmiah*, *8*(3), 124-134. http://dx.doi.org/10.31602/tji.v8i3.1116.

Arnawa, I. B. K. S. (2021). KOMPARASI ALGORITMA WINNOWING DAN RABIN KARP MENDETEKSI KEMIRIPAN TUGAS MAHASISWA. *Jurnal Teknologi Informasi dan Komputer*, *7*(4). https://doi.org/10.36002/jutik.v7i4.1527.

Bramantya, A. Y., Hasanuddin, T., & Umar, F. (2022). Analisis Metode Winnowing dalam Pendeteksian *Plagiarisme* Judul. *Buletin Sistem Informasi dan Teknologi Islam (BUSITI)*, *3*(4), 268-273. https://doi.org/10.33096/busiti.v3i4.1469.

Cholifah, W. N., Yulianingsih, Y., & Sagita, S. M. (2018). Pengujian black box testing pada aplikasi action & strategy berbasis android dengan teknologi phonegap. *STRING (Satuan Tulisan Riset dan Inovasi Teknologi)*, *3*(2), 206-210. http://dx.doi.org/10.30998/string.v3i2.3048.

Darmanto, D., Pradasari, N. I., & Wahyudi, E. (2024). Sistem Deteksi *Plagiarisme* Tugas Akhir Mahasiswa Berbasis Natural Language Processing Menggunakan Algoritma Jaro-Winkler dan TF-IDF. *Smart Comp: Jurnalnya Orang Pintar Komputer*, *13*(1), 201-211. https://doi.org/10.30591/smartcomp.v13i1.6375.

Darnita, Y., & Aulia, N. (2020). Pengaruh Algoritma Stemming Porter Terhadap Kinerja Algoritma Rabin Karp Untuk Mendeteksi *Plagiarisme* Teks Bahasa Indonesia. *Journal of Technopreneurship and Information System (JTIS)*, *3*(2), 53-62.

Idris, I. S. K., & Mustofa, Y. A. (2022). Typo Checking Menggunakan Algoritma Rabin-Karp. *Jambura Journal of Electrical and Electronics Engineering*, *4*(1), 87-91. https://doi.org/10.37905/jjeee.v4i1.12150.

Nangi, J., Asmara, I. B. G. P., Sarita, M. I., Jaya, L. M. G., Mokui, H. T., & Tajidun, L. M. Perbandingan Algoritma Winnowing dan Algoritma Rabin-Karp pada Aplikasi Pendeteksi Kesamaan Dokumen Skripsi. *Jurnal Sistem Informasi Bisnis*, *14*(2), 131-142. https://doi.org/10.21456/vol14iss2pp131-142.

Park, J. H., & George, K. M. (1999). Efficient parallel hardware algorithms for string matching. *Microprocessors and Microsystems*, *23*(3), 155-168. https://doi.org/10.1016/S0141-9331(99)00032-0.

Pressman, A. (2018). *Design thinking: A guide to creative problem solving for everyone*. Routledge.

Priambodo, J. (2018). Pendeteksian *plagiarisme* menggunakan algoritma Rabin-Karp dengan metode Rolling Hash. *Jurnal Informatika Universitas Pamulang*, *3*(1), 39-45. https://doi.org/10.32493/informatika.v3i1.1518.

Rahma, S. L., & Taufiq, U. (2023). Analisis Tingkat Akurasi Metode Pendeteksian *Plagiarisme* Ide dengan menggunakan Yake dan Sentence Transformer. *Journal of Internet and Software Engineering*, *5*(1), 15-22. https://doi.org/10.22146/jise.v5i1.9073.

E - ISSN: 2723-7079 | P - ISSN: 2776-8074
Vol. 6 No. 1 (2025) | Januari
OPEN ACCESS
https://doi.org/10.35870/jimik.v6i1.1191

Tinaliah, T., & Elizabeth, T. (2018). Perbandingan Hasil Deteksi *Plagiarisme* Dokumen dengan Metode Jaro-Winkler Distance dan Metode Latent Semantic Analysis. *Jurnal Teknologi dan Sistem Komputer*, *6*(1), 7-12. https://doi.org/10.14710/jtsiskom.6.1.2018.7-12.

Yuliyanti, S. (2020). Implementasi Algoritma Rabin Karp Untuk Mendeteksi Kemiripan Dokumen Stmik Bandung. *Jurnal Bangkit Indonesia*, *10*(02), 1-1. https://doi.org/10.52771/bangkitindonesia.v10i02.124.