IDENTIFIKASI KEMIRIPAN DOKUMEN AKADEMIK PADA TUGAS KULIAH MAHASISWA DI JURUSAN ILMU KOMPUTER UNIVERSITAS LAMPUNG

(Skripsi)

Oleh

THERESIA TRI OKTAVIA IRMAWANTI
NPM 2217051111

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG
2026

ABSTRAK

IDENTIFIKASI KEMIRIPAN DOKUMEN AKADEMIK PADA TUGAS KULIAH MAHASISWA DI JURUSAN ILMU KOMPUTER UNIVERSITAS LAMPUNG

Oleh

THERESIA TRI OKTAVIA IRMAWANTI

Integritas akademik dalam lingkungan perguruan tinggi, terutama pada program studi ilmu komputer, sangat bergantung pada kemampuan mendeteksi kemiripan antar dokumen yang dikumpulkan mahasiswa, baik berupa laporan tertulis maupun kode program. Penelitian ini merancang dan membangun sebuah sistem deteksi kemiripan dokumen akademik yang mampu memproses dokumen teks berbahasa Indonesia dan kode sumber Python secara terintegrasi dalam satu platform. Pada tahap pemrosesan dokumen teks, digunakan kombinasi representasi berbasis statistik TF-IDF dan representasi berbasis semantik Word2Vec (FastText), dengan bobot 0,6 untuk TF-IDF dan 0,4 untuk FastText, yang selanjutnya dihitung kemiripannya menggunakan metode *Cosine Similarity*. Pendekatan ini dirancang agar sistem tidak hanya mampu menangkap kesamaan kata secara langsung, tetapi juga mengenali kemiripan makna meskipun kalimat telah mengalami parafrase. Sementara itu, pada pemrosesan kesamaan kode program, sistem menggunakan strategi penggabungan dua skor kemiripan, yaitu TF-IDF *Cosine Similarity* dan *Line-based Cosine Similarity*, untuk menghasilkan penilaian yang lebih akurat dan menyeluruh. Guna mengatasi tantangan efisiensi pada dataset berskala besar, algoritma *K-Means Clustering* diimplementasikan untuk mengelompokkan dokumen sehingga proses perbandingan hanya dilakukan di dalam klaster yang relevan. Hasil pengujian terhadap 90 dokumen teks dan 117 file kode Python memperlihatkan bahwa sistem mampu mendeteksi masing-masing 6 pasang dokumen dan kode yang memiliki kemiripan tinggi. Selain itu, penggunaan klasterisasi terbukti menekan jumlah komputasi perbandingan secara signifikan, yakni sebesar 85,8% untuk dokumen teks dan 83,7% untuk kode sumber, dengan peningkatan kecepatan proses mencapai 1,90x dan 6,86x. Sistem ini diimplementasikan dalam bentuk aplikasi web menggunakan framework Flask, yang menyediakan modul pemrosesan dokumen teks (SimDoc) dan kode sumber (SimCode), visualisasi bagian yang mirip serta fitur ekspor hasil analisis ke dalam format laporan PDF.

Kata Kunci: deteksi kemiripan dokumen; *cosine similarity*; *k-means clustering*; TF-IDF; *word2vec*.

ABSTRACT

IDENTIFICATION OF DOCUMENT SIMILARITIES IN ACADEMIC ASSIGNMENTS SUBMITTED BY STUDENTS IN THE DEPARTMENT OF COMPUTER SCIENCE

By

THERESIA TRI OKTAVIA IRMAWANTI

Academic integrity in higher education, particularly in computer science programs, relies heavily on the ability to detect similarities among documents submitted by students, whether in the form of written reports or programming code. This research designs and develops an academic document similarity detection system capable of processing Indonesian text documents and Python source code in an integrated manner within a single platform. For text document processing, a combination of statistical-based representation using TF-IDF and semantic-based representation using Word2Vec (FastText) is employed, with weights of 0.6 for TF-IDF and 0.4 for FastText, and similarity is subsequently measured using the Cosine Similarity method. This approach is designed so that the system is not only able to capture direct word-level matches, but also recognize semantic similarities even when sentences have been paraphrased. Meanwhile, for source code similarity detection, the system applies a dual-score fusion strategy combining TF-IDF Cosine Similarity and Line-based Cosine Similarity to produce a more accurate and comprehensive similarity assessment. To address computational efficiency challenges on large-scale datasets, the K-Means Clustering algorithm is implemented to group documents so that pairwise comparisons are performed only within relevant clusters. Experimental results on 90 text documents and 117 Python source code files demonstrate that the system successfully detected 6 pairs of similar documents and 6 pairs of similar source code with high similarity scores. Furthermore, the use of clustering significantly reduced the number of pairwise comparisons by 85.8% for text documents and 83.7% for source code, with processing speed improvements of 1.90x and 6.86x, respectively. The system is implemented as a Flask-based web application featuring a text document processing module (SimDoc) and a source code processing module (SimCode), along with similarity visualization and the ability to export analysis results as PDF reports.

Keywords: *document similarity detection; cosine similarity; k-means clustering; TF-IDF; word2vec.*

IDENTIFIKASI KEMIRIPAN DOKUMEN AKADEMIK PADA TUGAS KULIAH MAHASISWA DI JURUSAN ILMU KOMPUTER UNIVERSITAS LAMPUNG

Oleh

Theresia Tri Oktavia Irmawanti

Skripsi

Sebagai Salah Satu Syarat untuk Mencapai Gelar

SARJANA KOMPUTER

Pada

Jurusan Ilmu Komputer

Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS LAMPUNG

BANDAR LAMPUNG

2026

Judul Skripsi	:	**IDENTIFIKASI KEMIRIPAN DOKUMEN AKADEMIK PADA TUGAS KULIAH MAHASISWA DI JURUSAN ILMU KOMPUTER UNIVERSITAS LAMPUNG**
Nama Mahasiswa	:	Theresia Tri Oktavia Irmawanti
Nomor Pokok Mahasiswa	:	2217051111
Program Studi	:	S1-Ilmu Komputer
Fakultas	:	Matematika dan Ilmu Pengetahuan Alam

MENYETUJUI

1. Komisi Pembimbing

Dr. rer. nat. Akmal Junaidi, S.Si., M.Sc.	**Muhammad Galih Ramaputra, S.Kom., M.T.I.**
NIP. 197101291997021001	NIP. 199303192024061001

MENGETAHUI

2. Ketua Jurusan Ilmu Komputer	3. Ketua Program Studi S1Ilmu Komputer
Dwi Sakethi, S.Si., M.Kom.	**Tristiyanto, S.Kom., M.I.S., Ph.D.**
NIP. 196806111998021001	NIP. 198104142005011001

MENGESAHKAN

1. Tim Penguji

   Ketua : **Dr. rer. nat. Akmal Junaidi, S.Si., M.Sc.** .....

   Sekretaris : **Muhammad Galih Ramaputra, S.Kom., M.T.I.** .....

   Penguji Utama : **Tristiyanto, S.Kom., M.I.S., Ph.D.** .....

2. Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam

Dr. Eng. Heri Satria, S.Si., M.Si.
NIP. 197110012005011002

Tanggal Lulus Ujian Skripsi: **03 Juni 2026**

PERNYATAAN

Saya yang bertanda tangan di bawah ini.

Nama : Theresia Tri Oktavia Irmawanti

NPM : 2217051111

Dengan ini menyatakan bahwa skripsi saya yang berjudul **"Identifikasi Kemiripan Dokumen Akademik Pada Tugas Kuliah Mahasiswa Di Jurusan Ilmu Komputer Universitas Lampung"** merupakan hasil karya saya sendiri dan bukan karya orang lain. Seluruh tulisan yang tertuang di skripsi ini telah mengikuti kaidah penulisan karya ilmiah Universitas Lampung. Apabila di kemudian hari terbukti skripsi saya merupakan hasil penjiplakan atau dibuat orang lain, maka saya bersedia menerima sanksi berupa pencabutan gelar yang telah saya terima.

Bandar Lampung, 10 Juni 2026

Theresia Tri Oktavia Irmawanti

NPM. 2217051111

RIWAYAT HIDUP

Penulis dilahirkan pada tanggal 5 Oktober 2004 di Bandar Lampung sebagai putri ketiga dari pasangan Bapak Thomas Irwan Koeswanto (Alm) dan Ibu Magdalena Yuniarti, S.Pd. Penulis menempuh pendidikan formal tingkat Sekolah Dasar di SD Sejahtera II Bandar Lampung dan lulus pada Tahun 2016. Selanjutnya, pendidikan menengah pertama di SMP Negeri 19 Bandar Lampung hingga lulus pada Tahun 2019 dan melanjutkan ke pendidikan menengah atas di SMKN 1 Bandar Lampung yang diselesaikan pada Tahun 2022. Pada tahun yang sama penulis terdaftar sebagai mahasiswa Jurusan Ilmu Komputer FMIPA Universitas Lampung melalui jalur SBMPTN. Selama menjadi mahasiswa penulis melakukan beberapa kegiatan antara lain.

1. Menjadi Anggota Muda Ilmu Komputer (ADAPTER) Himpunan Mahasiswa Jurusan Ilmu Komputer pada periode 2022/2023.
2. Menjadi Anggota Bidang Kaderisasi Himpunan Mahasiswa Jurusan Ilmu Komputer pada periode 2022/2023.
3. Menjadi Bendahara Bidang Keilmuan Himpunan Mahasiswa Jurusan Ilmu Komputer pada periode 2023/2024.
4. Menjadi Asisten Dosen Jurusan Ilmu Komputer pada mata kuliah Logika, Sistem Operasi dan Komunikasi Data dan Jaringan Komputer.
5. Mengikuti Kuliah Kerja Nyata (KKN) periode 2024/2025 di Desa Panca Tunggal, Kecamatan Merbau Mataram Lampung Selatan.
6. Melaksanakan kerja praktik di PT. Pelabuhan Indonesia (Persero) Regional 2 Panjang divisi Teknik pada Tahun 2025.

MOTTO

"Janganlah hendaknya kamu kuatir tentang apapun juga, tetapi nyatakanlah dalam segala hal keinginanmu kepada Allah dalam doa dan permohonan dengan ucapan syukur"
(Filipi 4:6)

"Dan apa saja yang kamu minta dalam doa dengan penuh kepercayaan, kamu akan menerimanya"
(Matius 21:22)

"Mintalah, maka akan diberikan kepadamu; carilah, maka kamu akan mendapat; ketoklah maka pintu akan dibukakan bagimu."
(Matius 7:7)

"Karena masa depan sungguh ada, dan harapanmu tidak akan hilang."
(Amsal 23:18)

" Ora et Labora – Doakan apa yang kamu usahakan, usahakan apa yang kamu doakan"

PERSEMBAHAN

Puji dan syukur kepada Tuhan Yang Maha Esa atas segala anugerah, berkat dan penyertaan-Nya sehingga skripsi ini dapat terselesaikan dengan baik. Segala kemuliaan dan hormat penulis panjatkan kepada Tuhan Yesus Kristus atas kasih dan kekuatan yang senantiasa diberikan serta kepada Bunda Maria atas doa dan perlindungan yang selalu menyertai penulis selama proses penyusunan skripsi ini.

Dengan penuh rasa syukur dan terima kasih,
karya ini kupersembahkan kepada:

Ibuku tersayang, yang selalu memberikan dukungan, cinta dan kasih sayang yang tak terhingga serta do'a yang tiada henti menyertai setiap langkahku. Terima kasih sebesar-besarnya atas segala pengorbanan dan perjuangan dalam mendidik serta membesarkanku, yang tidak akan pernah dapat terbalaskan. Para sahabat dan rekan seperjuangan, yang telah menemani perjalanan ini dengan dukungan dan bantuan yang sangat berarti. Diri saya sendiri, atas keberanian dan kegigihan untuk terus melangkah hingga akhirnya mampu menyelesaikan studi pada Program Studi S1 Ilmu Komputer FMIPA Unila.

SANWACANA

Puji Syukur kepada Tuhan Yang Maha Esa, yang telah melimpahkan berkat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "**Identifikasi Kemiripan Dokumen Akademik Pada Tugas Kuliah Mahasiswa Di Jurusan Ilmu Komputer Universitas Lampung"** dengan baik dan lancar.

Penyusunan skripsi ini dapat terselesaikan berkat dukungan, bimbingan serta arahan dari berbagai pihak. Untuk itu, penulis mengucapkan rasa terima kasih yang setulus-tulusnya kepada:

1. Ibu tersayang, Magdalena Yuniarti sosok luar biasa yang telah membesarkan dan mendidik penulis dengan penuh cinta dan kesabaran. Terima kasih atas segala dukungan moral maupun materiil serta do'a yang tiada henti dipanjatkan demi kelancaran dan keberhasilan penulis dalam menyelesaikan studi ini. Setiap pencapaian penulis tidak terlepas dari peran besar Ibu.
2. Bapak Dr. Eng. Heri Satria, S.Si., M.Si. selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam.
3. Bapak Dwi Sakethi S.Si., M.Kom. selaku Ketua Jurusan Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung.
4. Bapak Tristiyanto, S.Kom., M.I.S., Ph.D., selaku Ketua Program Studi S1 Ilmu Komputer FMIPA Universitas Lampung sekaligus Dosen Pembahas yang telah memberikan kritik, saran dan masukan yang sangat bermanfaat bagi penyempurnaan skripsi ini.
5. Ibu Dewi Asiah Shofiana, S.Komp., M.Kom. selaku Pembimbing Akademik yang telah memberikan bimbingan, arahan serta nasihat kepada penulis selama menempuh pendidikan di perkuliahan ini.

6. Bapak Dr. rer. nat. Akmal Junaidi, S.Si., M.Sc. selaku Dosen Pembimbing Utama yang telah bersedia meluangkan waktu dan tenaga untuk memberikan bimbingan serta masukan yang sangat berharga kepada penulis selama proses penyusunan skripsi ini.
7. Bapak Muhammad Galih Ramaputra, S.Kom., M.T.I. selaku Pembimbing Kedua yang selalu bersedia meluangkan waktu dan tenaga untuk membimbing, memberikan arahan serta kritik yang sangat berharga dalam menyelesaikan penelitian ini.
8. Bapak dan Ibu Dosen Jurusan Ilmu Komputer FMIPA Universitas Lampung yang telah memberikan ilmu dan pengalaman yang sangat berharga selama penulis menempuh pendidikan di perkuliahan ini.
9. Seluruh staf dan karyawan Jurusan Ilmu Komputer, terkhusus Ibu Ade Nora Maela, Bang Zainuddin, Pak Dahud dan Mas Syam yang telah membantu segala urusan administrasi penulis di Jurusan Ilmu Komputer.
10. Kakak tersayang mas Agung yang senantiasa memberikan dukungan dan motivasi kepada penulis. Terima kasih atas perhatian, bantuan dan kasih sayang yang selalu diberikan.
11. Teman-teman seperjuangan semasa kuliah terkhusus Citra, Nazwa, Laras, Patiya, Elen, Atikah dan Pradya. Terima kasih atas kebersamaan yang telah terjalin selama masa perkuliahan. Terima kasih karena selalu menjadi tempat berbagi cerita, keluh kesah, kebahagiaan serta selalu ada untuk saling menguatkan.
12. Fitria Az Zahra selaku rekan seperjuangan skripsi, terima kasih atas kerja sama, kekompakan serta kerja keras yang telah dicurahkan dalam menyelesaikan penelitian ini. Terima kasih telah menjadi rekan diskusi yang baik, saling membantu dan saling menyemangati di setiap proses pengerjaan skripsi ini.
13. Sahabat dari kecil penulis Maria Stefani Kusuma Wardani, terima kasih atas persahabatan yang tulus dan selalu terjaga hingga saat ini. Terima kasih karena selalu hadir memberikan semangat dan dukungan untuk penulis.

14. Sahabat – sahabat tersayang Xandra, Bella dan Chika, terima kasih atas segala dukungan, perhatian dan kebersamaan yang telah diberikan kepada penulis.
15. Untuk partner penulis Rio Arisandi yang selalu setia menemani dan memberikan dukungan, semangat serta perhatian. Terima kasih telah menjadi tempat berbagi cerita, keluh kesah dan kebahagiaan, baik dalam suka maupun duka selama proses penyusunan skripsi ini.
16. Teman-teman Pimpinan Himakom 2024, terima kasih telah memberikan kesempatan dan pengalaman berharga bagi penulis untuk belajar dan berkembang dalam berorganisasi. Terima kasih atas kebersamaan, kerja keras serta pengalaman luar biasa yang telah dilalui bersama.
17. Keluarga besar Ilmu Komputer 2022, terima kasih atas pengalaman, dukungan dan kebersamaan dalam menyelesaikan studi di Jurusan Ilmu Komputer Universitas Lampung.
18. Keluarga Besar "Saeran Famz" terkhusus Biyung, Bude, mbk Lili, Om dan Bule, terima kasih atas kasih sayang, cinta, nasihat dan dukungan yang selalu diberikan untuk penulis.
19. Teman-teman KKN yaitu Alpina, Dinda, Gress, Apip, Angga dan Maldi terima kasih atas kebersamaan, kerja sama dan pengalaman berharga selama menjalankan program KKN.

Penulis menyadari skripsi ini masih jauh dari sempurna. Penulis berharap skripsi ini dapat memberikan manfaat dalam upaya menjaga integritas akademik, khususnya di Jurusan Ilmu Komputer Universitas Lampung.

Bandar Lampung, 10 Juni 2026

Theresia Tri Oktavia Irmawanti

NPM. 2217051111

DAFTAR ISI

DAFTAR TABEL

Tabel Halaman

DAFTAR GAMBAR

Gambar Halaman

I. PENDAHULUAN

1.1 Latar Belakang

Kemiripan isi dokumen dalam tugas mahasiswa merupakan fenomena yang semakin mendapat perhatian di lingkungan pendidikan tinggi. Tindakan seperti menjiplak atau menyalin karya tanpa tindakan yang tepat, yang termasuk dalam praktik plagiarisme, dapat merusak integritas akademik dan menurunkan kualitas pendidikan. Mahasiswa sebagai pelaku utama kegiatan akademik, kerap kali terlibat dalam tindakan plagiarisme, baik secara tidak disengaja karena kurangnya pemahaman, maupun secara sadar dalam menyusun tugas kuliah, laporan praktikum, hingga skripsi. Di Jurusan Ilmu Komputer Universitas Lampung sendiri, jumlah mahasiswa yang terus meningkat dari tahun ke tahun turut menyebabkan lonjakan dokumen tugas yang dikumpulkan. Kondisi ini menyulitkan dosen untuk melakukan proses pemeriksaan kemiripan manual secara menyeluruh dan objektif dalam waktu terbatas.

Sejumlah penelitian telah membuktikan bahwa metode *Term Frequency–Inverse Document Frequency* (TF-IDF*)* yang dikombinasikan dengan algoritma *Cosine Similarity* cukup efektif dalam mendeteksi tingkat kemiripan antar dokumen teks. TF-IDF banyak digunakan dalam bidang *text mining* dan *information retrieval* karena membantu sistem mengenali kata-kata yang paling relevan terhadap isi dokumen (Balani & Varol, 2021). Selanjutnya, *Cosine Similarity* digunakan untuk menghitung sejauh mana kemiripan antara dua dokumen berdasarkan sudut vektor representasi teks mereka. Jika dua dokumen memiliki sudut yang kecil *(nilai cosine mendekati 1)*, maka dianggap memiliki tingkat kemiripan yang tinggi.

Berdasarkan penelitian Azmi (2022) menunjukkan bahwa pendekatan ini mampu memberikan hasil analisis yang akurat dalam mendeteksi plagiarisme

pada dokumen skripsi. Senada dengan itu, penelitian oleh Resta et al. (2021) juga mendukung efektivitas metode ini dalam mengevaluasi tingkat kemiripan dokumen tugas akhir mahasiswa. Meskipun demikian, pendekatan tersebut lebih optimal untuk mendeteksi plagiarisme langsung atau *copy–paste*, dan masih menghadapi keterbatasan ketika plagiarisme dilakukan dalam bentuk parafrasa, yakni penggunaan sinonim atau variasi bahasa yang berbeda namun tetap menyampaikan makna yang sama.

Untuk mengatasi kelemahan tersebut, penelitian ini mengintegrasikan metode *Word2Vec pretrained* model dalam representasi dokumen. *Word2Vec* merupakan teknik *word embedding* yang menggunakan jaringan saraf untuk menghasilkan vektor kata dengan memperhatikan hubungan semantik antar kata (Dani & Puspaningrum, 2024**).** Dengan pendekatan ini, kata-kata yang memiliki makna serupa dipetakan ke dalam ruang vektor yang berdekatan (Andriyani et al., 2025). Hal ini memungkinkan sistem mendeteksi kemiripan semantik antar dokumen, termasuk pada kasus parafrasa. Ketika dikombinasikan dengan TF-IDF, representasi dokumen menjadi lebih kaya karena memadukan kekuatan statistik (pentingnya kata dalam dokumen) dengan semantik (makna kata). Integrasi TF-IDF, *Word2Vec*, dan *Cosine Similarity* diharapkan dapat meningkatkan akurasi deteksi plagiarisme, baik yang berbentuk *copy–paste* maupun parafrasa.

Meskipun integrasi TF-IDF dan *Word2Vec* dengan *Cosine Similarity* dapat meningkatkan akurasi dalam mendeteksi plagiarisme, pendekatan ini tetap menghadapi tantangan dari sisi efisiensi pemrosesan. Ketika jumlah dokumen yang dianalisis sangat banyak, proses pembandingan antar dokumen satu per satu *(pairwise comparison)* menjadi sangat berat secara komputasi. Permasalahan inilah yang kemudian menjadi *research gap* yang masih jarang ditangani secara optimal (Halim & Lasut, 2024)**.**

Sebagai solusi atas permasalahan efisiensi tersebut, penelitian ini mengusulkan integrasi algoritma *K-Means Clustering* dalam proses deteksi *plagiarisme. K-Means* merupakan algoritma *unsupervised learning* yang mengelompokkan

dokumen ke dalam beberapa klaster berdasarkan kesamaan fitur (Irianto et al., 2022). Dengan pendekatan ini, proses pembandingan kemiripan tidak lagi dilakukan pada seluruh dokumen, melainkan hanya difokuskan pada dokumen dalam klaster yang sama. Cara ini terbukti mampu menekan beban komputasi tanpa mengurangi akurasi analisis, sebagaimana ditunjukkan dalam penelitian Usino et al. (2019) yang menyatakan bahwa penggunaan *K-Means* dapat meningkatkan efisiensi proses penghitungan kemiripan dokumen secara signifikan.

Dari sisi kontribusi ilmiah, penelitian ini tidak hanya menghasilkan model teknis pendeteksian plagiarisme, tetapi juga menawarkan kebaruan baik secara teoritis maupun empiris. Secara teoritis, penelitian ini mengembangkan pendekatan pendeteksian kemiripan yang dirancang untuk menangani dua jenis data, yaitu dokumen teks dan *source code*, sehingga analisis tidak hanya mencakup kesamaan isi tulisan tetapi juga pola dan struktur kode program. Pendekatan ini memberikan cakupan yang lebih komprehensif dalam mengidentifikasi kesamaan tugas mahasiswa. Kombinasi fitur tersebut memungkinkan sistem mendukung kebutuhan analisis yang lebih luas dan relevan dengan karakteristik tugas di lingkungan Jurusan Ilmu Komputer.

Secara empiris, penelitian ini diwujudkan dalam bentuk *web* sederhana berbasis *Flask* yang berfungsi sebagai antarmuka untuk menjalankan model deteksi kemiripan yang dikembangkan. *Web* ini memungkinkan dosen atau asisten pengajar untuk mengunggah dan menganalisis dokumen tugas mahasiswa secara langsung tanpa memerlukan mekanisme autentikasi tambahan, sehingga proses pemeriksaan dapat dilakukan dengan cepat dan praktis. Meskipun dirancang dalam bentuk *web* sederhana, sistem ini tetap mampu merepresentasikan alur kerja pendeteksian kemiripan dokumen dan berpotensi dikembangkan lebih lanjut menjadi sistem internal yang mendukung aktivitas akademik di jurusan.

Penelitian terdahulu seperti yang dilakukan oleh Eka Putra & Supriana (2022) serta Saeed & Taqa (2022), menunjukkan bahwa integrasi antara teknik

pembobotan dan pengukuran kesamaan dapat dimanfaatkan untuk mendeteksi kemiripan, bahkan pada dokumen dengan struktur yang berbeda. Namun, penerapan pendekatan *clustering* untuk mengurangi beban proses analisis dalam konteks akademik lokal masih belum banyak dikembangkan. Oleh karena itu, penelitian ini diyakini mampu memberikan kontribusi nyata, baik dalam pengembangan ilmu pengetahuan di bidang *text mining* dan pemrosesan bahasa alami, maupun dalam mendukung upaya menjaga etika akademik di lingkungan perguruan tinggi.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, maka rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana menerapkan kombinasi metode TF-IDF dan Word2Vec dengan algoritma *Cosine Similarity* dan *K-Means Clustering* untuk mendeteksi kemiripan dokumen tugas mahasiswa secara efektif dan efisien?
2. Bagaimana menerapkan kombinasi representasi TF-IDF dan *Cosine Similarity* berbasis baris kode dengan algoritma *K-Means Clustering* untuk mendeteksi kemiripan *source code* secara akurat dan efisien?
3. Bagaimana mengintegrasikan kedua model analisis kemiripan dokumen teks dan *source code* ke dalam satu antarmuka web berbasis Flask yang dapat digunakan langsung oleh dosen atau asisten pengajar di Jurusan Ilmu Komputer Universitas Lampung?

1.3 Batasan Masalah

Agar penelitian ini lebih terfokus dan terarah, maka ditetapkan beberapa batasan masalah sebagai berikut:

1. Penelitian menganalisis dua jenis data, yaitu dokumen teks dan *source code*, yang merepresentasikan dua tipe tugas mahasiswa di Jurusan Ilmu Komputer.
2. Penelitian ini hanya memproses dan menganalisis konten berbasis teks. Elemen non-teks seperti gambar, grafik, diagram, dan konten visual

lainnya yang terdapat dalam dokumen akan diabaikan dan tidak digunakan dalam proses ekstraksi fitur maupun perhitungan kemiripan.

3. Dokumen teks yang dianalisis terbatas pada dokumen berbahasa Indonesia dalam format txt, pdf dan docx, yang diperoleh dari tugas mahasiswa angkatan 2022 pada mata kuliah Metodologi Penelitian.
4. Data *source code* yang dianalisis dibatasi pada file dengan bahasa pemrograman *Python* yang diperoleh dari tugas mahasiswa angkatan 2023 pada mata kuliah Struktur Data dan Algoritma.
5. Analisis kemiripan dilakukan secara berpasangan (*pairwise*), di mana hasil kemiripan yang dihasilkan merepresentasikan tingkat kesamaan antara dua dokumen atau dua file *source code*.
6. Sistem yang dikembangkan berupa antarmuka *web* sederhana berbasis *Flask*, berfungsi sebagai media untuk menjalankan model pendeteksian kemiripan.

1.4 Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah sebagai berikut:

1. Membangun model analisis kemiripan dokumen teks menggunakan kombinasi metode TF-IDF dan Word2Vec dengan algoritma *Cosine Similarity* dan *K-Means Clustering* yang mampu mendeteksi tingkat kesamaan isi tugas mahasiswa secara efektif dan efisien.
2. Mengembangkan model analisis kemiripan *source code* menggunakan kombinasi representasi TF-IDF dan *Cosine Similarity* berbasis baris kode dengan algoritma *K-Means Clustering* yang mampu mendeteksi kesamaan struktur dan pola penulisan kode Python secara akurat dan efisien.
3. Mengintegrasikan kedua model analisis kemiripan dokumen teks dan *source code* ke dalam satu antarmuka *web* berbasis *Flask* yang dapat digunakan langsung oleh dosen atau asisten pengajar di lingkungan Jurusan Ilmu Komputer Universitas Lampung.

1.5 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan manfaat sebagai berikut:

1. Memberikan kontribusi ilmiah dalam pengembangan metode analisis kemiripan dokumen teks dan *source code,* khususnya melalui pemanfaatan representasi vektor, *Cosine Similarity,* dan klasterisasi *K-Means* yang disesuaikan dengan karakteristik masing-masing jenis data.
2. Menambah referensi akademik terkait penerapan teknik *text mining, embedding* kata, dan pemrosesan kode program dalam konteks pendeteksian plagiarisme tugas mahasiswa di lingkungan perguruan tinggi.
3. Menyediakan model analisis kemiripan yang dapat digunakan sebagai alternatif ringan bagi institusi pendidikan untuk mendeteksi kesamaan isi dokumen teks maupun *source code* tanpa bergantung pada platform komersial.
4. Menghadirkan antarmuka web sederhana yang memudahkan dosen atau asisten pengajar dalam melakukan pemeriksaan kemiripan tugas mahasiswa secara cepat dan praktis sehingga proses evaluasi dapat dilakukan lebih efisien.
5. Mendukung upaya peningkatan integritas dan etika akademik, dengan memberikan alat bantu otomatis untuk mengidentifikasi potensi plagiarisme baik pada tugas berbasis tulisan maupun pemrograman.

II. TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penelitian mengenai deteksi kemiripan teks dan plagiarisme telah banyak dilakukan oleh para peneliti sebelumnya dengan menggunakan berbagai pendekatan dalam bidang *text mining* dan *natural language processing (NLP)*. Ringkasan penelitian-penelitian terdahulu yang relevan dengan topik ini dapat dilihat pada Tabel 1 berikut.

Tabel 1. Penelitian Terdahulu

Penelitian	Metode	Hasil	Kelemahan/*Research Gap*
Analisis Perbandingan Teknik *Word2Vec* dan *Doc2Vec* dalam Mengukur Kemiripan Dokumen Menggunakan *Cosine Similarity* (Iskandar & Kurniawati, 2025)	*Word2Vec, Doc2Vec,* dan *Cosine Similarity*	Pengujian pada 20 jurnal ilmiah berbahasa Indonesia menunjukkan *Word2Vec* menghasilkan nilai *Cosine Similarity* lebih tinggi (0,892) dibandingkan *Doc2Vec* (0,434) untuk pasangan jurnal yang sama, membuktikan *Word2Vec* lebih efektif dalam menangkap kemiripan semantik antar dokumen	Jumlah dataset relatif kecil (20 jurnal) dan belum menguji skalabilitas pada dokumen dalam jumlah besar. Penelitian ini juga belum mengintegrasikan mekanisme klasterisasi untuk meningkatkan efisiensi komputasi saat memproses dokumen berskala besar.
Document Plagiarism Detection Application Using Web-Based TF-IDF and Cosine Similarity	TF-IDF dan *Cosine Similarity*	Hasil konsisten antara perhitungan manual (4%) dan aplikasi (4,34%). Aplikasi berhasil diimplementasikan berbasis *web* untuk deteksi plagiarisme	Waktu pemrosesan TF-IDF dan *Cosine Similarity* cukup lama ketika jumlah data besar. Selain itu, metode yang digunakan hanya berbasis statistik sehingga belum mampu menangani

Penelitian	Metode	Hasil	Kelemahan/*Research Gap*
Methods (Halim & Lasut, 2024)		dokumen berbahasa Indonesia	kemiripan secara semantik. Penelitian ini juga belum mengintegrasikan mekanisme klasterisasi untuk pengelompokan dokumen yang lebih efisien.
Analisis Tingkat Plagiasi Dokumen Skripsi Dengan *Metode Cosine Similarity* dan Pembobotan TF-IDF (Azmi, 2022).	*Cosine Similarity* dan TF-IDF	Hasil pengujian menunjukkan kesesuaian antara perhitungan manual dan implementasi algoritma. Aplikasi juga berhasil memproses dokumen dengan waktu yang efisien, di mana 610 kata dapat diproses dalam waktu kurang dari 1 detik menggunakan *library* Sastrawi.	Hanya mendeteksi kemiripan berbasis statistik sehingga belum optimal menangani kemiripan secara semantik seperti kalimat yang diparafrase. Selain itu, sistem hanya membandingkan dua dokumen secara langsung tanpa mekanisme pengelompokan dokumen untuk skenario pengujian skala besar
Plagiarism Detection in Students' Theses Using The Cosine Similarity Method (Resta et al., 2021)	TF-IDF, *Cosine Similarity*	Tingkat kemiripan antara dokumen uji dan dokumen latih sebesar 8%, menunjukkan skripsi masih tergolong unik dan bebas plagiarisme. Sistem mampu membantu universitas dalam pengambilan keputusan penerimaan atau penolakan skripsi secara otomatis berdasarkan nilai *threshold* kemiripan 80%.	Dataset yang digunakan terbatas pada Google Scholar dengan mekanisme scraping yang rumit dan memakan waktu. Selain itu, sistem hanya membandingkan dokumen satu per satu tanpa mekanisme pengelompokan, sehingga kurang efisien untuk pendeteksian skala besar dengan banyak dokumen secara bersamaan.
Document Similarity Detection using K-Means and	*K-Means Clustering,* TF-IDF,	Akurasi deteksi kemiripan dokumen mencapai 93,33% dari 17 skenario pengujian	Dataset yang digunakan masih terbatas pada 17 dokumen, sehingga diakui penulis perlu diuji pada

Penelitian	Metode	Hasil	Kelemahan/*Research Gap*
Cosine Distance (Usino et al., 2019)	dan *Cosine Distance*	dengan tiga kategori artikel. Waktu *preprocessing* tercatat 7,997 ms untuk 17 dokumen dan pemrosesan *vector space* model rata-rata 119,296 ms, menunjukkan sistem bekerja dengan waktu yang efisien.	dataset yang lebih besar. Selain itu, sistem hanya mengandalkan representasi statistik berbasis TF-IDF sehingga belum mampu menangkap kemiripan secara semantik, yang menjadi peluang pengembangan dengan mengintegrasikan pendekatan berbasis embedding seperti Word2Vec.

Berdasarkan kajian terhadap penelitian-penelitian sebelumnya, dapat disimpulkan bahwa metode TF-IDF yang dikombinasikan dengan *Cosine Similarity* telah terbukti efektif dalam mendeteksi kemiripan dokumen akademik berbahasa Indonesia dengan tingkat akurasi yang cukup baik. Penelitian Resta et al. (2021) dan Azmi (2022) menunjukkan bahwa kombinasi kedua metode tersebut mampu memberikan hasil deteksi yang akurat dan konsisten antara perhitungan manual dengan implementasi sistem. Namun demikian, pendekatan ini memiliki keterbatasan signifikan dalam mendeteksi plagiarisme bentuk parafrasa, di mana mahasiswa menggunakan sinonim atau variasi bahasa yang berbeda namun tetap menyampaikan makna yang sama.

Untuk mengatasi kelemahan deteksi parafrasa, penelitian Iskandar & Kurniawati (2025) membuktikan bahwa teknik *Word2Vec* mampu menangkap kemiripan semantik antar dokumen dengan lebih baik dibandingkan pendekatan konvensional, menghasilkan nilai *Cosine Similarity* hingga 0.892 pada dokumen berbahasa Indonesia. Integrasi *Word2Vec* dengan TF-IDF diharapkan dapat memadukan kekuatan representasi statistik dan semantik dalam mengukur kemiripan dokumen. Sementara itu, penelitian Usino et al. (2019) menunjukkan bahwa penerapan *K-Means Clustering* dapat meningkatkan efisiensi proses deteksi dengan akurasi 93,33%, di mana

dokumen dikelompokkan terlebih dahulu sehingga proses perbandingan hanya dilakukan pada dokumen dalam klaster yang sama. Namun, penelitian-penelitian tersebut belum mengintegrasikan keseluruhan metode secara komprehensif untuk mengatasi permasalahan deteksi parafrasa dan efisiensi komputasi secara bersamaan.

Penelitian Halim & Lasut (2024) mengonfirmasi bahwa meskipun sistem deteksi plagiarisme berbasis TF-IDF dan *Cosine Similarity* dapat diimplementasikan dengan baik, namun menghadapi kendala dalam hal waktu pemrosesan ketika jumlah dokumen bertambah banyak. Hal ini menjadi permasalahan serius di institusi pendidikan tinggi seperti Jurusan Ilmu Komputer Universitas Lampung, di mana jumlah mahasiswa dan dokumen tugas terus meningkat setiap tahunnya. Oleh karena itu, terdapat *research gap* yang jelas mengenai perlunya sistem yang tidak hanya akurat dalam mendeteksi kemiripan dokumen berbahasa Indonesia termasuk dalam bentuk parafrasa, tetapi juga efisien dalam memproses dokumen dalam jumlah besar. Penelitian ini berupaya mengisi celah tersebut dengan mengintegrasikan TF-IDF, *Word2Vec, Cosine Similarity*, dan *K-Means Clustering* dalam satu sistem yang komprehensif, sehingga mampu mendeteksi plagiarisme baik dalam bentuk *copy-paste* maupun parafrasa dengan tetap menjaga efisiensi komputasi melalui mekanisme klasterisasi dokumen.

2.2 *Plagiarisme*

Istilah *plagiat* berasal dari kata dalam bahasa Inggris *plagiarism*, yang bermakna tindakan menjiplak. Secara etimologis, plagiat dapat dipahami sebagai perbuatan mengambil atau mencuri gagasan maupun karya intelektual orang lain, lalu mengakuinya sebagai hasil karya sendiri. Dengan kata lain, plagiat merupakan bentuk kecurangan akademik yang terjadi ketika seseorang menyalin ide atau tulisan orang lain tanpa mencantumkan sumber aslinya dan menyatakannya sebagai hasil pemikiran pribadi (Silalahi et al., 2024).

Menurut Halim & Lasut (2024), Plagiarisme terbagi ke dalam empat jenis utama, yang masing-masing memiliki karakteristik dan bentuk tindakan yang berbeda. Keempat jenis tersebut dijelaskan sebagai berikut:

1. *Verbatim Plagiarism*

   Jenis plagiarisme ini terjadi ketika seseorang menyalin karya atau gagasan orang lain secara utuh tanpa melakukan modifikasi sedikit pun. Dengan kata lain, isi yang diambil identik dengan sumber aslinya. Bentuk plagiarisme ini umumnya dikenal sebagai *copy-paste.*

2. *Patchwork Plagiarism*

   Patchwork Plagiarism adalah bentuk plagiarisme yang terjadi ketika seseorang menyusun tulisan dengan menggabungkan bagian-bagian dari berbagai sumber. Pelaku biasanya mengambil potongan kalimat atau ide dari beberapa penulis, lalu merangkainya menjadi satu teks baru yang tampak orisinal. Namun, sumber aslinya tidak disebutkan, sehingga karya tersebut tetap dianggap hasil penjiplakan meskipun sudah melalui proses penyusunan ulang.

3. *Paraphrase Plagiarism*

   Paraphrase Plagiarism terjadi ketika seseorang mengubah susunan kata atau gaya penulisan dari sumber asli menggunakan bahasa sendiri, namun tetap menyampaikan makna yang sama tanpa mencantumkan sumbernya. Meskipun terlihat berbeda secara struktur kalimat, isi dan ide pokoknya tetap berasal dari penulis lain, sehingga tindakan ini tetap termasuk plagiarisme.

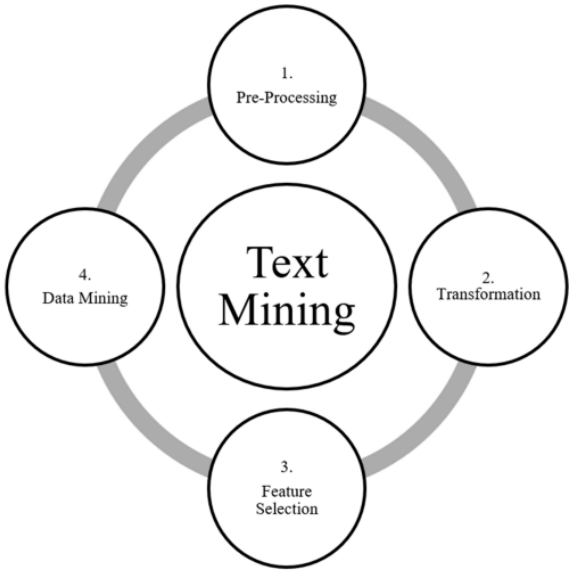
4. *Plagiarism of Keyword or Phrases*

   Pada jenis plagiarisme ini, pelaku menyalin beberapa kata kunci atau frasa penting dari karya orang lain, lalu merangkainya kembali menjadi tulisan baru yang tampak berbeda. Meskipun hasil akhirnya memiliki bentuk dan makna yang sedikit berubah, tindakan tersebut tetap dianggap plagiat karena tidak mencantumkan sumber asli.

2.3 *Text Mining*

Text mining merupakan metode untuk mengekplorasi dan menganalisis data teks yang bersifat tidak terstruktur dengan bantuan perangkat lunak yang dapat mengenali berbagai elemen seperti konsep, pola, topik, kata kunci, dan atribut lainnya dalam teks (Habib et al., 2025). Pemanfaatan *text mining* memungkinkan peneliti untuk mengekstraksi informasi bermakna dari kumpulan dokumen teks secara otomatis. Melalui proses ini, teks diolah untuk menemukan kata-kata yang paling mewakili isi dokumen sehingga hubungan antar dokumen dapat dianalisis secara lebih mendalam (Adek et al., 2022). Dengan demikian, *text mining* berperan penting dalam membantu analisis data berbasis teks, terutama ketika volume data yang diolah sangat besar dan tidak terstruktur. Dalam konteks penelitian ini, *text mining* digunakan untuk mengidentifikasi tingkat kemiripan antar dokumen tugas mahasiswa dengan memanfaatkan teknik representasi teks dan pengukuran kesamaan antar kata.

Proses *text mining* umumnya terdiri atas empat tahap utama seperti yang telihat pada Gambar 1, yaitu *text preprocessing*, *transformation*, *feature selection*, dan *data mining*. Tahap *text preprocessing* bertujuan mengubah teks mentah menjadi bentuk yang siap dianalisis melalui proses seperti *tokenization*, *normalization*, *stemming*, dan *lemmatization*. Selanjutnya, tahap *transformation* merepresentasikan dokumen ke dalam bentuk numerik menggunakan pendekatan seperti *vector space model* atau *bag of words (BoW)* agar dapat diproses secara matematis. Tahap *feature selection* kemudian digunakan untuk memilih kata-kata yang paling relevan guna meningkatkan akurasi dan efisiensi analisis. Terakhir, tahap *data mining* menggali pola dan hubungan dalam data teks menggunakan berbagai teknik seperti klasifikasi, klastering, atau pengukuran kemiripan (Chowdhury & Alzarrad, 2023).



Gambar 1. *Process Text Mining* (Chowdhury & Alzarrad, 2023).

Dalam penelitian ini, tahapan *text mining* tersebut diterapkan untuk mengubah dokumen tugas mahasiswa menjadi representasi vektor yang dapat dianalisis lebih lanjut menggunakan metode TF-IDF*, Word2Vec, Cosine Similarity,* dan *K-Means Clustering* guna mengidentifikasi tingkat kemiripan antar dokumen secara otomatis.

2.4 *Natural Language Processing (NLP)*

Natural Language Processing (NLP) merupakan cabang dari kecerdasan buatan (*Artificial Intelligence*) yang berfokus pada bagaimana komputer dapat memahami, memproses, dan menghasilkan bahasa manusia secara alami (Wahyudi, 2025). NLP terdiri dari serangkaian tahapan yang digunakan untuk mengonversi teks mentah menjadi bentuk numerik berupa vektor, sehingga memungkinkan sistem komputer mengenali struktur, makna, dan hubungan antar kata dalam suatu kalimat. Representasi ini membantu komputer menganalisis isi teks secara lebih efektif dan kontekstual (Ansis et al., 2024).

Salah satu langkah penting dalam proses NLP adalah *tokenizing*, yaitu proses memecah teks menjadi unit-unit yang lebih kecil yang disebut *token*. Langkah ini memudahkan sistem dalam mengenali elemen linguistik utama seperti

kata atau frasa, sehingga analisis dan pemrosesan teks dapat dilakukan dengan lebih terstruktur dan efisien (Wajhillah & Wibowo, 2022). Selain itu, tahapan seperti *stopword removal*, *stemming*, dan *lemmatization* juga dilakukan untuk menyederhanakan bentuk kata agar maknanya lebih mudah diolah. Dalam konteks penelitian ini, NLP berperan penting dalam mentransformasikan dokumen tugas mahasiswa menjadi representasi vektor yang dapat diolah menggunakan metode TF-IDF dan *Word2Vec*, sehingga sistem mampu mengukur tingkat kemiripan teks secara matematis maupun semantik.

2.5 *Machine Learning*

Machine Learning adalah bidang ilmu yang membuat komputer bisa belajar dari data tanpa harus diprogram secara langsung. Sistem ini menggunakan data sebagai contoh untuk mengenali pola dan membuat keputusan berdasarkan informasi yang ada. Tujuan utamanya adalah agar komputer dapat memahami pola yang kompleks dan menghasilkan keputusan cerdas secara otomatis (Kambey et al., 2020).

Menurut penelitian Adam (2025) *Machine Learning* terbagi menjadi 3 kategori, yaitu *supervised learning*, *unsupervised learning*, dan *reinforcement learning*. *Supervised Learning* adalah metode pembelajaran yang menggunakan data berlabel untuk memprediksi hasil pada data baru. *Unsupervised Learning* adalah metode yang digunakan untuk menemukan pola atau mengelompokkan data tanpa label, seperti pada algoritma *K-Means Clustering*. Sementara itu, *Reinforcement Learning* adalah metode yang memungkinkan sistem belajar melalui interaksi dengan lingkungan berdasarkan *reward* dan *punishment* untuk mencapai tujuan tertentu.

Dalam penelitian ini, fokus utamanya adalah pada *unsupervised learning,* khususnya penerapan *K-Means Clustering* untuk mengelompokkan dokumen berdasarkan tingkat kemiripan tanpa memerlukan label kelas. Dalam prosesnya, metode seperti TF-IDF dan *Word2Vec* digunakan pada tahap ekstraksi fitur untuk mengubah teks menjadi representasi numerik yang dapat

diproses oleh model *machine learning*. Dengan demikian, *machine learning* berperan penting dalam mendukung analisis kemiripan dokumen pada penelitian ini.

2.6 *Term Frequency – Inverse Document Frequency (TF-IDF)*

Metode TF-IDF *(Term Frequency–Inverse Document Frequency)* merupakan teknik yang digunakan untuk menentukan bobot atau tingkat kepentingan suatu kata *(term)* terhadap sebuah dokumen (Riyani et al., 2019). Menurut penelitian Septiani & Isabela (2022), metode TF-IDF menggabungkan dua konsep utama, yaitu *Term Frequency (TF)* dan *Inverse Document Frequency (IDF).*

Term Frequency (TF) menunjukkan seberapa sering suatu kata muncul dalam sebuah dokumen. Nilai TF biasanya dihitung dengan membagi jumlah kemunculan kata tersebut dengan total kata dalam dokumen, dan dapat disesuaikan menggunakan skema pembobotan tertentu. Rumus TF dapat dituliskan sebagai berikut:

$$TF = \frac{f(t,d)}{\sum f(w,d)} \tag{1}$$

Persamaan (1) menunjukkan bahwa nilai TF dihitung dengan membagi jumlah kemunculan kata (t) dalam dokumen (d) dengan total seluruh kata yang ada di dalam dokumen tersebut $\sum f(w,d)$. Nilai ini menggambarkan seberapa sering kata (t) muncul relatif terhadap panjang dokumen, sehingga kata yang muncul lebih sering akan memiliki bobot TF yang lebih tinggi.

Sementara itu, *Inverse Document Frequency (IDF)* menilai tingkat kepentingan suatu kata dalam keseluruhan kumpulan dokumen. Kata yang jarang muncul di seluruh dokumen akan memiliki nilai IDF yang lebih tinggi. Rumus IDF dituliskan sebagai berikut:

$$IDF = \log\left(\frac{N}{\mathrm{n}_t}\right) \tag{2}$$

Persamaan (2) diatas menunjukkan bahwa nilai IDF diperoleh dengan menghitung perbandingan antara jumlah seluruh dokumen (N) dan jumlah dokumen yang memuat kata t (n_t), kemudian mengambil nilai logaritmanya.

Setelah nilai TF dan IDF diperoleh, keduanya digabungkan untuk menghitung bobot akhir menggunakan metode TF-IDF, yang dirumuskan sebagai berikut:

$$TF - IDF = TF \times IDF \tag{3}$$

Persamaan (3) menunjukkan bahwa nilai $TF - IDF$ diperoleh dengan mengalikan nilai TF dan IDF. Hasil perkalian ini memberikan bobot akhir yang mencerminkan pentingnya suatu kata dalam sebuah dokumen sekaligus dalam keseluruhan korpus, sehingga kata yang sering muncul dalam dokumen tetapi jarang muncul di dokumen lain akan memiliki bobot TF-IDF yang lebih tinggi.

2.7 *Word2Vec*

Word2Vec merupakan algoritma yang digunakan untuk menghasilkan representasi kata dalam bentuk vektor, sehingga kata-kata dengan makna serupa akan ditempatkan pada posisi vektor yang berdekatan. Pendekatan ini membantu meningkatkan kinerja dalam proses *Natural Language Processing* (NLP) (Yanti & Utami, 2025). Model yang dikembangkan oleh Mikolov dan rekan-rekannya pada tahun 2013 ini memanfaatkan jaringan saraf dengan lapisan tersembunyi untuk mengubah kata menjadi representasi vektor padat. Kelebihan utama *Word2Vec* adalah kemampuannya dalam mengenali hubungan makna antar kata, seperti sinonim dan homonim (Rafli Aditya H et al., 2025).

Menurut penelitian Iskandar & Kurniawati (2025), *Word2Vec* memiliki dua model utama, yaitu *Continuous Bag of Words* (CBOW) dan *Skip-gram*. Model CBOW memprediksi suatu kata berdasarkan konteks kata-kata di sekelilingnya; model ini bekerja menggunakan jaringan saraf yang disederhanakan dengan menghilangkan *hidden layer non-linear* dan menggunakan lapisan proyeksi yang sama untuk setiap (Wahyu Kurniawan

& Maharani, 2020). Sebaliknya, *Skip-gram* memanfaatkan kata target sebagai masukan untuk memperkirakan kata-kata konteks yang kemungkinan muncul di sekitarnya, sehingga representasi vektor dibentuk berdasarkan hubungan antara kata target dan lingkungan terdekatnya (Dani & Puspaningrum, 2024).

2.7.1 *FastText* sebagai Pengembangan *Word2Vec*

FastText merupakan metode ekstraksi fitur kata yang merepresentasikan kata ke dalam bentuk vektor bilangan riil melalui pendekatan *word embedding* berbasis prediksi. Metode ini dikembangkan sebagai perluasan dari model *Skip-gram* dengan menambahkan informasi *subword.* Keunggulan *FastText* terletak pada efisiensi proses pelatihan pada korpus berskala besar serta kemampuannya dalam menghasilkan representasi vektor untuk kata yang tidak muncul selama proses pelatihan. Hal tersebut dimungkinkan karena *FastText* memanfaatkan karakter *n-gram* untuk membentuk representasi kata, sehingga *embedding* tetap dapat dihasilkan meskipun kata tersebut tidak terdapat dalam kosakata pelatihan (Rafli Aditya H et al., 2025).

2.7.2 *Pre-trained Word Embedding*

Pre-trained Word embedding adalah metode yang membentuk representasi vektor kata berdasarkan pola kemunculan kata dalam korpus berskala besar. Berbeda dengan *trainable embedding* yang dibangun dari awal menggunakan dataset terbatas, *pre-trained word embedding* diperoleh melalui proses pelatihan sebelumnya menggunakan metode *unsupervised learning* pada korpus luas. Oleh karena itu, model pra-latih ini memiliki informasi semantik yang lebih komprehensif serta stabil untuk digunakan dalam berbagai tugas pemrosesan bahasa alami (Susanty & Sahrul, 2021). Dalam penelitian ini, *Word2Vec* digunakan dalam bentuk *pre-trained word embedding* untuk membangun representasi semantik dokumen. Pendekatan ini dipilih karena lebih efisien dibandingkan melatih model dari awal,

terutama ketika jumlah data penelitian relatif terbatas. Model yang digunakan adalah *FastText pre-trained* Bahasa Indonesia, yang merupakan pengembangan *Word2Vec* berbasis *subword*, sehingga mampu menangkap hubungan makna antar kata secara lebih baik.

Dalam penelitian Susanty & Sahrul (2021) menunjukkan penggunaan *embedding* pra-latih seperti *Word2Vec* memberikan performa yang lebih baik dibanding *embedding* yang dilatih dari awal, karena bobotnya tidak diperbarui selama pelatihan sehingga lebih tahan terhadap *overfitting.* Selain itu, *pre-trained Word2Vec* berbasis *subword* mampu menangkap hubungan makna antar kata berdasarkan konteks kemunculan, sehingga representasi dokumen menjadi lebih akurat ketika digunakan pada tugas NLP seperti klasifikasi atau ekstraksi informasi. Dalam penelitian ini, *Word2Vec* pra-latih digunakan untuk membangun representasi dokumen teks sebelum dilakukan penghitungan kemiripan menggunakan *Cosine Similarity.*

Setelah setiap kata dalam dokumen diubah menjadi vektor *embedding*, representasi dokumen dibangun dengan menghitung rata-rata vektor kata (*mean pooling*). Pendekatan ini dilakukan dengan menjumlahkan seluruh vektor kata yang menyusun dokumen, kemudian membaginya dengan jumlah kata dalam dokumen tersebut, sebagaimana ditunjukkan pada persamaan 4 berikut:

$$\vec{d}_{w2v} = \frac{1}{n_d} \sum_{i=1}^{n_d} \vec{v}\,(w_i) \tag{4}$$

dengan keterangan:

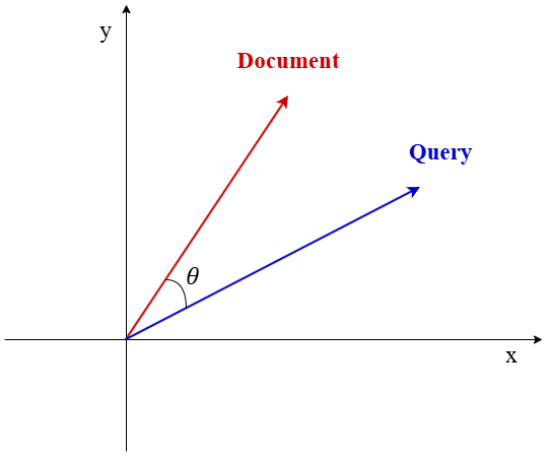
- $\vec{d}_{w2v}$ = vektor representasi dokumen berbasis *Word Embedding*,
- n_d = jumlah kata yang berhasil ditemukan dalam kosakata *FastText*,
- $\vec{v}(w_i)$ = vektor *FastText* dari kata ke- i

Melalui pendekatan *mean pooling* ini, setiap dokumen direpresentasikan sebagai satu vektor tunggal yang menggambarkan

makna semantik keseluruhan dokumen. Representasi ini selanjutnya digunakan sebagai dasar dalam proses pengelompokan dokumen dan penghitungan tingkat kemiripan antar dokumen menggunakan *Cosine Similarity.*

2.8 *Cosine Similarity*

Cosine Similarity merupakan metode yang digunakan untuk menilai tingkat kemiripan antara dua dokumen dengan menghitung nilai kosinus dari sudut antara dua vektor dalam ruang berdimensi (Rafli Aditya H et al., 2025). Semakin kecil sudut yang terbentuk antara kedua vektor, semakin tinggi tingkat kesamaannya. Nilai *Cosine Similarity* berada pada rentang 0 hingga 1, di mana nilai yang semakin mendekati 1 menunjukkan bahwa kedua dokumen memiliki tingkat kemiripan yang tinggi (Pawestri & Suyanto, 2024). Konsep *cosine similarity* dapat dilihat pada Gambar 2.



Gambar 2. Konsep *Cosine Similarity* (Lumbansiantar et al., 2023).

Gambar 2 di atas memperlihatkan ilustrasi konsep *Cosine Similarity*, di mana dua vektor, yaitu *Document* dan *Query*, digambarkan dalam ruang berdimensi untuk menunjukkan hubungan kemiripan antar dokumen. Pada ilustrasi tersebut, istilah *Query* tidak merujuk pada dokumen pencarian seperti pada konsep *information retrieval*, melainkan pada dokumen mahasiswa yang dibandingkan dengan dokumen lain untuk mengukur tingkat kemiripan. Sudut θ yang terbentuk antara kedua vektor menjadi dasar dalam perhitungan

nilai *cosine.* Semakin kecil sudut θ, semakin besar nilai cosinusnya, yang berarti tingkat kemiripan antara kedua dokumen semakin tinggi. Sebaliknya, semakin besar sudut θ, maka nilai *cosine* akan semakin kecil, menandakan bahwa kedua dokumen memiliki kesamaan yang rendah.

Dalam konteks penelitian analisis dokumen teks, *Cosine Similarity* digunakan untuk menghitung tingkat kemiripan antar dokumen mahasiswa berdasarkan representasi vektor yang dihasilkan dari metode *Word2Vec* dan TF-IDF, sehingga sistem dapat mengidentifikasi sejauh mana suatu dokumen memiliki kesamaan dengan dokumen lainnya.

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2 \times \sum_{i=1}^{n} (B_i)^2}} \quad (5)$$

Persamaan (5) menunjukkan rumus *cosine similarity,* di mana A dan B merepresentasikan dua vektor yang dibandingkan untuk mengukur tingkat kesamaan. Simbol $A.B$ menunjukkan perkalian *dot product* antara kedua vektor, sedangkan $|A|$ dan $|B|$ menyatakan panjang dari masing-masing vektor. Nilai $\|A\|\|B\|$ menunjukkan hasil perkalian panjang kedua vektor tersebut.

Dalam konteks analisis kemiripan file *source code*, *Cosine Similarity* diterapkan secara konsisten pada dua level analisis yang berbeda untuk mengukur kemiripan *source code.* Pada level dokumen, *Cosine Similarity* digunakan untuk mengukur kemiripan antara representasi vektor TF-IDF dari keseluruhan token *source code* dua file. Pada level baris, *Cosine Similarity* digunakan untuk mengukur kemiripan antara representasi baris-baris kode kedua file, yang dikenal sebagai *Line-based Cosine Similarity.*

2.8.1 *Line-based Cosine Similarity*

Line-based cosine similarity merupakan pendekatan pengukuran kemiripan *source code* yang bekerja pada level baris kode, di mana setiap baris kode program diperlakukan sebagai satu unit analisis yang berdiri sendiri. Berbeda dengan *cosine similarity* konvensional yang

menilai kesamaan berdasarkan distribusi token pada keseluruhan dokumen, pendekatan ini berfokus pada perbandingan langsung antar baris kode. Dengan cara ini, kemiripan implementasi dapat diamati secara lebih spesifik pada struktur dan isi tiap baris.

Pendekatan berbasis baris kode telah diakui sebagai metrik yang relevan dalam deteksi *plagiarism source code.* Penelitian oleh Gandhi et al. (2024) menunjukkan bahwa pengukuran kemiripan pada level struktur vertikal kode, yakni dengan membandingkan susunan baris-baris kode yang dapat dieksekusi, digunakan sebagai salah satu komponen penting dalam sistem deteksi plagiarisme *source code* yang dikembangkan. Selain itu, Mehsen & Joshi (2024) juga menegaskan bahwa pengelompokkan baris – baris kode yang memiliki tingkat kemiripan tinggi menjadi dasar penting dalam proses identifikasi plagiarisme pada *source code.*

Kelebihan utama *Line-based Cosine Similarity* terletak pada kemampuannya mendeteksi kesamaan langsung pada level implementasi baris kode. Selain itu, pendekatan ini mempertimbangkan bobot TF-IDF setiap baris, sehingga baris yang unik dan hanya muncul pada satu *file* mendapat bobot IDF yang lebih tinggi dan berkontribusi lebih besar terhadap penurunan skor kemiripan (Karnalim, 2020). Dengan demikian, *Line-based Cosine Similarity* lebih sensitif terhadap perbedaan implementasi antar *file source code* dibandingkan hanya mengandalkan *Cosine Similarity* berbasis distribusi token secara keseluruhan.

Dalam penelitian ini, *Line-based Cosine Similarity* diintegrasikan bersama *TF-IDF Cosine Similarity* level dokumen untuk membentuk skor kemiripan akhir (*final score*) yang lebih menyeluruh. Penggabungan beberapa indikator kemiripan melalui kombinasi linear merupakan pendekatan yang umum dan mudah dipahami dalam pengembangan sistem deteksi plagiarisme *source code* (Gandhi et al., 2024). Menurut Martinez-Gil (2024), skor akhir dapat dihitung

menggunakan rata-rata terbobot dari kedua metrik tersebut, sebagaimana dinyatakan pada Persamaan (6) berikut:

$$S_{final} = \alpha \cdot S_{doc} + (1 - \alpha) \cdot S_{line} \quad (6)$$

dengan keterangan:

- S_{final} = skor kemiripan akhir gabungan,
- S_{doc} = nilai *TF-IDF Cosine Similarity* level dokumen,
- S_{line}= nilai *Line-based Cosine Similarity* level baris,
- α = parameter bobot untuk *TF-IDF Cosine Similarity*.

Penetapan bobot antar metrik dalam sistem deteksi *plagiarisme source code* bukanlah hal yang sederhana dan perlu dirancang secara cermat (Gandhi et al., 2024). Pemilihan nilai bobot sebaiknya mempertimbangkan karakteristik masing-masing representasi. TF-IDF berbasis token mampu merepresentasikan struktur kode secara menyeluruh pada tingkat global sehingga memberikan gambaran yang lebih komprehensif. Di sisi lain, *Line-based Cosine Similarity* berperan sebagai komponen pendukung yang memperkuat analisis dengan menangkap kemiripan secara langsung pada tingkat implementasi tiap baris kode (Karnalim, 2020).

2.9 K-Means Clustering

Clustering merupakan metode analisis yang digunakan untuk mengelompokkan data berdasarkan tingkat kesamaan tertentu. Proses ini melibatkan pengelompokan data, objek, atau observasi dengan karakteristik yang mirip ke dalam satu kelompok (Fitriyah Salsabilah et al., 2024). Salah satu metode yang banyak digunakan adalah *K-Means. K-Means* merupakan salah satu metode pengelompokan data *non-hierarki* yang berfungsi untuk membagi data ke dalam dua atau lebih kelompok. Metode ini bekerja dengan mengelompokkan data yang memiliki karakteristik serupa ke dalam satu *cluster* yang sama, sedangkan data dengan karakteristik berbeda akan dimasukkan ke dalam *cluster* yang lain (Gustientiedina et al., 2019).

K-Means termasuk dalam kategori *unsupervised learning* karena mampu mengelompokkan data tanpa memerlukan label atau kategori tertentu (Bili et al., 2024). Algoritma ini bekerja dengan menentukan sejumlah *centroid* atau titik pusat *cluster* secara acak, kemudian setiap data akan dihitung kedekatannya terhadap tiap *centroid* untuk menentukan keanggotaan *cluster* (Fatkhudin et al., 2023). Proses pengelompokkan data ke dalam beberapa *cluster* dilakukan berdasarkan tingkat kesamaan fitur, yang dimulai dengan penentuan jumlah *cluster* optimal melalui metode *Elbow Criterion,* yaitu dengan memilih nilai k berdasarkan titik penurunan signifikan pada grafik *Sum of Squared Error (SSE).*

Dalam implementasi standarnya, *k-means* menggunakan *euclidean distance* untuk mengukur kedekatan antar data dengan meminimalkan SSE sebagai fungsi objektif. Namun metrik ini memiliki keterbatasan ketika diterapkan pada data teks yang direpresentasikan sebagai vektor berdimensi tinggi. Jarak *Euclidean* sangat sensitif terhadap *magnitudo* atau panjang vektor, yang dalam konteks dokumen teks seringkali hanya merepresentasikan panjang dokumen dan bukan kesamaan topiknya. Karakteristik vektor teks yang telah dinormalisasi membuat pengukuran berbasis sudut lebih relevan dibandingkan pengukuran berbasis jarak.

Untuk mengatasi keterbatasan ini, digunakan pendekatan lain yang dikenal sebagai *Spherical K-Means (SPKM)***,** yang menggantikan *Euclidean Distance* dengan *Cosine Similarity* dalam proses *clustering. Spherical K-means* adalah algoritma *k-means* yang dirancang khusus untuk data berdimensi tinggi (Rustam & Fijri, 2020). Algoritma ini mengukur kemiripan berdasarkan *cosinus* sudut antar vektor, sehingga dua dokumen dengan orientasi vektor yang sama akan dianggap mirip meskipun memiliki panjang vektor yang berbeda.

Karena SPKM menggunakan *Cosine Similarity*, fungsi objektifnya bukanlah meminimalkan SSE, melainkan memaksimalkan total *cosine similarity* rata-rata. Semua vektor (dokumen dan *centroid*) dinormalisasi menjadi vektor satuan (panjangnya 1). Pada kondisi ini *Cosine Similarity* ekuivalen dengan

dot product ($x^T\ \mu$). Fungsi objektif SPKM dinyatakan dalam Persamaan (7) berikut:

$$L = \sum_{x} x^T\ \mu k(x) \tag{7}$$

dengan keterangan:

- L = nilai objektif total yang akan dimaksimalkan,
- x = vektor data (dokumen) tunggal,
- $\mu k(x)$ = vektor *centroid* dari klaster k yang memiliki *cosine similarity* tertinggi (jarak terdekat) dengan x.

Pada *k-means* standar, *Elbow Method* mencari titik "siku" di mana nilai SSE tidak lagi menurun secara signifikan. Namun, pada *spherical k-means*, metode ini mencari titik "siku" di mana nilai *L* (Total *Cosine Similarity*) tidak lagi meningkat secara signifikan saat *K* ditambah. Nilai *K* optimal dipilih berdasarkan titik di mana penambahan jumlah klaster baru tidak lagi memberikan peningkatan kemiripan total *L* yang berarti.

Proses *clustering* dimulai dengan alokasi setiap data ke *cluster* terdekat berdasarkan nilai *Cosine Similarity* tertinggi (sudut terkecil) dengan *centroid cluster* menggunakan Persamaan (5). Setelah seluruh data dialokasikan, posisi *centroid* diperbarui. Dalam SPKM *centroid* baru dihitung dengan menjumlahkan semua vektor data dalam klaster tersebut, kemudian menormalisasi hasil penjumlahan agar kembali menjadi vektor satuan (panjang 1), yang dinyatakan pada Persamaan (8) berikut:

$$\mu k = \frac{\sum_{x\ \epsilon\ X_k} x}{\left\|\sum_{x\ \epsilon\ X_k} x\right\|}\ , X_k = \{x_n | y_n = k\} \tag{8}$$

dengan keterangan:

- μk = posisi *centroid* baru untuk klaster k,
- X_k = himpunan semua vektor data x, yang di alokasikan ke klaster k.

- $\sum_{x \in X_k} x$ = penjumlahan vektor dari semua data di klaster k,
- $\left\|\sum_{x \in X_k} x\right\|$ = L_2 norm dari vektor hasil penjumlahan, yang digunakan untuk normalisasi.

Proses alokasi data dan pembaruan *centroid* dilakukan secara iteratif hingga *centroid* tidak berubah secara signifikan atau kondisi konvergen tercapai.

Dalam konteks penelitian ini, *k-means* digunakan untuk mengelompokkan dokumen akademik mahasiswa berdasarkan kemiripan fitur hasil kombinasi TF-IDF dan *Word2Vec*. Pendekatan *Spherical K-Means* ini memberikan dua keuntungan utama, yaitu metode ini selaras antara metode pengukuran kemiripan (*Cosine Similarity*) yang digunakan baik dalam tahap deteksi plagiarisme maupun dalam tahap *clustering,* sehingga hasil pengelompokan benar-benar mencerminkan kesamaan semantik antar dokumen. Keuntungan yang lainnya adalah metode ini dapat meningkatkan efisiensi pencarian kemiripan, karena sistem hanya perlu membandingkan dokumen dalam satu klaster yang relevan, bukan seluruh korpus. Dengan demikian, kombinasi antara *Cosine Similarity* dan *K-Means* bukan hanya mempercepat proses identifikasi, tetapi juga menjaga keakuratan deteksi dokumen yang memiliki kesamaan makna.

2.9.1 L2 Normalisasi

L2 Normalisasi atau yang dikenal juga sebagai *Euclidean Normalization* merupakan teknik normalisasi vektor yang bekerja dengan cara membagi setiap elemen vektor dengan panjang (*magnitude*) vektor tersebut, sehingga menghasilkan vektor baru yang memiliki panjang bernilai 1 atau disebut sebagai *unit vector*. Teknik ini banyak digunakan dalam pemrosesan teks dan *machine learning* terutama ketika perhitungan kemiripan berbasis *cosine similarity* diterapkan, karena memastikan bahwa perbandingan antar vektor dilakukan berdasarkan arah bukan besarnya nilai (Aytekin et al., 2018).

Mengacu pada Wang et al. (2017), secara matematis L2 Normalisasi dinyatakan pada Persamaan (9) sebagai berikut:

$$u = \frac{x}{||x||_2} = \frac{x}{\sqrt{\sum_{i=1}^{n} x_i^2}} \tag{9}$$

dengan keterangan:

- u = vektor hasil normalisasi
- x = vektor asal
- $||x||_2$ = L2 *norm* (panjang vektor), yaitu akar kuadrat dari jumlah kuadrat seluruh elemennya
- x_i = nilai elemen ke- i dari vektor x
- n = jumlah dimensi vektor

Hasil dari L2 Normalisasi memiliki sifat bahwa $||u||_2 = 1$, artinya seluruh vektor yang telah dinormalisasi berada pada permukaan bola satuan *unit sphere* di ruang berdimensi n. Wang *et al.* (2017) menjelaskan bahwa kondisi ini memiliki implikasi penting dalam perhitungan *cosine similarity*, dimana ketika dua vektor telah dinormalisasi L2, persamaan *cosine similarity* menjadi lebih sederhana karena penyebutnya bernilai 1, sehingga rumus *cosine similarity* menjadi seperti Persamaan (10) berikut:

$$\cos(\theta) = \frac{x \cdot y}{||x||_2 \times ||y||_2} = x \cdot y = \sum_{i=1}^{n} x_i \times y_i \tag{10}$$

Dalam penelitian ini, L2 Normalisasi diterapkan pada dua tahap yang berbeda dan dengan tujuan yang berbeda, yaitu L2 Normalisasi diterapkan secara terpisah pada masing-masing matriks TF-IDF dan FastText sebelum proses penggabungan (*feature concatenation*), dengan tujuan menyetarakan skala kedua representasi agar pembobotan menggunakan parameter alpha dan beta dapat bekerja secara proporsional tanpa salah satu representasi mendominasi yang

lain. Selanjutnya, L2 Normalisasi diterapkan kembali pada vektor gabungan hasil *feature concatenation*, dengan tujuan memastikan seluruh vektor dokumen berada pada *unit sphere* sehingga sesuai digunakan sebagai *input* pada proses *clustering* menggunakan pendekatan *Spherical K-Means* yang mensyaratkan perhitungan berbasis *cosine similarity.*

2.10 Metrik Evaluasi *Clustering*

Metrik evaluasi *clustering* digunakan untuk menilai kualitas hasil pengelompokan yang dihasilkan oleh algoritma *clustering* seperti K-Means. Karena pendekatan *unsupervised learning* tidak memiliki label kebenaran *(ground truth)* sebagai acuan, maka diperlukan metrik evaluasi internal yang mampu mengukur kohesi dan separasi antar *cluster* secara mandiri. Dalam penelitian ini metrik evaluasi yang digunakan adalah *Silhouette Score*.

2.10.1 *Silhouette Score*

Silhouette Score merupakan metrik evaluasi yang digunakan untuk menilai kualitas hasil *clustering* pada pendekatan *unsupervised learning* di mana label kebenaran tidak tersedia (Shutaywi & Kachouie, 2021). Metrik ini menggabungkan dua aspek sekaligus, yaitu kohesi *(intra-cluster compactness)* dan separasi *(inter-cluster separation)* dalam satu nilai tunggal, sehingga evaluasi *cluster* dapat memastikan bahwa *cluster* yang terbentuk tidak hanya terpisah dengan baik, tetapi juga homogen di dalamnya . Nilai *Silhouette Score* untuk setiap titik data *i* dihitung menggunakan Persamaan 11 berikut:

$$s(i) = \frac{b(i) - a(i)}{\max\{b(i), a(i)\}} \tag{11}$$

dengan keterangan: $a(i)$ adalah rata-rata jarak antara titik data *i* dengan seluruh titik lain dalam *cluster* yang sama *(within dissimilarity)*, sedangkan $b(i)$ adalah rata-rata jarak minimum antara

titik data *i* terhadap semua titik di *cluster* tetangga yang paling dekat *(between dissimilarity)* (Shutaywi & Kachouie, 2021).

Nilai $s(i)$ dapat berkisar antara -1 hingga 1. Nilai negatif tidak diinginkan karena menunjukkan bahwa $a(i)$ lebih besar dari $b(i)$, yang berarti rata-rata jarak dalam *cluster* lebih besar dari jarak antar *cluster*. Nilai positif diperoleh ketika $a(i) < b(i)$, dan *Silhouette Score* mencapai nilai maksimum $s(i) = 1$ ketika $a(i) = 0$. Semakin besar nilai $s(i)$ yang positif, semakin tinggi kemungkinan titik data tersebut berada di *cluster* yang tepat, sebaliknya, elemen dengan $s(i)$ negatif lebih mungkin ditempatkan pada *cluster* yang salah (Shutaywi & Kachouie, 2021).

Rata-rata nilai $s(i)$ dari seluruh titik data dalam setiap *cluster* disebut *average Silhouette width*, dan rata-rata dari seluruh *cluster* merupakan *Silhouette Score* keseluruhan model. Nilai yang lebih tinggi mencerminkan hasil *clustering* yang lebih baik (Yulisasih et al., 2024).

2.11 *Python*

Python adalah bahasa pemrograman tingkat tinggi yang mendukung pemrograman berorientasi objek dan dapat dijalankan di berbagai sistem operasi karena bersifat *multi-platform*. Dibandingkan dengan banyak bahasa lainnya, *Python* lebih mudah dipahami berkat sintaksnya yang menyerupai bahasa Inggris. Bahasa ini sering digunakan untuk membangun berbagai aplikasi maupun membuat script otomatisasi. *Python* juga bersifat open source, sehingga kode sumbernya tersedia untuk umum dan memungkinkan siapa saja ikut berkontribusi dalam pengembangannya (Londjo, 2021).

Python menekankan pada aspek keterbacaan kode, sehingga memungkinkan penulisan program dengan jumlah baris yang lebih ringkas dibandingkan dengan bahasa pemrograman lain. Struktur sintaksnya yang sederhana membuat *Python* mudah dipahami dan digunakan, bahkan oleh pemula (Bota & Setiyawati, 2022).

2.12 *Google Colab*

Google Colab adalah layanan *cloud* berbasis *Jupyter Notebook* yang disediakan oleh Google, yang memungkinkan pengguna untuk menulis dan menjalankan kode program *Python* secara interaktif melalui *browser web.* Menurut penelitian Andarsyah & Yanuar (2024), *colab* menyediakan berbagai fitur yang mendukung penulisan dan eksekusi kode Python langsung melalui browser tanpa instalasi tambahan. Layanan ini menawarkan akses GPU untuk kebutuhan komputasi yang tinggi, seperti pelatihan model *machine learning.* Pengguna juga bisa terhubung dengan *Google Drive* untuk menyimpan dan mengelola file. Lingkungan *notebook* yang interaktif memudahkan penggabungan kode, teks, gambar, dan output dalam satu dokumen. *Colab* sudah menyertakan banyak pustaka populer seperti *NumPy, pandas, TensorFlow*, dan *PyTorch,* sehingga mudah digunakan bahkan oleh pemula. Selain itu, *notebook* dapat dibagikan kepada orang lain sehingga memungkinkan kolaborasi secara langsung.

2.13 *Framework Flask*

Flask merupakan *web framework* berbasis bahasa pemrograman *Python* yang tergolong dalam kategori *microframework*. *Framework* ini berperan sebagai kerangka kerja untuk membangun logika aplikasi dan antarmuka *web*. *Framework Flask* digunakan untuk mempercepat proses pengembangan karena sudah menyediakan berbagai *library* dan kumpulan kode yang bisa langsung dimanfaatkan. Dengan begitu, pembuatan *webapp* dapat dilakukan tanpa perlu membangun seluruh komponen dari nol (Jonathan & Setiyawati, 2023).

Flask disebut *microframework* karena tidak mewajibkan penggunaan alat atau pustaka tertentu. Sebagian besar fungsi tambahan seperti validasi formulir, pengelolaan basis data, dan komponen pendukung lainnya tidak disertakan secara bawaan, melainkan dapat diintegrasikan melalui pustaka pihak ketiga sesuai kebutuhan pengembangan (Azhari, 2022).

2.14 *Unified Modeling Language* (UML)

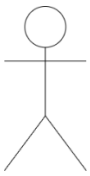
Unified Modeling Language (UML) adalah bahasa pemodelan standar yang dipakai untuk menggambarkan, mendefinisikan, serta merancang sistem perangkat lunak. UML menyediakan notasi dan diagram yang seragam, sehingga memudahkan para pengembang untuk berkomunikasi dalam proses analisis maupun perancangan sistem (Fitrani & Puspitaningrum, 2023). Dalam penerapannya, UML terdiri dari berbagai jenis diagram, seperti *Use Case Diagram, Activity Diagram,* dan *Class Diagram,* yang masing-masing berperan dalam memodelkan kebutuhan, alur proses, serta struktur data dari sistem yang akan dibangun

2.13.1. *Use Case Diagram*

Use Case Diagram adalah representasi visual dari fungsi-fungsi yang diharapkan tersedia dalam suatu sistem, serta menunjukkan bagaimana aktor berinteraksi dengan sistem tersebut. Dalam diagram ini, aktor digambarkan sebagai entitas, baik manusia maupun sistem lain yang berperan menjalankan atau memicu suatu aktivitas di dalam sistem (Ramdany et al., 2024). Singkatnya, *use case diagram* adalah perancangan sistem yang bertujuan untuk mendeskripsikan kebutuhan sistem. *Use Case Diagram* menjadi elemen penting dalam tahap perancangan perangkat lunak untuk mengurangi ambiguitas serta memperjelas komunikasi antara para pemangku kepentingan, seperti pengembang, pemilik produk, dan pengguna (Molla et al., 2024).

Use Case Diagram terdiri dari beberapa simbol yang memiliki fungsi tertentu dalam menggambarkan hubungan antara aktor dan sistem. Untuk menghindari kesalahan interpretasi, Tabel 2 berikut memberikan deskripsi singkat dari setiap simbol yang digunakan.

Tabel 2. Simbol – simbol *Use Case Diagram* (Ramdany et al., 2024).

Simbol	Nama	Keterangan
	*Use Case*	*Use case* menggambarkan fungsionalitas yang disediakan oleh sistem dalam bentuk unit-unit yang saling berinteraksi, baik antar unit maupun dengan aktor. Nama *use case* umumnya diawali dengan kata kerja karena merepresentasikan tindakan atau layanan yang dilakukan oleh sistem.
	*Actor*	Aktor merupakan pihak yang berinteraksi dengan sistem informasi, baik berupa manusia, proses, maupun sistem lain yang berada di luar sistem tersebut. Nama aktor umumnya diawali kata benda, dan aktor sendiri merepresentasikan peran yang dijalankan oleh pengguna atau entitas lain saat berinteraksi dengan sistem.
	Association	Relasi ini menunjukkan adanya komunikasi atau interaksi antara aktor dengan *use case* yang dilibatkan, menandakan bahwa aktor tersebut berpartisipasi dalam menjalankan fungsi yang ada pada use case.
	Extend	Relasi *extend* merupakan hubungan di mana sebuah *use case* tambahan memperluas perilaku *use case* utama. *Use case* tambahan ini tetap dapat berdiri sendiri meskipun tanpa *use case* yang diperluas, dan konsepnya menyerupai mekanisme *inheritance* pada pemrograman berorientasi objek.

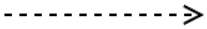
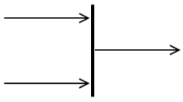
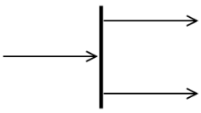
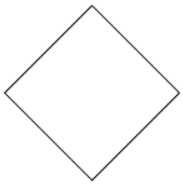
	Generalitation	Relasi ini menggambarkan hubungan generalisasi–spesialisasi antara dua *use case*, di mana salah satu *use case* memiliki fungsi yang lebih umum, sementara *use case* lainnya merupakan versi yang lebih khusus dari fungsi tersebut.
	*Include*	Relasi *include* menunjukkan bahwa sebuah *use case* tambahan akan selalu memanggil *use case* lain yang disertakan di dalamnya setiap kali dijalankan. Dengan kata lain, *use case* utama wajib mengeksekusi *use case* yang di-*include* terlebih dahulu karena fungsinya menjadi bagian yang tidak terpisahkan dari alur *use case* tersebut.

2.13.2. *Activity Diagram*

Activity Diagram merupakan salah satu diagram perilaku *(behavioral diagram)* dalam UML yang digunakan untuk memvisualisasikan alur aktivitas dalam sebuah proses. Diagram ini berfungsi sebagai representasi grafis dari skenario *use case* dengan menampilkan urutan peristiwa *(event flow)* yang terjadi di dalam sistem. Melalui diagram ini, proses kerja dapat digambarkan secara lebih jelas sehingga memudahkan pengembang maupun pemangku kepentingan dalam memahami bagaimana suatu aktivitas dimulai, dijalankan, dan berakhir di dalam sistem (Hnatkowska & Mateusz, 2021).

Komponen utama dalam *Activity Diagram* terdiri dari beberapa jenis notasi yang masing-masing menggambarkan elemen penting dalam alur proses. Tabel 3 berikut ini menunjukkan simbol – simbol yang terdapat pada *activity diagram.*

Tabel 3. Simbol – simbol *Activity Diagram* (Ramdany et al., 2024).

Simbol	Nama	Keterangan
	*Activity*	Menyatakan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain.
	Control Flow	Menunjukkan Urutan Eksekusi.
	*Object Flow*	Menunjukkan aliran objek dari sebuah *action* atau *activity* ke *action.*
	*Start Point*	Menyatakan bahwa sebuah objek dibentuk atau diawali.
	*End Point*	Menyatakan bahwa sebuah objek dibentuk atau diakhiri.
	*Join/ Penggabungan*	Menyatakan untuk menggabungkan kembali *activity* atau *action* yang *parallel.*
	*Fork*	Menyatakan untuk memecah *behavior* menjadi *activity* atau *action* yang *parallel.*
	*Decision*	Menunjukkan penggambaran suatu keputusan/tindakan yang harus di ambil pada kondisi tertentu.

2.15 *Software Development Life Cycle* (SDLC)

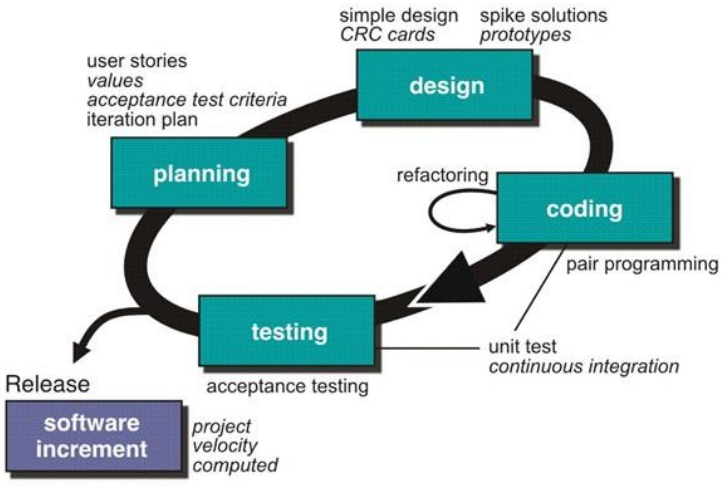
Software Development Life Cycle (SDLC) merupakan suatu metode yang digunakan untuk menganalisis, merancang, dan mengembangkan sebuah sistem agar sistem tersebut dapat memenuhi kebutuhan yang telah ditentukan (Samad et al., 2024). SDLC ini menyediakan serangkaian tahapan terstruktur yang membantu pengembang memahami kebutuhan pengguna, merancang solusi yang tepat, serta memastikan sistem berjalan sesuai tujuan. Setiap tahap dalam SDLC yang ditunjukkan pada Gambar 3, mulai dari *planning, analysis, design, development, testing, implementation* hingga *maintenance* berperan penting dalam menjaga kualitas sistem yang dibangun.

Gambar 3. Tahapan Metode Pengembangan SDLC (Nilma, 2022).

SDLC menawarkan berbagai model yang merepresentasikan pendekatan berbeda dalam pengembangan sistem, seperti model *Waterfall* yang bersifat berurutan dan model iteratif seperti *Prototyping.* Selain itu, terdapat pula model *Agile,* salah satunya *Extreme Programming* (XP), yang menekankan pengembangan secara iteratif, kolaborasi tim, serta kemampuan beradaptasi terhadap perubahan kebutuhan pengguna. Setiap model memiliki karakteristik, kelebihan, dan keterbatasan masing-masing, sehingga pemilihannya perlu disesuaikan dengan kebutuhan, skala, dan kompleksitas proyek yang dikembangkan.

2.15.1 *Extreme Programming*

Extreme Programming (XP) merupakan model pengembangan sistem yang menekankan kerja sama tim, komunikasi yang intensif, serta kemampuan beradaptasi terhadap perubahan kebutuhan pengguna. Menurut Supriyatna (2018) XP adalah metode rekayasa perangkat lunak yang berorientasi objek dan paling efektif diterapkan pada tim berskala kecil hingga menengah. Metode ini sangat sesuai untuk proyek dengan kebutuhan pengguna yang belum sepenuhnya jelas atau cenderung berubah selama proses pengembangan. Melalui kolaborasi aktif antara pengembang dan pemangku kepentingan, XP memungkinkan perubahan dilakukan secara berulang sehingga perangkat lunak dapat terus disesuaikan dengan kebutuhan yang berkembang (Illahi et al., 2023).



Gambar 4. Metode *Extreme Programming (XP)* (Fachrurozi et al., 2025).

Gambar 4 menunjukkan kerangka atau tahapan dalam pelaksanaan pengembangan yang dilakukan pada sebuah perangkat lunak. Penjelasan mengenai setiap tahap berdasarkan Rahman et al. (2024) disajikan sebagai berikut.

1) *Planning* (Perencanaan)

   Tahap *Planning* berfokus pada identifikasi kebutuhan sistem melalui diskusi dengan pengguna. Tahapan perencanaan

dilakukan dengan terlebih dahulu menganalisis konteks bisnis aplikasi yang akan dikembangkan, meliputi penetapan *output* sistem, fitur, fungsi aplikasi, serta alur proses pengembangannya. Pada tahap ini dirumuskan kebutuhan dan fungsionalitas sistem secara menyeluruh sebagai dasar dalam pengembangan sistem

2) *Design* (Perancangan)

Pada tahap perancangan, dilakukan penyusunan pemodelan sistem berdasarkan hasil analisis kebutuhan yang telah diperoleh. Proses ini mencakup perancangan arsitektur sistem secara menyeluruh, termasuk pemetaan kebutuhan antara komponen perangkat keras dan perangkat lunak, serta pemodelan basis data untuk menggambarkan hubungan antar data. Selain itu, desain perangkat lunak dilakukan dengan mengidentifikasi abstraksi sistem, pembagian modul, struktur data, dan keterkaitan antar komponen yang akan digunakan pada tahap *coding*. Perancangan yang tersusun dengan baik diharapkan dapat mempermudah proses pengkodean dan mendukung pengembangan sistem secara terstruktur.

3) *Coding* (Pengkodean)

Tahap pengkodean merupakan fase implementasi di mana pengembang mulai menuliskan kode program berdasarkan rancangan sistem dan kebutuhan *user* yang telah ditetapkan. Pada tahap ini, desain yang telah dibuat diwujudkan ke dalam perangkat lunak yang dapat dijalankan, sehingga aplikasi yang dikembangkan mampu berfungsi sebagai solusi atas permasalahan yang ada.

4) *Testing* (Pengujian)

Tahap pengujian merupakan tahap akhir yang dilakukan untuk memastikan seluruh layanan, fitur, dan fungsionalitas pada aplikasi berjalan sesuai dengan yang diharapkan. Melalui proses

ini, dapat diperoleh kesimpulan mengenai kinerja dan kelayakan sistem yang telah dibangun.

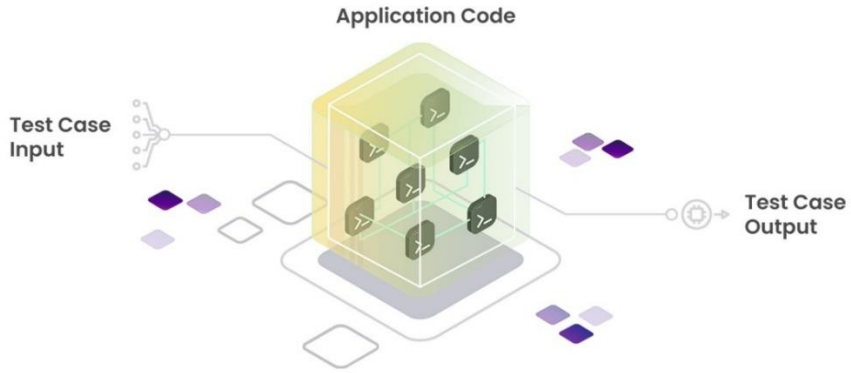
Metode *Extreme Programming* (XP) digunakan dalam penelitian ini karena karakteristik pengembangannya yang fleksibel dan adaptif terhadap perubahan kebutuhan sistem. Penelitian ini melibatkan dua skenario analisis, yaitu model deteksi kemiripan dokumen teks dan model deteksi kemiripan *source code*, yang masing-masing memerlukan proses eksperimen, penyesuaian parameter, serta evaluasi hasil secara berulang. Pendekatan XP memungkinkan pengembangan kedua model dilakukan secara iteratif melalui siklus perencanaan, pengkodean, dan pengujian yang berkesinambungan, sehingga perbaikan pada tahapan *preprocessing*, representasi fitur, maupun metode pengukuran kemiripan dapat segera diterapkan dan divalidasi.

Selain itu, penerapan praktik pengujian berkelanjutan dalam XP mendukung pengujian fungsionalitas model secara konsisten pada setiap iterasi, baik untuk pemrosesan NLP pada dokumen teks maupun analisis *source code.* Sistem web yang berfungsi sebagai antarmuka eksekusi model juga sejalan dengan prinsip XP yang menekankan desain sederhana dan fokus pada fungsi inti. Dengan demikian, *Extreme Programming* menjadi pendekatan yang tepat untuk memastikan pengembangan sistem berjalan responsif, terkontrol, dan sesuai dengan kebutuhan penelitian yang bersifat eksploratif dan dinamis.

2.16 *White Box Testing*

White Box Testing merupakan metode pengujian perangkat lunak yang memanfaatkan struktur kontrol di dalam kode sebagai dasar dalam merancang *test case* pada tahap desain komponen (Londjo, 2021). Metode ini berfokus pada pemeriksaan langsung terhadap kode program guna mengidentifikasi kesalahan atau kekurangan yang mungkin terjadi. Pengujian dilakukan

dengan menganalisis setiap modul secara mendalam untuk memastikan bahwa seluruh bagian aplikasi berfungsi sesuai yang diharapkan (Helmi & Nuryasin, 2024).

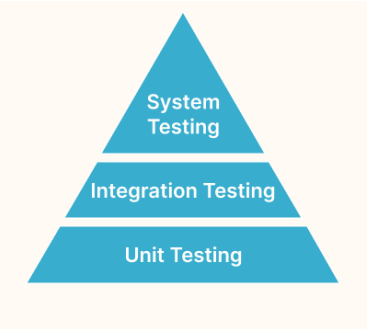


Gambar 5. Skema *White Box Testing* (Hidayat et al., 2025).

Gambar 5 menunjukkan alur dasar *White Box Testing* dalam memeriksa logika dan struktur internal program. Skema tersebut menggambarkan bagaimana *test case* yang telah dirancang dimasukkan ke dalam kode aplikasi, kemudian diproses sesuai alur kontrol dan struktur pemrograman yang ada. Proses ini memungkinkan penguji menelusuri jalur eksekusi secara mendalam untuk memastikan setiap bagian kode berjalan sebagaimana mestinya. *Output* yang dihasilkan dari eksekusi *test case* kemudian dianalisis untuk menilai apakah perilaku program sudah sesuai dengan yang diharapkan.

White Box Testing memiliki sejumlah teknik pengujian, seperti *Loop Testing* yang digunakan untuk memvalidasi struktur perulangan dalam program, Data *Flow Testing* yang menelusuri aliran data untuk memastikan nilai diproses dengan benar, serta *Control Flow Testing* yang menjadikan alur kontrol program sebagai dasar pembentukan *test case.* Selain itu, terdapat *Branch Testing* yang memfokuskan pengujian pada setiap percabangan logika untuk memastikan semua kondisi dapat dijalankan. Teknik lainnya adalah Basis *Path Testing,* yang mengidentifikasi jalur eksekusi independen dalam program untuk memastikan seluruh jalur logis telah diuji. Melalui berbagai teknik tersebut, *White Box Testing* membantu pengembang memastikan

bahwa struktur internal perangkat lunak benar-benar bekerja sesuai rancangan dan bebas dari kesalahan tersembunyi (Pratiwi & Widianti, 2025).



Gambar 6. Sistem Pengujian Mutu Perangkat Lunak (Hidayat et al., 2025).

Gambar 6 menunjukkan level pengujian perangkat lunak. Menurut Hidayat et al. (2025) Dalam konteks siklus pengembangan perangkat lunak, *white box testing* dapat diterapkan pada beberapa level pengujian, khususnya *unit testing*, *integration testing*, dan *system testing*. Ketiga tahap ini dijelaskan sebagai berikut.

1) *Unit Testing*

   Unit Testing merupakan proses pengujian perangkat lunak yang dilakukan pada unit atau komponen terkecil dari program secara individual. Unit Testing bertujuan untuk memverifikasi bahwa setiap fungsi, prosedur, atau modul dasar dalam sistem dapat berjalan sesuai dengan spesifikasi. Pengujian dilakukan langsung pada kode sumber (*white box*) sehingga penguji dapat melihat struktur internal, logika, serta aliran kontrol program

2) *Integration Testing*

   Integration Testing bertujuan memastikan bahwa modul atau komponen dalam sistem dapat berinteraksi dengan benar satu sama lain. Tahap ini penting untuk menemukan kesalahan yang muncul akibat penggabungan antar modul yang sebelumnya telah diuji secara terpisah pada unit testing. Perbedaan utama antara unit *test* dan *integration test* terletak pada objek yang diuji: unit *test* berfokus pada bagian kecil dan spesifik dari sistem, sedangkan *integration test* menguji beberapa komponen

sekaligus untuk memastikan bahwa seluruh komponen yang digabungkan mampu berfungsi dengan baik.

3) *Sytem Testing*

System Testing merupakan proses evaluasi menyeluruh terhadap perangkat lunak untuk memastikan bahwa seluruh fitur berfungsi sesuai dengan spesifikasi yang telah ditetapkan. Pada tahap ini, dilakukan dua jenis pengujian utama, yakni pengujian fungsional yang bertujuan memastikan setiap fitur bekerja sesuai kebutuhan pengguna, serta pengujian non-fungsional yang mencakup aspek performa, keamanan, dan kompatibilitas sistem. Pengujian ini penting untuk memastikan bahwa perangkat lunak tidak hanya berjalan dengan benar, tetapi juga memenuhi standar kualitas yang diperlukan dalam lingkungan operasional.

III. METODOLOGI PENELITIAN

3.1 Tempat dan Waktu Penelitian

3.1.1 Tempat Penelitian

Penelitian ini dilaksanakan di Jurusan Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Lampung, yang berlokasi di Jalan Prof. Dr. Ir. Sumantri Brojonegoro No. 1, Gedong Meneng, Kecamatan Rajabasa, Kota Bandar Lampung.

3.1.2 Waktu Penelitian

Penelitian ini dimulai pada bulan September dan diperkirakan selesai sampai dengan bulan Januari 2026 yang ditunjukkan pada Tabel 4.

Tabel 4. Jadwal Pelaksanaan Penelitian.

Kegiatan Penelitian	2025																2026					
	Okt				Nov				Des				Jan				Feb				Mar	
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2
Studi Literatur																						
Analisis Kebutuhan																						
Desain																						
Penyusunan *Draft* Laporan BAB I-III																						
Implementasi *(Preprocessing)*																						
Implementasi (Pembuatan Model)																						
Implementasi (Evaluasi Model)																						
Implementasi *Web* (*Flask*)																						
Testing																						
Penyusunan *Draft* Laporan (BAB IV-V)																						

3.2 Alat Pendukung

3.2.1 Perangkat Keras (*Hardware*)

Perangkat keras atau *hardware* yang digunakan dalam penelitian ini adalah laptop yang menjadi perangkat utama untuk pengolahan data, pengembangan model dan sistem *web*, dengan spesifikasi sebagai berikut:

- Perangkat: Acer Aspire A314-22
- Prosesor: AMD Ryzen 3 3250U with Radeon Graphics (4 CPUs)
- RAM: 4 GB
- Penyimpanan: 256 GB
- GPU: AMD Radeon (TM) Graphics

3.2.2 Perangkat Lunak *(Software)*

Dalam penelitian ini digunakan sejumlah perangkat lunak *(software)* yang berperan penting dalam mendukung proses pengolahan data, pengembangan model, serta pembuatan sistem berbasis *web*. Adapun daftar perangkat lunak yang digunakan selama penelitian adalah sebagai berikut:

a. Sistem Operasi

   Perangkat laptop yang digunakan dalam penelitian ini menggunakan sistem operasi *Windows 11 Home Single Language 64-bit* dengan arsitektur *x64-based processor*. Sistem operasi ini memiliki stabilitas yang tinggi, dukungan keamanan berlapis, serta kompatibilitas yang baik terhadap berbagai aplikasi pengembangan perangkat lunak dan analisis data. *Windows 11* juga menyediakan lingkungan kerja yang optimal untuk menjalankan berbagai perangkat lunak pendukung seperti *Visual Studio Code, Python*, dan *Laragon* secara bersamaan tanpa gangguan performa. Dengan kestabilan sistem operasi ini, seluruh proses penelitian, mulai dari pra-pemrosesan data, pelatihan

model, hingga pengujian sistem *web* dapat dijalankan secara efisien dan terintegrasi.

b. Google Chrome

Google Chrome digunakan sebagai *browser* utama dalam penelitian ini untuk melakukan penelusuran literatur, pengujian sistem *web*, serta analisis tampilan antarmuka pengguna. Selain itu, *browser* ini juga berperan penting dalam mengakses berbagai lingkungan pengembangan berbasis *cloud* seperti Google Colab dan Google Drive, yang digunakan selama proses pengembangan dan pengujian sistem deteksi plagiarisme berbasis web. Google Chrome memiliki keunggulan dalam hal kecepatan, keamanan, serta kompatibilitas terhadap teknologi web modern, sehingga memastikan sistem yang dikembangkan dapat dijalankan secara optimal tanpa kendala kompatibilitas maupun performa.

c. Python

Python berfungsi sebagai bahasa pemrograman utama yang digunakan dalam penelitian ini untuk pengolahan teks, pelatihan model, dan pengembangan sistem *web.* Versi *Python* yang digunakan adalah *Python* 3.10 keatas, karena versi ini memiliki stabilitas tinggi serta kompatibel dengan berbagai pustaka pendukung seperti *Scikit-learn*, *Gensim*, *Flask*, dan *NLTK*. Pustaka-pustaka tersebut digunakan untuk proses *text preprocessing*, pembobotan kata menggunakan TF-IDF, pembentukan vektor *Word2Vec,* perhitungan kesamaan dengan *Cosine Similarity*, serta penerapan algoritma *K-Means Clustering*. Selain itu, *Python* juga mendukung integrasi dengan *Flask* untuk membangun sistem *web* yang interaktif dan efisien.

d. Google Colab

Dalam penelitian ini, *Google Colab* digunakan sebagai *platform* utama untuk menulis dan menjalankan kode *Python* yang berkaitan

dengan proses pengolahan teks dan pelatihan model deteksi kemiripan dokumen. Colab juga menyediakan dukungan komputasi GPU secara gratis, sehingga mempercepat proses pengolahan data dan pelatihan model seperti *Word2Vec* serta penerapan algoritma *K-Means Clustering,* TF-IDF, dan *Cosine Similarity* dalam analisis kemiripan dokumen. Selain itu, Colab dilengkapi dengan berbagai pustaka *machine learning* dan *natural language processing* seperti *Scikit-learn, Gensim, NumPy, Pandas, dan Matplotlib* yang digunakan untuk pembobotan kata, analisis kemiripan dokumen, serta visualisasi hasil penelitian.

e. Google Drive

Pemilihan *Google Drive* pada penelitian ini didasarkan pada kemampuannya dalam menyediakan layanan penyimpanan data yang aman, fleksibel, serta dapat diakses secara *real-time* dari berbagai perangkat. Selain itu, *Google Drive* memiliki integrasi langsung dengan *Google Colab*, sehingga memudahkan proses pemuatan, pembaruan, dan pengelolaan model tanpa memerlukan instalasi tambahan.

Google Drive dimanfaatkan sebagai media penyimpanan data dan model selama proses pelatihan dan pengujian di lingkungan *Colab.* Selanjutnya, pada tahap implementasi sistem, diterapkan pendekatan *indirect cloud computing,* di mana proses komputasi dan analisis kemiripan dokumen dilakukan pada server *Flask,* sedangkan file model dan data disimpan secara terpusat di *Google Drive* melalui integrasi *Google Drive API*. Pendekatan ini dipilih untuk meningkatkan efisiensi sistem dari sisi performa dan biaya, sekaligus mempertahankan fleksibilitas, kemudahan akses, serta kemampuan pembaruan model secara berkelanjutan.

f. Visual Studio Code (VS Code)

Dalam penelitian ini, *VS Code* digunakan sebagai lingkungan pengembangan utama untuk membangun sistem *web* deteksi kemiripan dokumen berbasis *Flask*. Editor ini memiliki integrasi yang baik dengan *framework Flask*, sehingga peneliti dapat menjalankan web server langsung melalui terminal bawaan tanpa memerlukan konfigurasi tambahan. Selain itu, *Python extension* dan *Flask Snippets* di *VS Code* memudahkan dalam membuat *routing*, *debugging*, serta mengatur struktur proyek berbasis *Flask.*

g. *Framework Flask*

Dalam penelitian ini, *Flask* berperan sebagai penghubung utama antara model analisis yang telah dilatih (TF-IDF, *Word2Vec*, dan *K-Means*) dengan antarmuka web yang digunakan oleh dosen. *Framework* ini menangani proses utama sistem, mulai dari menerima *request* pengguna, memproses dokumen yang diunggah, menjalankan perhitungan tingkat kemiripan, hingga menampilkan hasil analisis dalam bentuk persentase kemiripan pada halaman *web.* Seluruh proses komputasi dijalankan di sisi *backend Flask,* sedangkan hasilnya ditampilkan secara dinamis melalui *frontend* berbasis HTML dan CSS. Dengan demikian, *Flask* berfungsi sebagai jembatan antara logika pemrosesan data dan tampilan antarmuka pengguna, sehingga memungkinkan sistem berjalan secara interaktif dan efisien.

3.3 Studi Literatur

Tahap studi literatur merupakan langkah awal yang bertujuan untuk memperoleh pemahaman mendalam mengenai konsep, teori, dan metode yang relevan dengan penelitian. Pada tahap ini dilakukan penelusuran terhadap berbagai sumber ilmiah seperti artikel jurnal nasional maupun internasional, serta publikasi akademik daring yang membahas topik terkait deteksi plagiarisme dan pengukuran kemiripan dokumen. Kajian difokuskan

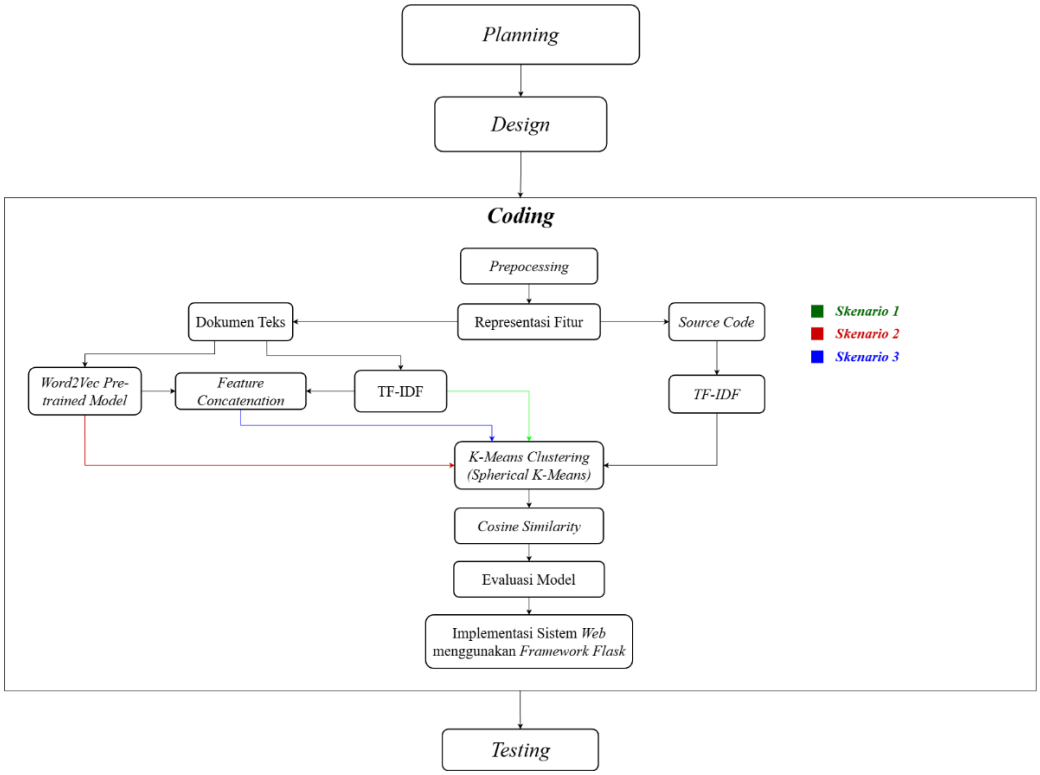
pada teori-teori dasar mengenai representasi teks menggunakan metode *Term Frequency–Inverse Document Frequency (TF-IDF)* dan *Word2Vec,* serta teknik pengukuran kemiripan menggunakan algoritma *Cosine Similarity.* Selain itu, ditelaah pula konsep *K-Means Clustering* yang berfungsi dalam proses pengelompokan dokumen berdasarkan kemiripan fitur untuk meningkatkan efisiensi analisis. Hasil dari tahap ini menjadi landasan teoritis dalam perancangan sistem deteksi kemiripan dokumen berbasis kombinasi TF-IDF, *Word2Vec, Cosine Similarity*, dan *K-Means Clustering.*

Pada skenario analisis *source code,* studi literatur diarahkan pada teknik-teknik *preprocessing* khusus kode program. Kajian teori juga diarahkan pada penggunaan metode representasi berbasis token dengan TF-IDF, yang mampu menangkap pola frekuensi sintaksis dan struktur permukaan kode secara efektif. Selain itu, penelitian ini juga mengkaji literatur terkait pendekatan pengukuran kemiripan struktur kode menggunakan *Cosine Similarity.* Metode ini bekerja dengan membandingkan arah vektor TF-IDF dari masing-masing dokumen *source code,* sehingga kesamaan pola penulisan maupun struktur logika di antara program dapat teridentifikasi secara lebih akurat.

Hasil studi literatur ini menjadi dasar teoritis dalam merancang dua skenario model pendeteksian kemiripan yang berbeda yaitu dokumen teks dan *source code* serta menjadi pijakan dalam mengintegrasikan kedua model tersebut ke dalam sebuah antarmuka *web* sederhana berbasis *Flask* yang berfungsi sebagai media eksekusi model tanpa mekanisme autentikasi.

3.4 Tahapan Penelitian

Penelitian ini terdiri atas beberapa tahapan utama yang dilakukan secara berurutan untuk memastikan hasil analisis yang akurat dan sesuai dengan tujuan penelitian. Gambaran lengkap mengenai alur tahapan penelitian dapat dilihat pada Gambar 7 berikut.



Gambar 7. Tahapan Penelitian.

Berdasarkan Gambar 7, penelitian ini mengikuti alur *Extreme Programming* yang terdiri atas tahap *Planning*, *Design*, *Coding*, dan *Testing*. Tahap *Coding* merupakan tahap inti yang terbagi menjadi dua *pipeline* analisis. *Pipeline* dokumen teks menerapkan tiga skenario representasi fitur yaitu TF-IDF saja (Skenario 1), *Word2Vec Pre-trained Model* saja (Skenario 2), dan *Feature Concatenation* yang menggabungkan keduanya (Skenario 3). *Pipeline source code* menerapkan ekstraksi fitur berbasis TF-IDF dengan pendekatan *dual scoring* yang menggabungkan TF-IDF *Cosine Similarity* dan *Line-based Cosine Similarity* dalam perhitungan kemiripannya. Kedua *pipeline* melalui tahap *K-Means Clustering* dan *Cosine Similarity,* kemudian dievaluasi sebelum diintegrasikan ke dalam sistem web berbasis Flask. Tahap *Testing* dilakukan menggunakan metode *White Box Testing* untuk memverifikasi kebenaran logika internal sistem secara menyeluruh.

3.5 *Planning*

3.5.1 Kebutuhan Fungsional

Identifikasi kebutuhan sistem dilakukan untuk menentukan fungsi-fungsi apa saja yang harus disediakan oleh aplikasi agar dapat bekerja sesuai tujuan penelitian. Kebutuhan fungsional yang ditunjukkan pada Tabel 5 berisi uraian mengenai layanan dan perilaku sistem yang terlihat secara langsung oleh pengguna, serta menggambarkan hubungan antara aktor dengan fitur yang disediakan sistem.

Tabel 5. Kebutuhan Fungsional

No	ID	Kebutuhan Fungsional
1	KF-001	Sistem harus dapat menganalisis kemiripan dokumen tugas teks secara otomatis antar dokumen yang diunggah dalam satu proses dan menampilkan hasil kemiripannya
2	KF-002	Sistem harus menyediakan fitur untuk mengunggah dokumen tugas teks secara *batch* dalam format txt, pdf, dan docx
3	KF-003	Sistem harus dapat menganalisis kemiripan file *source code* secara otomatis antar file yang diunggah dalam satu proses dan menampilkan hasil kemiripannya
4	KF-004	Sistem harus menyediakan fitur untuk mengunggah file *source code* secara *batch* dengan format .py
5	KF-005	Sistem harus menyediakan fitur untuk mengunduh hasil analisis kemiripan.

3.5.2 Kebutuhan Non-Fungsional

Selain kebutuhan fungsional, sistem juga memiliki kebutuhan non-fungsional yang ditunjukkan pada Tabel 6. Kebutuhan non-fungsional mendeskripsikan karakteristik kualitas yang harus dipenuhi agar aplikasi dapat berjalan secara optimal. Kebutuhan non-fungsional tidak berkaitan langsung dengan fitur yang digunakan pengguna, tetapi mencakup aspek seperti performa, keamanan, keandalan, kompatibilitas, skalabilitas, dan kemudahan penggunaan. Aspek-aspek ini memastikan bahwa sistem tidak hanya mampu melakukan

analisis kemiripan, tetapi juga memberikan pengalaman penggunaan yang nyaman dan aman.

Tabel 6. Kebutuhan Non-Fungsional.

No	ID	Kebutuhan Non-Fungsional
1	KNF-001	Antarmuka sistem harus sederhana, sehingga pengguna cukup memilih jenis analisis, mengunggah file, dan menekan tombol analisis tanpa perlu navigasi yang kompleks.
2	KNF-002	Sistem harus menampilkan pesan kesalahan jika format file tidak sesuai atau proses analisis gagal harus ditampilkan dengan bahasa yang jelas dan mudah dipahami.
3	KNF-003	Sistem harus mampu menyelesaikan proses analisis kemiripan dokumen atau *source code* dalam waktu yang wajar untuk ukuran dataset yang digunakan dalam penelitian
4	KNF-004	Sistem harus dapat diakses melalui *browser* modern seperti *Google Chrome* pada sistem operasi umum (misalnya *Windows*) tanpa memerlukan instalasi perangkat lunak tambahan di sisi pengguna.

3.5.3 Pengumpulan Data

Tahap pengumpulan data pada penelitian ini dilakukan dengan mengumpulkan dua jenis dataset yang dibedakan berdasarkan karakteristik tugas mahasiswa, yaitu dataset dokumen teks dan dataset *source code.* Pemisahan jenis data ini diperlukan karena penelitian ini tidak hanya melakukan deteksi kemiripan pada dokumen akademik, tetapi juga memperluas analisis pada tugas pemrograman yang umum diberikan di Jurusan Ilmu Komputer Universitas Lampung.

Untuk dataset dokumen teks, data diperoleh dari mata kuliah berbasis teori seperti *Metodologi Penelitian.* Mata kuliah tersebut dipilih karena bentuk tugasnya berupa esai atau laporan naratif yang memiliki potensi kemiripan isi antar mahasiswa, sehingga relevan untuk proses pelatihan model TF-IDF, *Word2Vec*, dan *K-Means.* Dokumen yang dikumpulkan berformat *.pdf* dan *.docx*, kemudian diekstraksi menjadi teks biasa sebagai input awal untuk *tahap preprocessing*. Dataset ini

bersifat non-komersial dan hanya digunakan untuk keperluan penelitian akademik dengan tetap menjaga kerahasiaan identitas mahasiswa.

Sementara itu, untuk dataset *source code*, data diperoleh dari mata kuliah rumpun pemrograman yaitu Struktur Data Algoritma (SDA). Tugas-tugas dari mata kuliah ini umumnya memiliki struktur logika yang serupa antar mahasiswa, sehingga berpotensi menimbulkan plagiarisme dalam bentuk penyalinan kode. File yang dikumpulkan berformat *.py* kemudian diolah melalui proses ekstraksi kode, pembersihan komentar, normalisasi *whitespace,* serta tokenisasi spesifik bahasa pemrograman. Dataset ini digunakan untuk membangun model deteksi kemiripan *source code* yang berbeda alur pemrosesannya dari dokumen teks.

Seluruh dataset baik dokumen teks maupun *source code* disimpan secara terpusat di *Google Drive* untuk mempermudah proses pemuatan data di *Google Colab* selama tahap pengembangan dan pelatihan model. Pendekatan terpusat ini juga memudahkan integrasi dengan sistem *web* berbasis *Flask* pada tahap implementasi, karena file data dan model dapat diakses dengan aman dan konsisten tanpa perlu penyimpanan lokal tambahan lebih lanjut.

3.6 *Design*

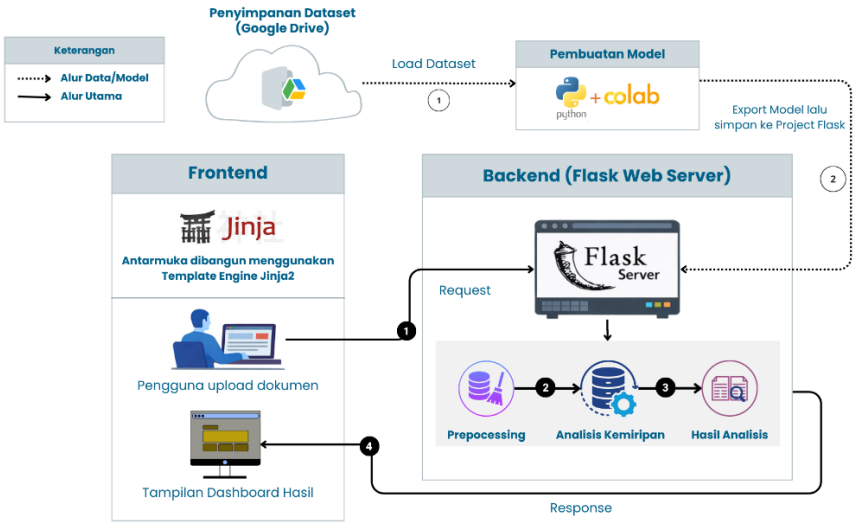
Tahap *system design* berfungsi untuk menerjemahkan kebutuhan fungsional dan non-fungsional menjadi rancangan teknis sistem yang lebih rinci. Pada tahap ini ditentukan bagaimana sistem pendeteksi kemiripan dokumen dan *source code* akan dibangun, mulai dari arsitektur perangkat lunak, alur proses, pemodelan fungsional melalui *Use Case Diagram*, perancangan basis data, hingga rancangan antarmuka pengguna. Hasil dari tahap ini menjadi acuan langsung pada tahap implementasi, sehingga perancangan dilakukan secara sistematis dan konsisten dengan kebutuhan yang telah didefinisikan sebelumnya.

3.6.1 Arsitektur Sistem

Arsitektur sistem pada penelitian ini menerapkan arsitektur aplikasi web berbasis *client – server* yang terdiri dari komponen *frontend* dan *backend* serta didukung oleh proses pengembangan model yang dilakukan secara terpisah. Sistem dirancang untuk melakukan analisis kemiripan dokumen teks dan *source code*, di mana proses pembuatan model dilakukan menggunakan lingkungan komputasi eksternal, sedangkan proses analisis dokumen dijalankan melalui aplikasi web berbasis *Flask.*

Pada arsitektur ini, proses pembuatan model analisis kemiripan dilakukan menggunakan Google Colab dengan bahasa pemrograman Python, sedangkan aplikasi web bertugas menangani proses penerimaan dokumen, pemrosesan data, serta perhitungan kemiripan dokumen. Sistem tidak menggunakan *database* karena proses analisis dilakukan secara langsung terhadap file yang diunggah oleh pengguna dan hasil analisis dihasilkan secara dinamis saat proses berlangsung.

Hubungan antar komponen dalam sistem digambarkan pada Gambar 8 yang menunjukkan alur pengembangan model, interaksi pengguna melalui antarmuka web, serta proses analisis yang dijalankan oleh *backend* sistem.



Gambar 8. Arsitektur *System.*

Arsitektur secara umum terdiri dari tiga komponen utama:

1. Penyimpanan Dataset (Google Drive)

   Google Drive digunakan sebagai media penyimpanan dataset yang digunakan dalam proses pengembangan model analisis kemiripan. Dataset tersebut dimuat ke dalam lingkungan Google Colab pada saat proses pembuatan model dilakukan. Penyimpanan dataset pada *cloud* bertujuan untuk memudahkan proses akses dan pengelolaan data selama tahap eksperimen dan pengembangan model.

2. Pembuatan Model

   Pada tahap ini model analisis kemiripan dikembangkan menggunakan Google Colab dengan bahasa pemrograman Python. Lingkungan Google Colab dipilih karena menyediakan fasilitas komputasi yang mendukung proses pengolahan data dan eksperimen model secara interaktif. Dataset yang tersimpan pada Google Drive dimuat ke dalam Colab untuk digunakan dalam proses pengembangan model. Setelah model selesai dibangun, model tersebut diekspor dan disimpan ke dalam *project* aplikasi Flask sehingga dapat digunakan pada saat sistem melakukan analisis kemiripan dokumen.

3. *Frontend*

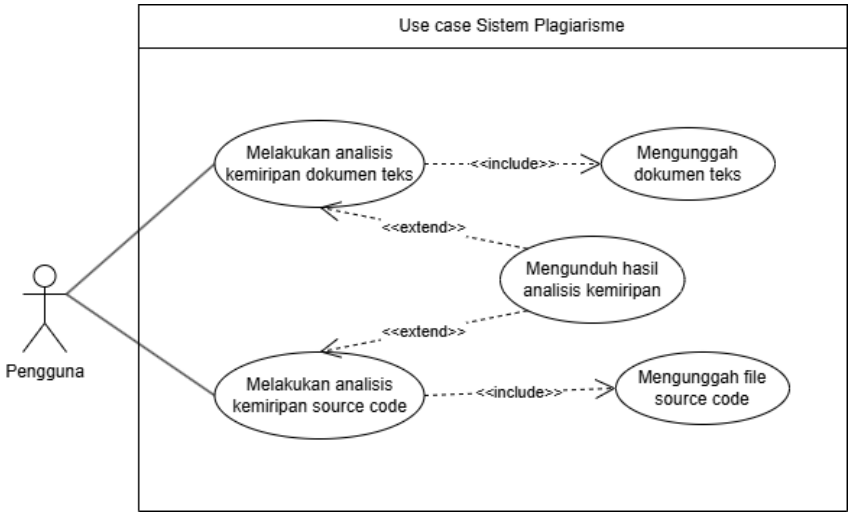
   Frontend merupakan antarmuka pengguna yang diakses melalui browser tanpa memerlukan instalasi aplikasi tambahan. Antarmuka sistem dibangun menggunakan HTML dan CSS dengan mekanisme *templating* Jinja2 yang disajikan oleh *framework* Flask. Penggunaan Jinja2 memungkinkan halaman web ditampilkan secara dinamis berdasarkan hasil pemrosesan yang dilakukan pada sisi server. Antarmuka ini menyediakan fitur bagi pengguna untuk mengunggah dokumen teks atau *source code* yang

akan dianalisis serta menampilkan hasil analisis kemiripan dalam bentuk tampilan *dashboard.*

4. *Backend* (Web Server)
   Backend sistem dibangun menggunakan *Flask framework* berbasis *Python.* Komponen ini berfungsi sebagai pusat pengolahan sistem yang menangani penerimaan dokumen dari pengguna, proses *preprocessing* dokumen, serta perhitungan nilai kemiripan menggunakan model yang telah disimpan pada *project* aplikasi. Setelah proses analisis selesai dilakukan, hasil analisis kemudian dikirim kembali ke *frontend* untuk ditampilkan kepada pengguna dalam bentuk tampilan *dashboard* hasil.

3.6.2 *Use Case Diagram*

Use Case Diagram digunakan untuk memodelkan kebutuhan fungsional sistem dari sudut pandang pengguna (*user-centered).* Tujuan utama *Use Case Diagram* adalah untuk memberikan gambaran sederhana namun jelas mengenai perilaku sistem pada level tinggi sebelum masuk ke tahap perancangan teknis atau implementasi. Dengan demikian, diagram ini membantu memastikan bahwa sistem yang dibangun benar-benar memenuhi kebutuhan pengguna dan tidak keluar dari batas ruang lingkup yang telah ditetapkan.



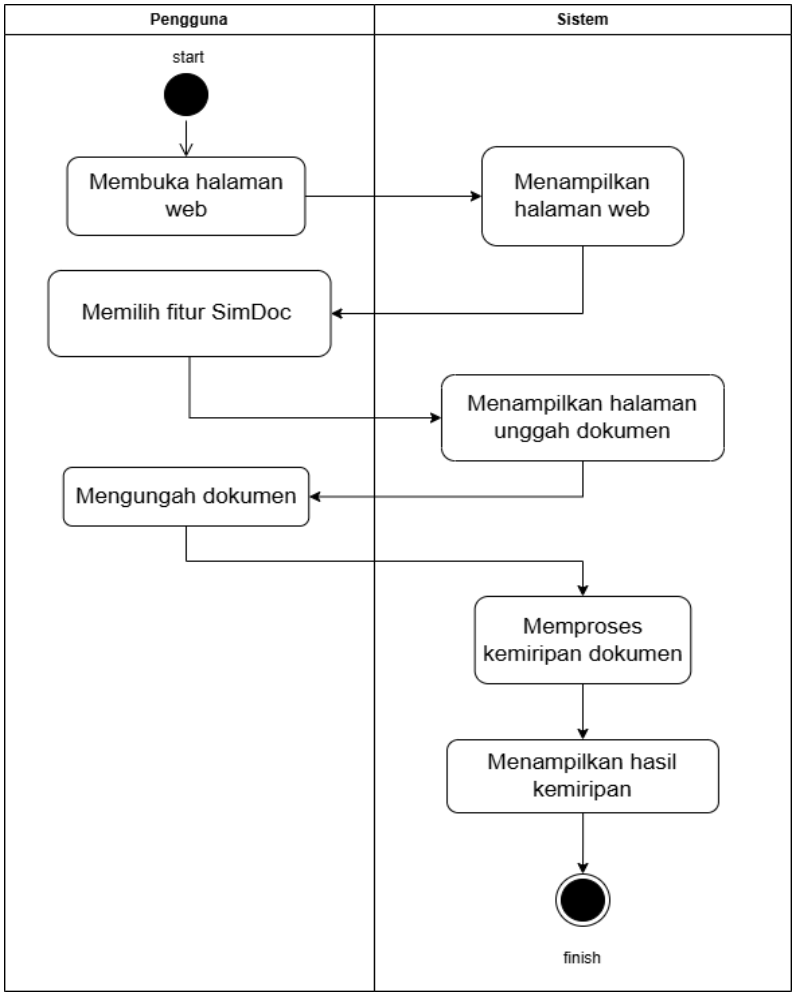
Gambar 9. *Use Case Diagram* Sistem Plagiarisme.

Pada Gambar 9 menunjukkan *use case* sistem analisis plagiarisme yang melibatkan satu aktor yaitu Pengguna. Sistem menyediakan dua fungsi utama yaitu analisis kemiripan dokumen teks dan analisis kemiripan *source code.* Kedua proses tersebut memiliki relasi *include* dengan proses mengunggah berkas, yang menunjukkan bahwa file harus diunggah terlebih dahulu sebelum proses analisis dilakukan. Selain itu, sistem juga menyediakan fitur mengunduh hasil analisis kemiripan yang memiliki relasi *extend*, yang berarti proses tersebut bersifat opsional dan dapat dilakukan setelah proses analisis selesai.

3.6.3 *Activity Diagram*

Activity Diagram digunakan untuk menggambarkan alur aktivitas atau proses yang terjadi di dalam sistem, baik dari sisi pengguna maupun sistem itu sendiri. Diagram ini menampilkan urutan aktivitas, percabangan keputusan, dan interaksi antar komponen sistem secara terstruktur sehingga memberikan gambaran menyeluruh mengenai bagaimana suatu proses dijalankan dari awal hingga selesai.

1. Melakukan Analisis Kemiripan Dokumen Teks



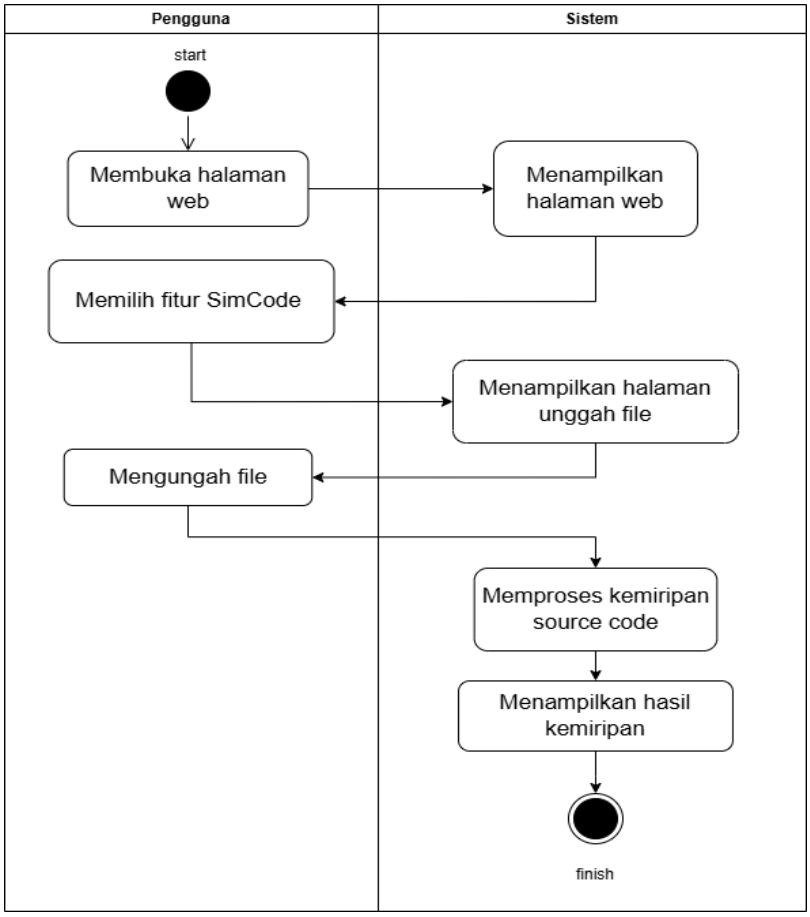
Gambar 10. *Activity Diagram* Melakukan analisis Kemiripan Dokumen Teks.

Activity Diagram pada Gambar 10 menggambarkan alur proses analisis kemiripan dokumen teks yang dilakukan oleh pengguna melalui sistem. Proses dimulai ketika pengguna membuka halaman web, kemudian sistem menampilkan halaman utama web. Selanjutnya pengguna memilih fitur SimDoc untuk melakukan analisis kemiripan dokumen teks sehingga sistem menampilkan halaman unggah dokumen. Setelah halaman tersebut ditampilkan, pengguna mengunggah dokumen yang akan dianalisis.

Setelah dokumen berhasil diunggah, sistem melanjutkan proses dengan memproses kemiripan dokumen menggunakan metode

yang telah diimplementasikan dalam sistem. Setelah proses analisis selesai, sistem menampilkan hasil kemiripan dokumen kepada pengguna sebagai keluaran dari proses analisis.

2. *Activity Diagram* Melakukan analisis Kemiripan *Source Code*

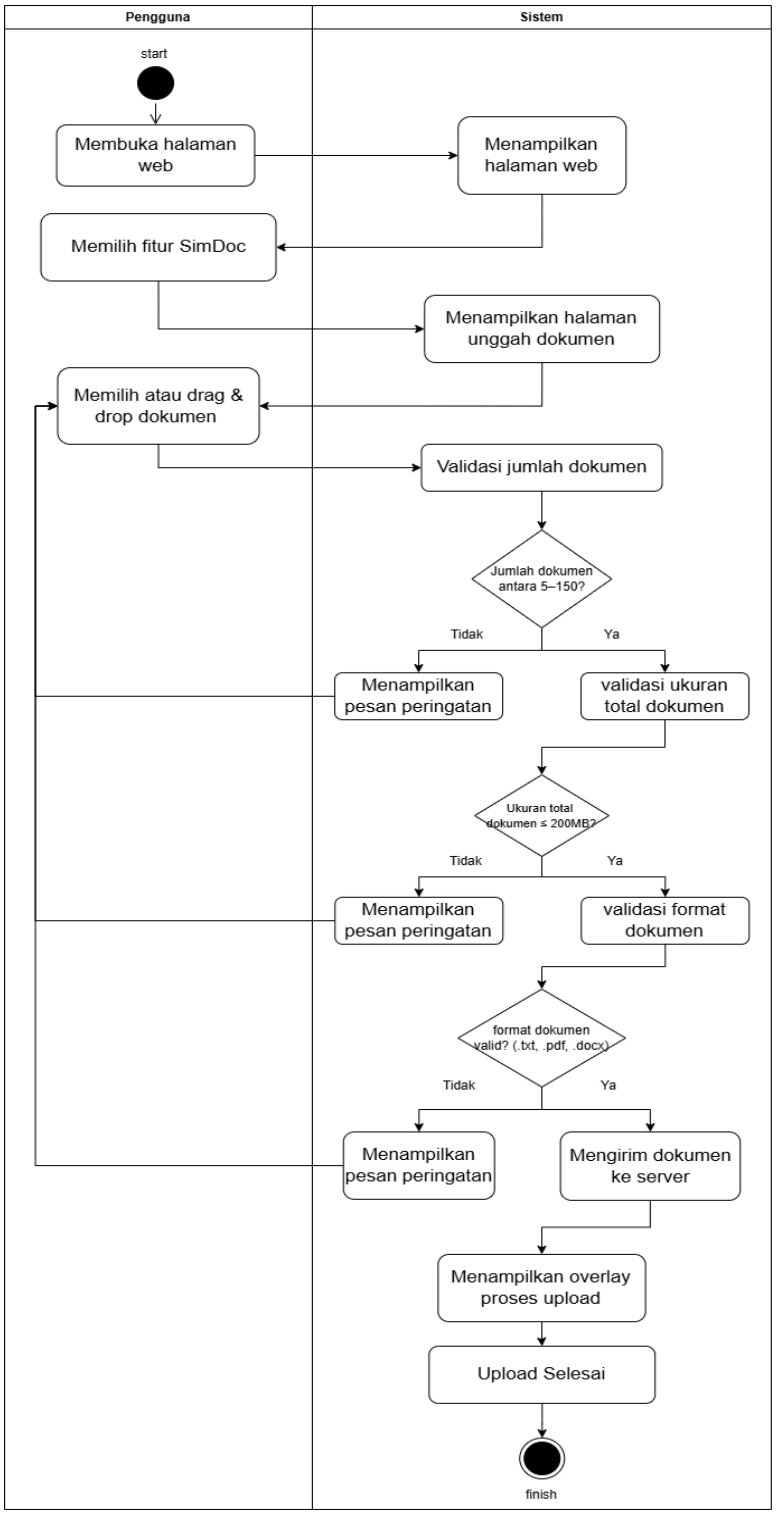


Gambar 11. *Activity Diagram* Melakukan analisis Kemiripan *Code.*

Activity Diagram pada Gambar 11 menggambarkan alur proses analisis kemiripan *source code* pada sistem. Proses dimulai ketika pengguna membuka halaman web dan sistem menampilkan halaman utama. Pengguna kemudian memilih fitur SimCode sehingga sistem menampilkan halaman untuk mengunggah file. Setelah file berhasil diunggah, sistem akan melanjutkan proses dengan menghitung kemiripan *source code* menggunakan metode yang telah diimplementasikan. Setelah proses analisis selesai,

sistem menampilkan hasil kemiripan *source code* kepada pengguna sebagai keluaran dari proses analisis.

3. *Activity Diagram* Mengunggah Dokumen Teks

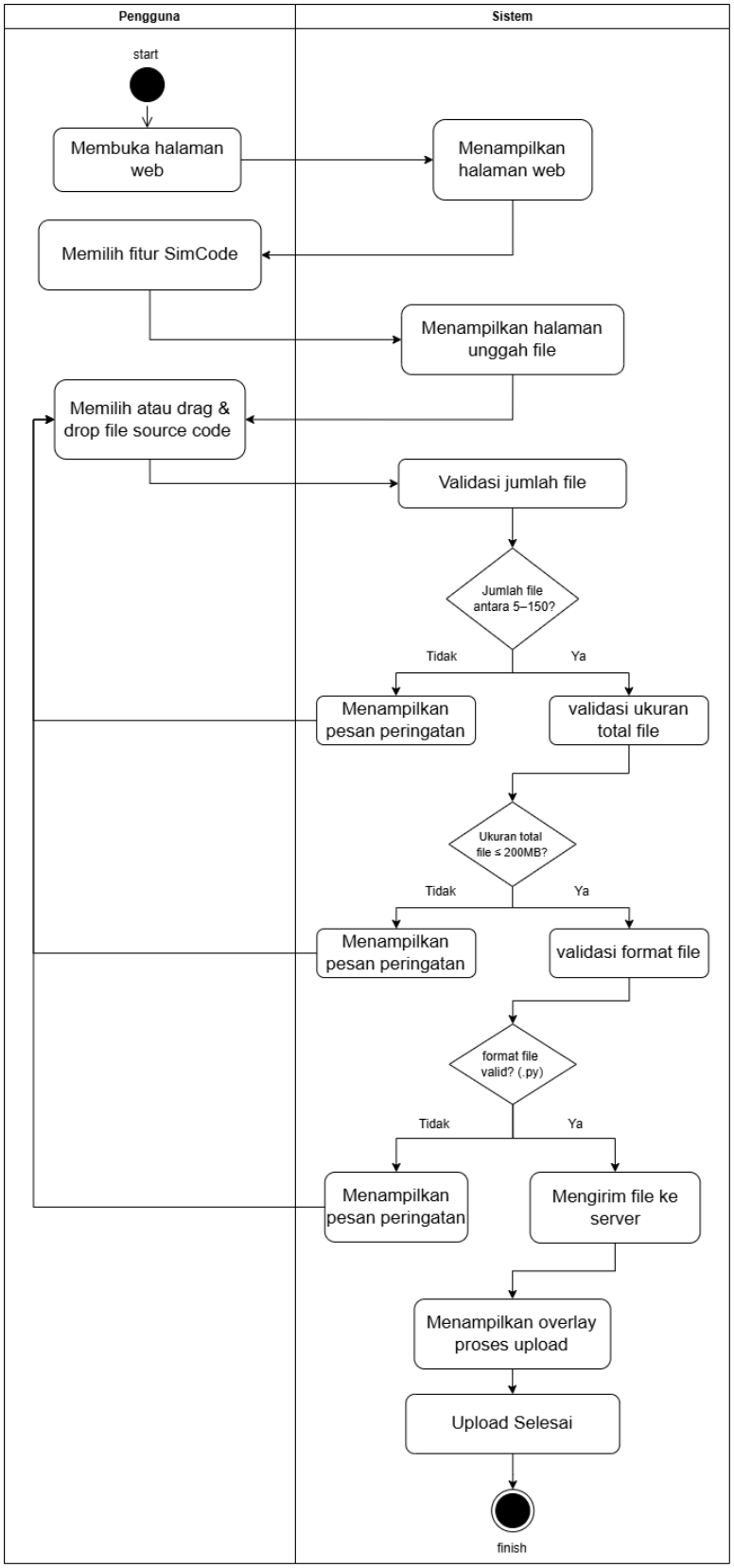


Gambar 12. Activity Diagram Mengunggah Dokumen Teks.

Activity Diagram pada Gambar 12 menggambarkan proses unggah dokumen yang dilakukan oleh pengguna sebelum proses analisis kemiripan dokumen dijalankan. Proses dimulai ketika pengguna membuka halaman web dan memilih fitur SimDoc sehingga sistem menampilkan halaman unggah dokumen. Pengguna kemudian dapat memilih dokumen secara manual atau menggunakan fitur drag and drop untuk mengunggah beberapa dokumen sekaligus. Setelah dokumen dipilih, sistem melakukan validasi jumlah dokumen dengan ketentuan minimal 5 dokumen dan maksimal 150 dokumen. Jika jumlah dokumen tidak memenuhi ketentuan, sistem akan menampilkan pesan peringatan kepada pengguna.

Apabila jumlah dokumen valid, sistem melanjutkan dengan memvalidasi ukuran total dokumen dengan batas maksimum 200 MB. Jika ukuran dokumen melebihi batas yang ditentukan, sistem kembali menampilkan pesan peringatan. Selanjutnya sistem memvalidasi format dokumen yang diunggah dengan format yang diperbolehkan yaitu .txt, .pdf, dan .docx. Jika format dokumen sesuai, sistem akan mengunggah dokumen ke server dan menampilkan *overlay* proses upload sebagai indikator bahwa proses unggah sedang berlangsung. Setelah proses upload selesai, dokumen siap untuk digunakan pada tahap analisis kemiripan dokumen.

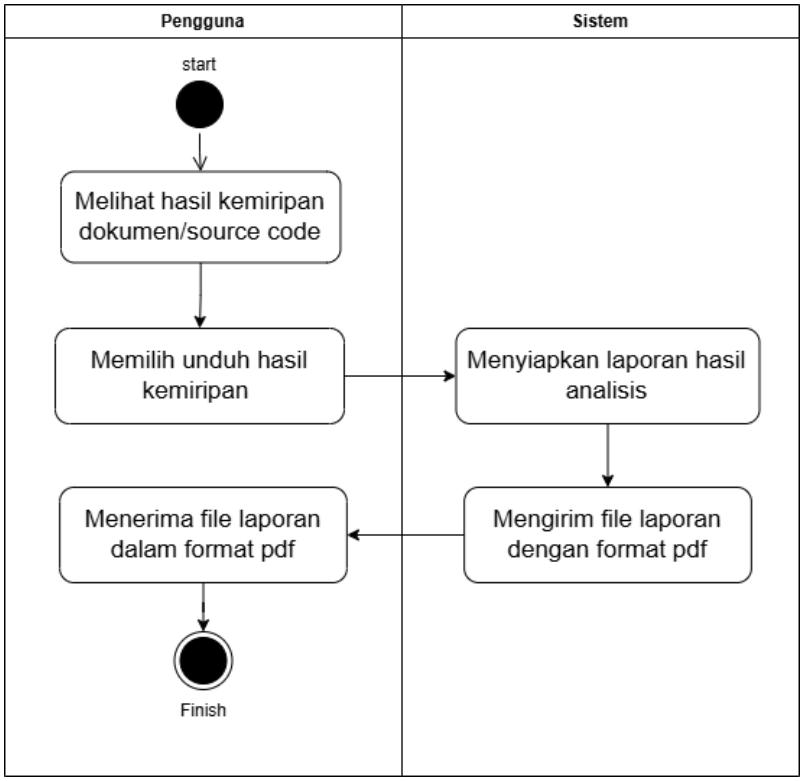
4. Activity Diagram Mengunggah *Source Code*



Gambar 13. *Activity Diagram* Mengunggah *Source Code.*

Activity Diagram pada Gambar 13 menggambarkan alur proses Mengunggah file *Source Code* oleh pengguna. Alur proses pada diagram ini secara umum memiliki tahapan yang serupa dengan mengunggah dokumen teks, namun perbedaannya terletak pada jenis format dokumen. Dalam *source code,* file yang diunggah harus berformat *.py.*

5. *Activity Diagram* Mengunduh Hasil Analisis Kemiripan



Gambar 14. *Activity Diagram* Mengunduh Hasil Analisis Kemiripan.

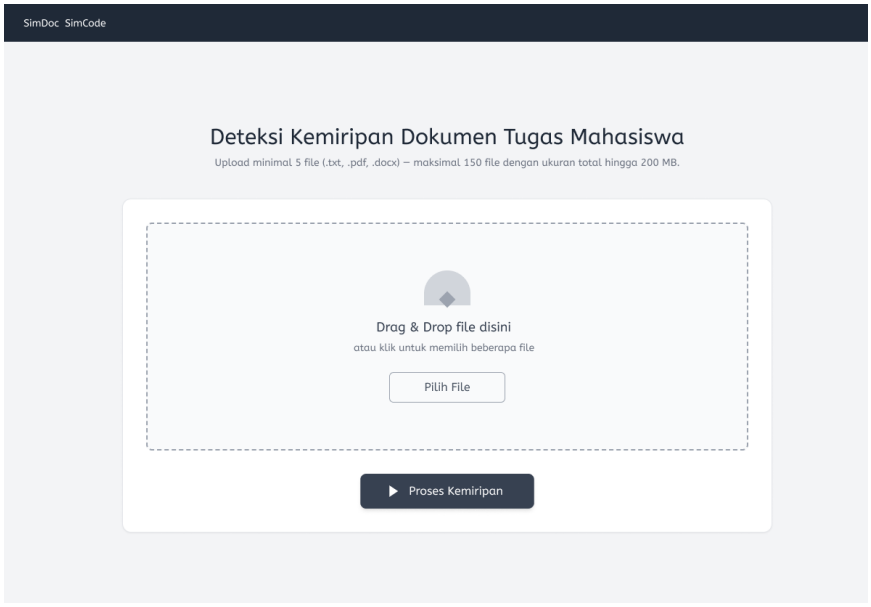
Activity Diagram pada Gambar 14 menggambarkan proses pengunduhan hasil analisis kemiripan oleh pengguna. Proses dimulai ketika pengguna telah melihat hasil kemiripan dokumen atau *source code* yang ditampilkan oleh sistem. Selanjutnya pengguna memilih fitur untuk mengunduh hasil analisis kemiripan. Setelah itu, sistem menyiapkan laporan hasil analisis dalam bentuk dokumen PDF berdasarkan hasil perhitungan kemiripan yang telah diperoleh pada proses analisis sebelumnya. Laporan tersebut kemudian dikirimkan kepada pengguna

sehingga pengguna dapat menyimpan hasil analisis kemiripan pada perangkatnya. Laporan hasil analisis dihasilkan secara dinamis oleh sistem berdasarkan hasil perhitungan kemiripan yang telah dilakukan sebelumnya tanpa melalui proses penyimpanan data pada *database.*

3.6.4 Rancangan Antarmuka

Pada tahap perancangan sistem, dilakukan penyusunan *wireframe* antarmuka untuk memberikan gambaran awal mengenai struktur tampilan dan interaksi yang akan disediakan oleh sistem. *Wireframe* digunakan sebagai representasi visual sederhana yang menunjukkan tata letak komponen utama pada halaman sistem sebelum proses implementasi antarmuka dilakukan.

1. *Wireframe* Halaman Deteksi Dokumen Teks



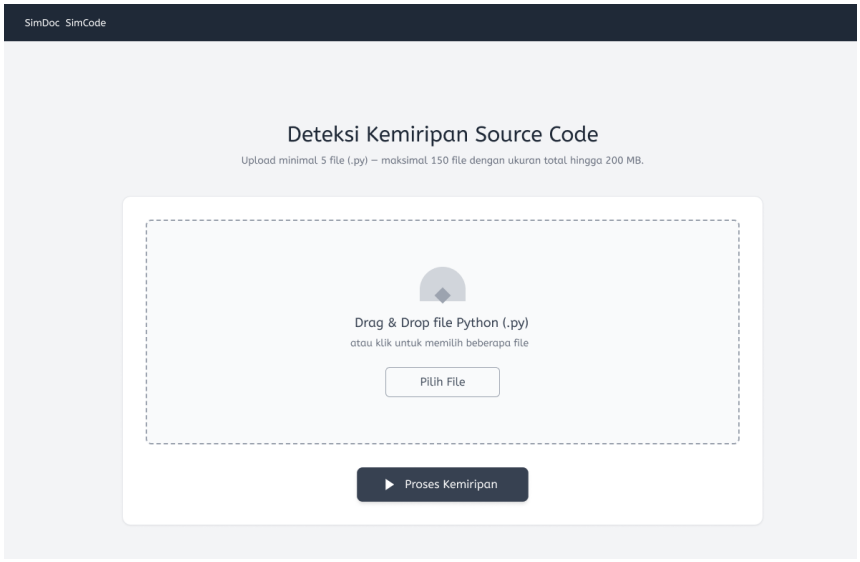
Gambar 15. *Wireframe* Halaman Deteksi Dokumen Teks.

Pada Gambar 15 *wireframe* tampilan sistem yang digunakan oleh pengguna untuk melakukan proses analisis kemiripan dokumen tugas mahasiswa. Pada halaman ini pengguna diberikan fasilitas

untuk mengunggah beberapa dokumen teks yang akan dianalisis oleh sistem.

Bagian utama halaman terdiri dari area unggah dokumen yang mendukung mekanisme *drag and drop* maupun pemilihan file secara manual melalui tombol *Pilih File*. Sistem memungkinkan pengguna untuk mengunggah beberapa dokumen sekaligus dengan format file tertentu seperti *.txt*, *.pdf*, dan *.docx*. Setelah dokumen berhasil dipilih, pengguna dapat menjalankan proses analisis dengan menekan tombol Proses Kemiripan.

2. *Wireframe* Halaman Deteksi *Source Code*

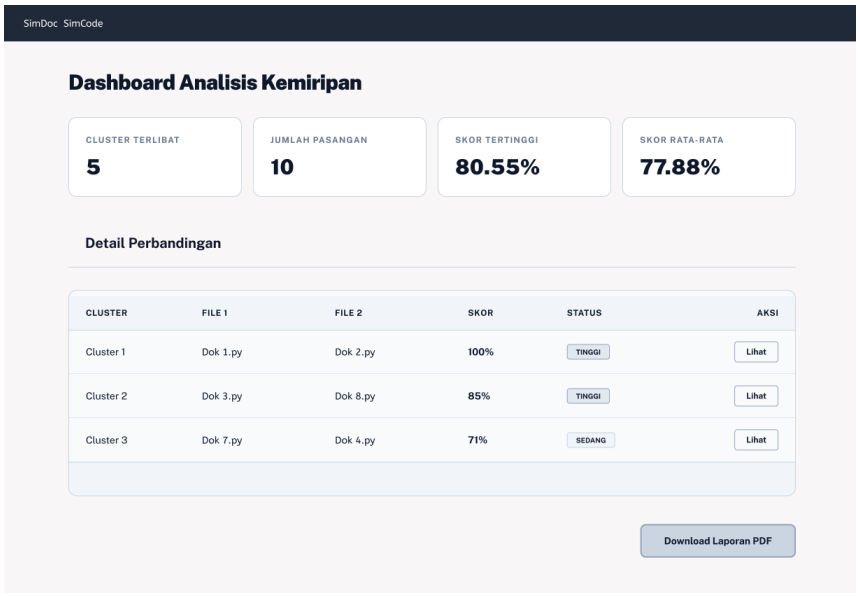


Gambar 16. *Wireframe* Halaman Deteksi *Source Code.*

Pada Gambar 16 di atas menunjukkan *wireframe* yang digunakan untuk melakukan analisis kemiripan pada file *source code.* Pada halaman ini pengguna dapat mengunggah beberapa file program yang akan dibandingkan tingkat kemiripannya oleh sistem. Bagian utama halaman menyediakan area unggah file yang mendukung mekanisme *drag and drop* maupun pemilihan file melalui tombol Pilih File. Sistem dirancang untuk menerima beberapa file *source code* dengan format *.py*. Setelah file dipilih,

pengguna dapat menjalankan proses analisis dengan menekan tombol Proses Kemiripan.

3. *Wireframe* Halaman Hasil Deteksi



Gambar 17. *Wireframe* Halaman Hasil Deteksi.

Gambar 17 menunjukkan *wireframe* halaman *Dashboard* Analisis Kemiripan yang menampilkan hasil perhitungan kemiripan dokumen atau *source code* yang telah diproses oleh sistem. Halaman ini berfungsi untuk memberikan ringkasan informasi hasil analisis secara keseluruhan kepada pengguna.

Pada bagian atas halaman ditampilkan beberapa informasi ringkasan, seperti jumlah cluster terlibat, jumlah pasangan dokumen, skor kemiripan tertinggi, serta skor rata-rata kemiripan. Informasi tersebut bertujuan untuk memberikan gambaran umum mengenai tingkat kemiripan dari dokumen atau *source code* yang dianalisis. Selain itu, halaman ini juga menampilkan tabel detail perbandingan yang berisi informasi pasangan file yang dibandingkan, nilai skor kemiripan yang dihasilkan, serta status tingkat kemiripan. Pengguna juga dapat memilih tombol Lihat untuk melihat detail hasil analisis dari pasangan dokumen

tertentu. Pada bagian bawah halaman tersedia tombol *Download Laporan PDF* untuk mengunduh hasil analisis dalam bentuk laporan.

3.7 *Coding*

3.7.1 Pembentukan Model Representasi Dokumen Teks

3.7.1.1 *Preprocessing* Dokumen Teks

Penelitian tahap *preprocessing* merupakan langkah penting dalam pengolahan teks yang bertujuan untuk mentransformasi data mentah menjadi bentuk bersih, seragam, dan siap dianalisis oleh model. Proses ini dilakukan untuk menghilangkan elemen yang tidak relevan, menyamakan struktur teks, serta meningkatkan kualitas representasi data. Alur *preprocessing* yang dilakukan terdiri dari beberapa langkah, yaitu:

a. Text Cleaning

Text Cleaning merupakan tahap pembersihan teks untuk menghilangkan elemen yang tidak diperlukan dalam analisis. Proses ini mencakup beberapa tugas, seperti:

1) *Case Folding*

   Ini merupakan proses untuk mengubah seluruh karakter dalam teks menjadi huruf kecil *(lowercase)* agar sistem tidak membedakan kata berdasarkan kapitalisasi huruf.

2) *Number Removal*

   Proses untuk menghapus semua karakter numerik (angka) dari teks karena angka umumnya tidak relevan dalam analisis kemiripan semantik dokumen akademik.

3) *Punctuation Removal*

   Proses menghilangkan tanda baca seperti titik (.), koma (,), tanda seru (!), tanda tanya (?), dan karakter khusus lainnya yang tidak membawa makna dalam *text mining.*

4) *Whitespace Removal*

Proses menormalisasi spasi dengan menghapus spasi berlebih, tab, dan *newline* yang tidak perlu, sehingga hanya tersisa satu spasi antar kata.

b. *Tokenization*

Proses pemecahan teks menjadi potongan – potongan kata individual yang disebut *token*. Proses ini menggunakan pustaka *nltk* atau *Sastrawi* agar memudahkan analisis frekuensi dan konteks.

c. *Stopword Removal*

Token hasil tokenisasi selanjutnya diproses dengan tahap penghapusan *stopword* menggunakan *library Sastrawi. Stopword* merupakan kata-kata umum dalam Bahasa Indonesia yang tidak memberikan kontribusi signifikan terhadap makna dokumen. Oleh karena itu, kata-kata seperti *"dan"*, *"yang"*, *"atau"*, dan *"di"* dihapus dari hasil tokenisasi agar analisis lebih berfokus pada kata-kata yang memiliki makna penting dalam dokumen.

3.7.1.2 Representasi Dokumen

Setelah data teks melalui tahap *preprocessing*, langkah selanjutnya adalah melakukan representasi dokumen ke dalam bentuk numerik agar dapat diproses oleh algoritma analisis kemiripan. Pada penelitian ini, proses representasi dilakukan melalui tiga skenario berbeda, yang masing-masing menggunakan pendekatan representasi fitur yang berbeda untuk membandingkan kinerja metode dalam analisis kemiripan dokumen.

1) Skenario 1 (TF-IDF)

Metode pertama yang digunakan adalah TF-IDF. Kata yang sering muncul pada satu dokumen namun jarang muncul pada dokumen lain akan memiliki bobot TF-IDF tinggi, sedangkan

kata yang umum digunakan di banyak dokumen akan memiliki bobot rendah. Proses ini dilakukan menggunakan model `TfidfVectorizer` yang dilatih pada korpus teks bersih (hasil *preprocessing)* untuk membangun kosakata dan menghitung bobot IDF. Hasilnya, setiap dokumen ditransformasikan menjadi vektor TF-IDF yang merepresentasikan kata-kata kunci paling penting dalam dokumen tersebut

2) Skenario 2 *(Word2Vec)*

Dalam penelitian ini, representasi semantik dokumen teks diperoleh menggunakan *pre-trained Word2Vec (FastText).* Model pra-latih *Word2Vec* tersebut dimuat menggunakan pustaka *Gensim,* kemudian digunakan sebagai *lookup – table* untuk mengubah setiap token dalam dokumen menjadi vektor berdimensi tetap. Pada tahap ini, bobot *embedding* tidak diperbarui *(non-trainable),* sehingga proses lebih efisien dan menghindari risiko *overfitting.* Setiap token valid yang terdapat pada *vocabulary* model akan direpresentasikan sebagai vektor numerik berdimensi *d*, sedangkan token yang tidak ditemukan dapat ditangani dengan mekanisme inisialisasi acak atau diabaikan sesuai kebijakan *preprocessing*. Untuk memperoleh representasi vektor dokumen, seluruh vektor kata yang ditemukan dalam kosakata model dijumlahkan lalu dibagi jumlah kata valid *(mean pooling*), sehingga menghasilkan satu vektor dokumen berdimensi 300.

3) Skenario 3 (TF-IDF + *Word2Vec*)

Representasi dokumen teks dibangun dengan mengombinasikan dua pendekatan, yaitu TF-IDF dan *pretrained Word2Vec (FastText)*, untuk menghasilkan representasi yang lebih kaya. TF-IDF digunakan untuk menangkap informasi statistik dan kepentingan kata, sedangkan *pretrained Word2Vec* digunakan

untuk merepresentasikan konteks dan makna semantik. Penggabungan dilakukan dengan memberikan bobot α pada vektor TF-IDF dan bobot β pada vektor *FastText* sebelum digabungkan, di mana nilai optimal α dan β ditentukan melalui proses *tuning* pada tahap implementasi. Representasi gabungan ini diharapkan mampu memberikan performa analisis kemiripan yang lebih baik dibandingkan penggunaan satu metode saja.

3.7.1.3 *Feature Concatenation*

Tahap penggabungan fitur *(feature concatenation*) hanya diterapkan pada Skenario 3, yaitu skenario yang dilakukan untuk memperoleh representasi dokumen yang lebih komprehensif dengan mengintegrasikan dua jenis informasi berbeda, yaitu aspek statistik dan aspek semantik. Pada tahap ini, vektor hasil pembobotan TF-IDF, yang merepresentasikan frekuensi relatif serta tingkat kepentingan kata dalam dokumen, digabungkan dengan vektor hasil *pretrained Word2Vec (FastText),* yang menangkap makna kontekstual dan hubungan semantik antar kata.

Sebelum digabungkan, kedua vektor dinormalisasi L2 terlebih dahulu agar berada pada skala yang sama, kemudian masing-masing diberi bobot sesuai kontribusinya. Proses penggabungan dilakukan secara horizontal (*concatenation*) menggunakan rumus berikut:

$$v\_hybrid = [\ \alpha \times V_{TFIDF} \ ||\ \beta\ \times V_{FastText}\ ] \qquad (12)$$

Di mana pada Persamaan 12, α adalah bobot untuk vektor TF-IDF, β adalah bobot untuk vektor *FastText*, dan || menyatakan operasi *concatenation* horizontal. Nilai optimal α dan β ditentukan melalui proses *tuning* pada tahap implementasi. Vektor *hybrid* yang dihasilkan kemudian dinormalisasi L2 kembali sebelum digunakan sebagai input pada tahap *K-Means Clustering* dan perhitungan *Cosine Similarity.*

3.7.2 Pembentukan Model Representasi Teks *Source Code*

3.7.2.1 *Prepocessing Source Code*

Preprocessing source code merupakan tahapan awal yang bertujuan untuk menyederhanakan dan menstandarkan kode program sebelum dilakukan analisis kemiripan. Tahap ini penting karena *source code* yang ditulis oleh mahasiswa dapat memiliki perbedaan gaya penulisan, format, dan penamaan, meskipun secara logika program memiliki kesamaan. Tanpa *preprocessing,* perbedaan-perbedaan tersebut dapat memengaruhi hasil analisis dan menurunkan akurasi pendeteksian kemiripan.

Oleh karena itu, *preprocessing* pada *source code* dalam penelitian ini difokuskan pada normalisasi dan standarisasi representasi kode agar analisis lebih menekankan pada struktur dan logika program dibandingkan aspek visual atau format penulisan. Pendekatan ini sejalan dengan teknik *token-based code clone detection* yang umum digunakan dalam penelitian deteksi kemiripan *source code* (Sneha et al., 2024). Tahapan *preprocessing source code* yang dilakukan meliputi:

1) Penghapusan Komentar dan *Docstring*

Komentar dan *docstring* tidak memengaruhi logika eksekusi program, sehingga berpotensi menimbulkan bias dalam pengukuran kemiripan apabila disertakan dalam proses analisis. Oleh karena itu, seluruh komentar dan *docstring* dihilangkan dari proses ekstraksi fitur. Jenis komentar yang ditangani meliputi komentar satu baris menggunakan simbol #, serta *docstring,* yaitu *string literal* bertipe *multi-baris* yang dituliskan menggunakan tanda kutip tiga (''' ... ''' atau """ ... """) dan digunakan untuk dokumentasi modul, kelas, atau fungsi.

Penghapusan dilakukan menggunakan pendekatan *masking* dengan dua mekanisme terpisah. Pertama, penghapusan *docstring*

dilakukan menggunakan *Abstract Syntax Tree* (AST) untuk mendeteksi posisi *docstring* secara sintaksis pada modul, fungsi, dan kelas. Selanjutnya, dilakukan penghapusan komentar # menggunakan pemrosesan berbasis karakter dengan menelusuri setiap karakter secara berurutan sambil melacak status apakah karakter berada di dalam *string literal* atau tidak, sehingga simbol # di dalam *string* tidak ikut terhapus.

Pada kedua mekanisme tersebut, karakter yang dihapus diganti dengan spasi tanpa mengubah panjang keseluruhan teks. Pendekatan ini dipilih agar posisi karakter dan *offset* token tetap konsisten terhadap *source code* asli, sehingga baris kode yang terdeteksi memiliki kemiripan dapat di-*highlight* secara tepat pada tampilan antarmuka web. Hasil dari tahap ini disimpan sebagai *masked code* yang digunakan sebagai *input* pada seluruh tahap selanjutnya.

2) Tokenisasi *Source Code*

Proses memecah *masked code* menjadi unit token menggunakan *tokenizer* yang dirancang khusus untuk *source code Python.* Token yang dihasilkan mencakup: *keyword (def, for, if, return, dan class),* operator sintaksis (+, =, !=, -), *identifier,* serta *literal* angka dan *string.* Pendekatan berbasis token dipilih karena lebih sesuai untuk mengukur kemiripan *source code,* sebab fokusnya pada pola logika dan struktur program, bukan kesamaan teks mentah (Sneha et al., 2024). Dalam proses tokenisasi, elemen format seperti spasi, tab, dan indentasi secara otomatis diabaikan karena tidak termasuk dalam token sintaks utama, sehingga perbedaan format penulisan tidak memengaruhi representasi token yang digunakan dalam analisis. Hasil tokenisasi kemudian dinormalisasi agar lebih tahan terhadap upaya manipulasi plagiarisme, dengan dua strategi utama:

1. Tokenisasi dengan *Identifier* Asli

   Berbeda dengan pendekatan normalisasi generik yang menyeragamkan nama variabel menjadi token seperti VAR atau FUNC, penelitian ini mempertahankan nama *identifier* asli (nama variabel, fungsi, dan parameter). Pendekatan ini dipilih berdasarkan pertimbangan bahwa perbedaan penamaan *identifier* merupakan salah satu indikator orisinalitas kode yang perlu dipertahankan dalam proses analisis. Dengan mempertahankan nama asli, sistem dapat membedakan antara dua file yang menggunakan nama variabel berbeda meskipun memiliki struktur logika yang sama. Hanya *keyword* bawaan bahasa *Python* (seperti *if, for, def, class, return*) yang dibiarkan apa adanya karena merupakan bagian dari sintaksis bahasa dan tidak dapat dimodifikasi oleh mahasiswa. Selain itu, operator sintaksis juga dipertahankan karena merepresentasikan struktur logika dan alur program yang relevan dalam analisis kemiripan.

2. Normalisasi *Whitespace*

   Setelah komentar dan *docstring* dihapus, dilakukan normalisasi *whitespace* yaitu proses membersihkan spasi berlebih, tab, dan indentasi yang tidak konsisten antar file. Hal ini bertujuan agar perbedaan gaya penulisan format kode tidak memengaruhi hasil analisis kemiripan. Nilai literal angka dan string tetap dipertahankan dalam bentuk aslinya sebagai bagian dari token yang ikut dihitung dalam representasi TF-IDF, sehingga perbedaan nilai literal juga berkontribusi dalam membedakan file *source code* satu dengan lainnya.

Setelah tokenisasi dan normalisasi, semua token digabungkan menjadi dokumen baru yang berisi representasi token dari *source code,* lalu dokumen ini siap digunakan untuk perhitungan TF-IDF dan pengukuran kesamaan dengan *Cosine Similarity.*

3.7.2.2 Ekstraksi Fitur *Source Code*

Setelah melalui tahap *preprocessing, source code* yang sudah dinormalisasi dan ditokenisasi perlu diubah menjadi bentuk representasi numerik agar dapat dianalisis menggunakan metode yang digunakan dalam penelitian ini. Pada tahap ini, ekstraksi fitur *source code* dilakukan menggunakan pendekatan TF-IDF dengan memandang setiap token kode sebagai istilah *(term)* yang dapat dihitung bobotnya (Eka Putra & Supriana, 2022).

Dalam konteks *source code,* TF-IDF digunakan untuk mengukur seberapa penting suatu token seperti *keyword Python (if, for, while, return, def),* nama identifier, maupun operator (==, =, +) di dalam satu berkas program dibandingkan dengan seluruh dokumen *source code* lainnya. Prosesnya berlangsung dalam dua langkah utama:

1) Perhitungan *Term Frequency* (TF)
   Pada tahap ini dihitung frekuensi kemunculan setiap token di dalam satu dokumen *source code* menggunakan pendekatan *sublinear TF*. Penggunaan *sublinear TF* dipilih untuk meredam dominasi token yang muncul sangat sering sehingga token dengan frekuensi sedang tetap mendapatkan bobot yang proporsional. Misalnya, jika struktur perulangan *for* atau pemanggilan fungsi tertentu sering digunakan dalam satu program, nilai TF token-token tersebut akan lebih tinggi namun tidak mendominasi secara berlebihan.

2) Perhitungan *Inverse Document Frequency* (IDF)
   Selanjutnya dihitung IDF untuk setiap *token* berdasarkan jumlah dokumen yang mengandung *token* tersebut. *Token* yang sangat umum dan muncul di hampir semua dokumen akan memiliki nilai IDF yang rendah. Sebaliknya, *token* yang hanya muncul pada sebagian dokumen, misalnya kombinasi pola tertentu atau fungsi yang khas, akan memiliki nilai IDF yang lebih tinggi. Dengan

cara ini, TF-IDF menurunkan bobot token yang terlalu generik dan menonjolkan token yang lebih spesifik terhadap pola kode tertentu.

Gabungan antara nilai TF dan IDF menghasilkan bobot TF-IDF untuk setiap token pada masing-masing dokumen *source code.* Melalui pendekatan ini, setiap program direpresentasikan sebagai sebuah vektor fitur yang berisi bobot setiap token yang muncul dalam korpus. Representasi berbasis vektor tersebut memungkinkan sistem untuk mengenali pola penulisan program serta struktur logika yang digunakan, seperti kecenderungan penggunaan perulangan, banyaknya operasi perbandingan, atau pola pemanggilan fungsi tertentu.

Vektor TF-IDF ini selanjutnya digunakan sebagai masukan dalam dua tahap analisis yaitu sebagai *input* perhitungan *TF-IDF Cosine Similarity* pada level dokumen untuk mengukur kemiripan distribusi token secara keseluruhan, serta sebagai dasar representasi dalam perhitungan *Line-based Cosine Similarity* pada level baris kode. Selain itu, vektor TF-IDF juga digunakan dalam proses klasterisasi dengan algoritma *K-Means* untuk mengelompokkan *file source code* berdasarkan kemiripan pola tokennya.

3.7.2.3 *Line-based Cosine Similarity*

Selain ekstraksi fitur berbasis token secara global, penelitian ini juga menerapkan pendekatan *Line-based Cosine Similarity* sebagai komponen kedua dalam *dual scoring*. Sebagaimana ditunjukkan oleh Gandhi et al. (2024) bahwa pengukuran kemiripan pada level struktur baris kode yang dapat dieksekusi mampu menangkap aspek kemiripan yang tidak tertangkap oleh pendekatan berbasis token secara global, pendekatan ini dirancang untuk mengukur kemiripan antara dua file *source code* pada level baris kode.

Proses perhitungan *Line-based Cosine Similarity* dimulai dengan memecah *masked code* setiap file menjadi daftar baris kode aktif, yaitu baris yang tidak kosong setelah komentar dan *docstring* dihapus pada tahap *preprocessing.* Mengacu pada pendekatan yang dikemukakan oleh Karnalim (2020) bahwa pembobotan TF-IDF pada unit kode memprioritaskan kecocokan yang unik dan jarang muncul sebagai indikator kemiripan yang lebih kuat, setiap baris kode dalam penelitian ini diperlakukan sebagai satu term dalam representasi TF-IDF sehingga satu baris kode secara utuh menjadi satu unit analisis tersendiri. Konfigurasi TF-IDF yang digunakan pada tahap ini juga menerapkan *sublinear TF* agar konsisten dengan representasi TF-IDF global.

Setelah kedua file direpresentasikan sebagai vektor TF-IDF berbasis baris, *Cosine Similarity* dihitung antara kedua vektor tersebut untuk menghasilkan nilai *line_cosine* yang merepresentasikan seberapa besar proporsi baris kode yang serupa antara dua file. Sejalan juga dengan yang dikemukakan oleh Martinez-Gil (2024) bahwa TF-IDF *Similarity* mengukur distribusi *term* berdasarkan frekuensi relatif secara keseluruhan tanpa bergantung pada urutan kemunculannya, pendekatan ini efektif mendeteksi kemiripan meskipun terdapat perbedaan urutan baris atau penambahan baris baru pada file yang dibandingkan.

Nilai *line_cosine* yang dihasilkan kemudian digabungkan dengan nilai TF-IDF *Cosine Similarity* melalui mekanisme pembobotan sebagai bagian dari pendekatan *dual scoring*. Sebagaimana ditekankan oleh Gandhi et al. (2024) bahwa pemilihan bobot relatif antar komponen berperan signifikan dalam menentukan efektivitas *final score,* bobot dalam penelitian ini ditetapkan berdasarkan karakteristik masing-masing representasi. TF-IDF berbasis token diberi bobot lebih besar (0,6) karena mencakup pola penggunaan kata kunci, nama fungsi, dan struktur token secara global, sementara *Line-based Cosine Similarity*

diberi bobot lebih kecil (0,4) sebagai komponen pelengkap yang memperkuat deteksi pada level implementasi baris kode, sehingga *final score* dihitung menggunakan Rumus (10) berikut.

$$final\ score = (0{,}6\ \times TF - IDF\ CS) + (0{,}4\ \times Line - based\ CS) \quad (10)$$

Nilai *final score* yang dihasilkan dari kombinasi kedua komponen ini selanjutnya digunakan sebagai dasar penentuan tingkat kemiripan antar file *source code* dalam sistem deteksi yang dikembangkan.

3.7.3 *K-Means Clustering*

Pada tahap ini *clustering* berfungsi untuk mengelompokkan dokumen berdasarkan kemiripan representasi fitur yang telah dibentuk sebelumnya. Seperti yang diterapkan dalam Usino et al. (2019) metode *K-Means Clustering* sebagai teknik *unsupervised learning* dimanfaatkan untuk membagi dokumen ke dalam beberapa klaster. Tujuan utamanya adalah agar dokumen-dokumen dengan tingkat kemiripan tinggi dapat ditempatkan pada satu klaster yang sama.

Penelitian ini menerapkan pendekatan *Spherical K-Means*, yaitu varian dari *K-Means* yang memanfaatkan *Cosine Similarity* untuk menghitung kedekatan antar dokumen. Pendekatan ini dipilih karena representasi fitur yang digunakan dalam penelitian ini bersifat vektor berdimensi tinggi yang lebih tepat diukur berdasarkan arah (orientasi) vektor dibandingkan jaraknya secara *absolut* (Mehsen & Joshi, 2024). Dengan *Cosine Similarity*, dua dokumen yang memiliki makna serupa akan membentuk sudut kecil antar vektor (nilai cos θ mendekati 1), meskipun magnitudo atau panjang vektornya berbeda.

Proses *clustering* diawali dengan normalisasi L2 pada matriks representasi fitur agar seluruh vektor memiliki panjang yang seragam, sehingga perbedaan arah antar vektor yang diukur oleh algoritma *K-Means* menghasilkan pengelompokan yang secara praktis setara dengan pengelompokan berbasis *Cosine Similarity* (Knittel et al.,

2021). Setiap dokumen atau file kemudian dialokasikan ke *cluster* yang memiliki jarak terdekat terhadap *centroid.* Setelah seluruh dokumen ter-*cluster,* posisi *centroid* diperbarui dengan menghitung rata-rata koordinat seluruh anggota *cluster.* Proses iteratif ini diulang hingga mencapai kondisi konvergen, yaitu ketika tidak ada dokumen yang berpindah *cluster* antar iterasi.

Penentuan jumlah *cluster* optimal dilakukan dengan mengevaluasi rentang nilai k menggunakan *Silhouette Score* sebagai metrik utama. Sejalan dengan yang dikemukakan oleh Januzaj et al. (2023) bahwa nilai *Silhouette Score* tertinggi mengindikasikan jumlah *cluster* yang paling optimal untuk data yang dianalisis, nilai k yang menghasilkan *Silhouette Score* tertinggi dipilih sebagai konfigurasi *final* dalam penelitian ini. *Clustering* diterapkan pada kedua *pipeline* analisis secara terpisah dengan konfigurasi yang disesuaikan, yaitu *pipeline* dokumen teks menggunakan matriks *hybrid* TF-IDF dan *FastText,* sedangkan *pipeline source code* menggunakan matriks TF-IDF berbasis token. Hasil *clustering* dimanfaatkan untuk membatasi perhitungan kemiripan hanya pada pasangan dokumen atau file yang berada dalam *cluster* yang sama, sehingga proses deteksi kemiripan menjadi lebih efisien secara komputasi tanpa mengorbankan akurasi hasil deteksi.

3.7.4 *Cosine Similarity*

Tahap selanjutnya adalah menghitung tingkat kemiripan antar dokumen menggunakan *Cosine Similarity,* yang diterapkan baik pada *pipeline* dokumen teks maupun *pipeline source code.* Setelah representasi vektor untuk masing-masing dokumen terbentuk, baik dari kombinasi TF-IDF dan *FastText* untuk dokumen teks maupun dari TF-IDF berbasis token untuk *source code,* setiap dokumen dipetakan ke dalam ruang vektor berdimensi tinggi yang siap digunakan untuk perhitungan kemiripan.

Untuk meningkatkan efisiensi, perhitungan *Cosine Similarity* dilakukan di dalam *cluster* hasil *K-Means*, bukan secara *pairwise* ke seluruh dokumen. Pendekatan *intra-cluster comparison* ini mengurangi kompleksitas perhitungan secara signifikan karena hanya membandingkan dokumen yang berada pada kelompok dengan pola fitur yang relatif serupa. Dengan demikian, proses deteksi kemiripan menjadi lebih cepat tanpa mengorbankan akurasi analisis.

Pada *pipeline* dokumen teks, *Cosine Similarity* dihitung dari vektor *hybrid* yang merupakan hasil penggabungan representasi TF-IDF dan FastText melalui *feature concatenation* berbobot, sehingga sistem mampu menangkap kesamaan kata secara statistik sekaligus kesamaan makna secara semantik dalam satu perhitungan. Pasangan dokumen dengan nilai *Cosine Similarity* di atas *threshold* 0,70 diklasifikasikan sebagai pasangan mirip. Pada *pipeline source code,* perhitungan kemiripan menggunakan pendekatan *dual scoring* yang menggabungkan TF-IDF *Cosine Similarity* dengan bobot 0,6 dan *Line-based Cosine Similarity* dengan bobot 0,4 untuk menghasilkan satu nilai kemiripan akhir yang disebut *final score*. Pasangan file dengan *final score* di atas *threshold* 0,80 diklasifikasikan sebagai pasangan mirip.

Melalui pendekatan ini, baik dokumen teks maupun *source code* dapat dianalisis secara konsisten dalam kerangka vektor yang sama-sama menggunakan *Cosine Similarity*. Hasil perhitungan kemiripan digunakan untuk menilai potensi plagiarisme, baik dalam bentuk *copy–paste* maupun modifikasi ringan. Kombinasi klasterisasi *K-Means* dan *Cosine Similarity* memungkinkan sistem melakukan deteksi secara lebih efisien, terstruktur, dan relevan terhadap karakteristik masing-masing jenis data.

3.7.5 Evaluasi Model

Tahap evaluasi model dilakukan untuk menilai sejauh mana metode yang diimplementasikan mampu mengidentifikasi kemiripan antar dokumen secara efektif dan efisien. Evaluasi ini bertujuan untuk memastikan bahwa hasil yang dihasilkan oleh model benar-benar mencerminkan tingkat kemiripan konten antar dokumen tugas mahasiswa serta mampu mengelompokkan dokumen dengan baik berdasarkan kesamaan topik atau isi. Karena penelitian ini menggunakan pendekatan *unsupervised learning*, maka proses evaluasi tidak dilakukan dengan membandingkan hasil prediksi terhadap label kebenaran *(ground truth)*, melainkan melalui pengukuran tingkat kemiripan, kualitas *cluster*, dan efisiensi proses pemrosesan dokumen.

1) Evaluasi kemiripan dokumen

   Evaluasi dilakukan dengan menganalisis nilai *Cosine Similarity* yang dihasilkan antara pasangan dokumen pada kedua *pipeline*. Nilai *Cosine Similarity* berkisar antara 0 hingga 1, di mana nilai mendekati 1 menunjukkan tingkat kemiripan yang tinggi, sedangkan nilai mendekati 0 menandakan perbedaan yang besar antar dokumen. Pada *pipeline* dokumen teks, pasangan dokumen yang terdeteksi memiliki nilai *Cosine Similarity* di atas *threshold* dibandingkan isi kontennya secara kualitatif. Pada *pipeline source code*, evaluasi kemiripan dilakukan terhadap pasangan file yang terdeteksi oleh pendekatan *dual scoring* untuk memvalidasi kesesuaian nilai *final score* dengan kemiripan kode yang sebenarnya.

2) Evaluasi kualitas *cluster*

   Evaluasi dilakukan terhadap hasil pengelompokan dokumen menggunakan algoritma K-Means pada kedua *pipeline*. Karena K-Means termasuk dalam kategori *unsupervised learning* yang tidak memiliki label kebenaran sebagai acuan, maka metrik

evaluasi yang digunakan adalah *Silhouette Score* untuk mengukur seberapa baik setiap dokumen berada dalam *cluster*-nya sendiri dibandingkan dengan *cluster* lain yang terdekat. Metrik ini dipilih karena mampu menggabungkan dua aspek kualitas *clustering* sekaligus, yaitu kohesi *(intra-cluster compactness)* dan separasi *(inter-cluster separation)* dalam satu nilai tunggal, sehingga cukup komprehensif untuk menilai sejauh mana K-Means mampu membentuk pengelompokan dokumen yang baik. Selain itu, stabilitas model terhadap variasi inisialisasi acak diuji dengan menjalankan model pada beberapa nilai *random state* yang berbeda, di mana konsistensi jumlah pasangan dokumen yang terdeteksi di atas *threshold* pada seluruh variasi digunakan sebagai indikator kestabilan model.

3) Evaluasi efisiensi pemrosesan

   Evaluasi dilakukan dengan membandingkan beban komputasi antara dua skenario pada masing-masing *pipeline*, yaitu perhitungan *Cosine Similarity* secara *all-pairs* tanpa *clustering* dan perhitungan secara *intra-cluster* dengan *clustering*. Perbandingan dilakukan dengan mengukur jumlah pasangan yang diperiksa, waktu eksekusi, dan *speedup* yang dihasilkan pada kedua *pipeline* secara terpisah. Hasil pengujian ini menjadi indikator sejauh mana penerapan *K-Means Clustering* berkontribusi dalam mengurangi beban komputasi tanpa menurunkan kualitas hasil deteksi kemiripan.

Dari keseluruhan proses evaluasi ini diharapkan diperoleh hasil bahwa kombinasi metode TF-IDF dan *FastText* mampu menghasilkan representasi dokumen yang baik untuk *pipeline* dokumen teks, sementara pendekatan *dual scoring* berbasis TF-IDF dan *Line-based Cosine Similarity* mampu menghasilkan deteksi kemiripan yang akurat untuk *pipeline source code*. Selain itu, algoritma *K-Means* diharapkan memberikan peningkatan efisiensi yang signifikan pada

kedua *pipeline*. Dengan demikian, model yang dibangun tidak hanya efektif dalam mendeteksi kemiripan, tetapi juga efisien ketika diintegrasikan ke dalam sistem web yang dapat digunakan dosen untuk memeriksa tugas mahasiswa secara otomatis.

3.7.6 Implementasi Sistem Web

Setelah tahapan pembuatan model selesai, langkah berikutnya adalah tahap pengembangan sistem web menggunakan *framework* Flask. Model yang telah dilatih, seperti *TF-IDF Vectorizer*, *Word2Vec Embedding*, dan *K-Means Clustering*, diekspor dan disimpan ke dalam *project Flask* untuk digunakan pada saat sistem menjalankan proses analisis kemiripan. *Flask b*erperan sebagai penghubung utama antara model deteksi kemiripan dengan antarmuka pengguna. Melalui sistem ini, dosen dapat mengunggah dokumen tugas mahasiswa yaitu format *.txt, .pdf*, dan *.docx* untuk dokumen teks dan format *.py* untuk file *source code* yang kemudian akan diproses oleh model untuk menghitung nilai kemiripan menggunakan *Cosine Similarity* serta menampilkan hasil klasterisasi dokumen secara otomatis.

Dari sisi arsitektur, *backend* Flask menangani seluruh alur kerja sistem, mulai dari menerima *request* pengguna, menjalankan proses *preprocessing* dokumen, mengeksekusi *pipeline* analisis kemiripan, hingga mengirimkan hasil analisis ke sisi *frontend* untuk ditampilkan kepada pengguna. *Frontend* sistem dibangun menggunakan HTML dan CSS dengan mekanisme *templating* Jinja2 yang memungkinkan halaman *web* ditampilkan secara dinamis berdasarkan hasil pemrosesan *backend*. Sistem tidak menggunakan *database* karena proses analisis dilakukan secara langsung terhadap file yang diunggah dan hasil analisis dihasilkan secara dinamis saat proses berlangsung. Sistem ini dirancang sebagai antarmuka sederhana dan dapat digunakan secara langsung dan cepat oleh dosen atau asisten pengajar di lingkungan Jurusan Ilmu Komputer Universitas Lampung. Dengan

pendekatan ini, penelitian tidak hanya menghasilkan model analisis kemiripan tugas mahasiswa, tetapi juga menyediakan platform sederhana yang mempermudah penerapan proses deteksi kemiripan dalam kegiatan akademik sehari-hari.

3.8 *Testing*

3.8.1 *White Box Testing*

White Box Testing digunakan dalam penelitian ini untuk memverifikasi kebenaran logika internal pada komponen – komponen utama sistem deteksi kemiripan, baik pada proses pemodelan teks maupun *source code.* Metode ini dipilih karena seluruh fungsi inti sistem mulai dari *preprocessing*, ekstraksi fitur TF-IDF/*Word2Vec,* dan perhitungan *Cosine Similarity,* hingga integrasi antarmuka *Flask* dapat diuji langsung melalui struktur kode program.

1) Unit Testing

Unit Testing diterapkan pada komponen–komponen terkecil dari sistem, seperti fungsi *preprocessing* teks, fungsi normalisasi *source code,* proses perhitungan TF-IDF, pemanggilan *embedding* pra-latih *Word2Vec,* serta fungsi perhitungan *Cosine Similarity*. Pengujian ini memastikan bahwa setiap fungsi dapat menerima *input,* menjalankan logika, dan menghasilkan *output* yang benar secara individual sebelum digunakan dalam proses analisis yang lebih besar.

2) *Integration Testing*

Integration Testing dilakukan untuk menguji hubungan antar modul dalam alur pemrosesan pendeteksian kemiripan. Tahap ini memverifikasi integrasi antara modul *preprocessing*, ekstraksi fitur, pembentukan vektor, klasterisasi, perhitungan *Cosine Similarity,* baik pada skenario dokumen teks maupun *source code.* Pengujian ini memastikan bahwa data dapat mengalir dengan benar di setiap

tahap, dan bahwa integrasi dua model analisis (teks & *cod*e) di dalam antarmuka *web Flask* berjalan tanpa kesalahan.

3) *System Testing*

System Testing dilakukan untuk menguji sistem secara keseluruhan, termasuk proses unggah dokumen*/source code,* eksekusi model teks dan model *source code,* perhitungan kemiripan, serta penampilan hasil dalam antarmuka *web Flask.* Pengujian ini bertujuan memastikan bahwa seluruh komponen mulai dari fungsi internal hingga integrasi antarmuka berjalan sesuai kebutuhan fungsional penelitian, dan bahwa sistem dapat digunakan secara langsung oleh dosen atau asisten pengajar tanpa kendala.

V. SIMPULAN DAN SARAN

5.1 Simpulan

Berdasarkan hasil implementasi, evaluasi, dan pengujian yang telah dilakukan, dapat diperoleh beberapa simpulan sebagai berikut:

1. Penerapan kombinasi metode TF-IDF dan Word2Vec (FastText) dengan algoritma *Cosine Similarity* berhasil mendeteksi kemiripan dokumen tugas mahasiswa secara efektif. Pendekatan *feature concatenation* terbukti lebih unggul dibandingkan penggunaan masing-masing metode secara terpisah, dengan berhasil mendeteksi 6 pasangan dokumen yang memiliki kemiripan signifikan dari dataset 90 dokumen. Selain itu, penerapan algoritma *K-Means Clustering* pada *pipeline* dokumen teks terbukti mereduksi jumlah pasangan yang diperiksa hingga 85,8% dibandingkan pendekatan *all-pairs*, tanpa mengorbankan akurasi hasil deteksi.

2. Pendekatan *dual scoring* pada analisis *source code* yang menggabungkan TF-IDF *Cosine Similarity* dan *Line-based Cosine Similarity*, menghasilkan penilaian kemiripan yang lebih representatif dibandingkan menggunakan satu metode saja, dengan berhasil mendeteksi 6 pasangan file Python yang memiliki kemiripan tinggi dari 117 file yang dianalisis. Penerapan algoritma *K-Means Clustering pada pipeline source code* juga terbukti mereduksi jumlah pasangan yang diperiksa hingga 83,7%, tanpa mengorbankan akurasi hasil deteksi.

3. Kedua model berhasil diintegrasikan dalam sistem berbasis Flask yang menyediakan modul SimDoc dan SimCode dalam satu sistem web yang terintegrasi. Hasil pengujian menunjukkan seluruh fungsi sistem berjalan

sesuai kebutuhan dan siap diimplementasikan sebagai alat bantu dosen maupun asisten pengajar di Jurusan Ilmu Komputer Universitas Lampung dalam memeriksa kemiripan dokumen tugas mahasiswa.

5.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan, terdapat beberapa saran yang dapat dipertimbangkan untuk pengembangan penelitian selanjutnya, yaitu sebagai berikut.

1. Penelitian selanjutnya dapat menggunakan jumlah dataset yang lebih besar serta berasal dari berbagai bidang penelitian agar sistem analisis kemiripan dokumen dapat diuji pada variasi konten dokumen yang lebih luas.

2. Metode representasi teks yang digunakan dalam penelitian ini dapat dikembangkan dengan memanfaatkan model bahasa berbasis *transformer* seperti IndoBERT yang dilatih khusus pada korpus bahasa Indonesia, atau LaBSE *(Language-agnostic BERT Sentence Embeddings)* yang dirancang untuk merepresentasikan teks lintas bahasa secara seragam. Penggunaan kedua model tersebut diharapkan dapat meningkatkan kemampuan sistem dalam memahami makna semantik dokumen yang ditulis dalam bahasa Indonesia maupun dokumen yang mengandung campuran istilah bahasa Inggris.

3. Penelitian selanjutnya dapat mengembangkan metode analisis kemiripan *source code* dengan menggunakan pendekatan yang lebih lanjut, seperti analisis struktur kode menggunakan *Abstract Syntax Tree* (AST) atau model pembelajaran mesin khusus kode seperti *CodeBERT* dan *GraphCodeBERT.* Pendekatan tersebut diharapkan dapat membantu sistem dalam menganalisis struktur kode dengan lebih baik sehingga mampu mendeteksi kemiripan pada berbagai bahasa pemrograman, tidak hanya terbatas pada *Python.*

4. Penelitian selanjutnya dapat melakukan pengujian performa sistem secara lebih komprehensif, khususnya melalui *performance testing* untuk mengevaluasi kemampuan sistem dalam menangani penggunaan secara bersamaan oleh banyak pengguna (*concurrent users*). Pengujian ini mencakup aspek seperti waktu respon sistem, stabilitas server, dan kemampuan sistem dalam memproses beberapa permintaan analisis secara bersamaan. Pengujian tersebut idealnya dilakukan setelah sistem di-*hosting* pada server yang sesungguhnya, sehingga hasil pengujian dapat mencerminkan kondisi penggunaan nyata di lingkungan produksi.

5. Penelitian selanjutnya dapat mengembangkan sistem dengan mengintegrasikan basis data (*database*) untuk menyimpan dokumen dan file *source code* yang telah diunggah oleh pengguna sebelumnya. Dengan adanya basis data, proses analisis kemiripan tidak hanya membandingkan file-file yang diunggah dalam satu sesi yang sama, tetapi juga dapat membandingkan file yang baru diunggah dengan seluruh file yang telah tersimpan di dalam korpus dari sesi-sesi sebelumnya. Pendekatan ini diharapkan dapat meningkatkan cakupan deteksi kemiripan secara signifikan karena sistem mampu mendeteksi kemiripan antar dokumen lintas waktu, sehingga lebih efektif dalam mengidentifikasi potensi kemiripan yang terjadi pada tugas-tugas yang dikumpulkan di periode yang berbeda.

DAFTAR PUSTAKA

Adam. (2025). Systematic Literature Review : Pemanfaatan Machine Learning dalam Berbagai Bidang. *Jurnal BATIRSI*, *9*(1), 19–22.

Adek, R. T., Dinata, R. K., & Ditha, A. (2022). Online Newspaper Clustering in Aceh using the Agglomerative Hierarchical Clustering Method. *International Journal of Engineering, Science and Information Technology*, *2*(1), 70–75. https://doi.org/10.52088/ijesty.v1i1.206

Andarsyah, R., & Yanuar, A. (2024). SENTIMEN ANALISIS APLIKASI POSAJA PADA GOOGLE PLAYSTORE UNTUK PENINGKATAN POSPAY SUPERAPP MENGGUNAKAN SUPPORT VECTOR MACHINE. *Jurnal Teknik Informatika*, *16*(2), 1–7.

Andriyani, W., Astuti, Y., Wisesa, B. A., & Hengki, H. (2025). Analisis Sentimen pada Ulasan Produk dengan SVM dan Word2Vec. *JIKO (Jurnal Informatika Dan Komputer)*, *8*(1), 173–185. https://doi.org/10.26798/jiko.v9i1.1498

Ansis, S., Listyaningsih, E. P., & Soetanto, H. (2024). Deteksi Plagiat Tesis Berbahasa Indonesia Menggunakan Metode Cosine Similarity. *INOVTEK Polbeng - Seri Informatika*, *9*(1), 153–167. https://doi.org/10.35314/isi.v9i1.4003

Aytekin, C., Ni, X., Cricri, F., & Aksu, E. (2018). Clustering and Unsupervised Anomaly Detection with l 2 Normalized Deep Auto-Encoder Representations. *2018 International Joint Conference on Neural Networks (IJCNN)*, 1–6. https://doi.org/10.1109/IJCNN.2018.8489068

Azhari, R. Y. (2022). Web Service Framework : flask dan fastAPI. *Technology and Informatics Insight Journal*, *1*(1), 80–87. https://doi.org/https://doi.org/10.32639/tiij.v1i1.54

Azmi, M. (2022). Analisis Tingkat Plagiasi Dokumen Skripsi Dengan Metode Cosine Similarity Dan Pembobotan Tf-Idf. *TEKNIMEDIA: Teknologi Informasi Dan Multimedia*, *2*(2), 90–95. https://doi.org/10.46764/teknimedia.v2i2.51

Balani, Z., & Varol, C. (2021). Combining Approximate String Matching Algorithms and Term Frequency In The Detection of Plagiarism. *International Journal of Computer Science and Security (IJCSS)*, *15*(4), 97–105. https://www.researchgate.net/profile/Zina-Balani/publication/365744521_Combining_Approximate_String_Matching_Algorithms_and_Term_Frequency_In_The_Detection_of_Plagiarism/links/6380e9c4c2cb154d292725d4/Combining-Approximate-String-Matching-Algorithms-and-T

Bili, N., Thimotius Abineno, R., & Arini Aha, P. (2024). Penerapan Algoritma K-Means Clustering Untuk Pengelompokkan Performa Siswa Pada Pembelajaran Bahasa Indonesia. *SATI: Sustainable Agricultural Technology Innovation*, *3*(1), 523–537.

Bota, Y. T., & Setiyawati, N. (2022). Pengembangan Sistem Informasi Perantara Bisnis Menggunakan Framework Flask. *Jurnal of Information Technology Ampera*, *3*(2), 79–93. https://journal-computing.org/index.php/journal-ita/index%0Aterlebih

Chowdhury, S., & Alzarrad, A. (2023). Applications of Text Mining in the Transportation Infrastructure Sector: A Review. *Information (Switzerland)*, *14*(4), 1–24. https://doi.org/10.3390/info14040201

Dani, A. H., & Puspaningrum, E. Y. (2024). Studi Performa TF-IDF dan Word2Vec Pada Analisis Sentimen Cyberbullying. *Router : Jurnal Teknik Informatika Dan Terapan*, *2*(2), 94–106. https://doi.org/10.62951/router.v2i2.76

Eka Putra, I. G. A., & Supriana, I. W. (2022). Deteksi Plagiarisme Source Code Tugas Mahasiswa Menggunakan Algoritma Cosine Similarity dan Pembobotan TF-IDF. *JNATIA : Jurnal Nasional Teknologi Informasi Dan Aplikasi*, *1*(November), 575–582.

Fachrurozi, A., Fitriana, L. A., Faddillah, U., & Sugiyarto, I. (2025). Implementasi Extreme Programming pada Pembuatan Website Sistem Informasi E-Accountant PT Naga Emas Internasional. *Remik: Riset Dan E-Jurnal Manajemen Informatika Komputer*, *9*(1), 73–87. https://doi.org/http://doi.org/10.33395/remik.v9i1.14294

Fatkhudin, A., Khambali, A., Artanto, F. A., Mundriyah, M., & Putra Zade, N. A. (2023). Implementasi Algoritma Clustering K-Means Dalam Pengelompokan Mahasiswa Studi Kasus (Prodi Manajemen Informatika). *Jurnal Minfo Polgan*, *12*(2), 777–783. https://doi.org/https://doi.org/10.33395/jmp.v12i2.12494 e-ISSN

Fitrani, L. D., & Puspitaningrum, A. C. (2023). Utilization of Unified Modeling Language (UML) in the Design of Academic Information Systems based on the OOAD Method. *SISTEMASI : Jurnal Sistem Informasi*, *12*(2), 614–625. http://sistemasi.ftik.unisi.ac.id

Fitriyah Salsabilah, A., Muzadid Zamzani, Z., & Gustrifa, W. (2024). Implementasi Data Mining Menggunakan Metode K-Means Clustering Untuk Analisis Tingkat Pengangguran Terbuka di Jawa Timur. *Jurnal Informatika Dan Sistem Informasi (JIFoSI)*, *5*(3), 1–12. https://doi.org/10.33005/jifosi.v5i3.460

Gandhi, N., Gopalan, K., & Prasad, P. (2024). A Support Vector Machine based approach for plagiarism detection in Python code submissions in undergraduate settings. *Frontiers in Computer Science*, *6*, 1–9.

Gustientiedina, G., Adiya, M. H., & Desnelita, Y. (2019). Penerapan Algoritma K-Means Untuk Clustering Data Obat-Obatan Pada RSUD Pekanbaru. *Jurnal Nasional Teknologi Dan Sistem Informasi*, *5*(1), 17–24. https://doi.org/10.25077/teknosi.v5i1.2019.17-24

Habib, T., Abdullah, D., & Rosnita, L. (2025). Plagiarism Detection Application for Computer Science Student Theses Using Cosine Similarity and Rabin-Karp. *International Journal of Engineering, Science, and Information Technology*, *5*(1), 185–194.https://doi.org/https://doi.org/10.52088/ijesty.v5i1.686

Halim, J., & Lasut, D. (2024). Document Plagiarism Detection Application Using Web-Based TF-IDF and Cosine Similarity Methods. *Jurnal Bit-Tech*, *7*(2), 202–213. https://doi.org/10.32877/bt.v7i2.1697

Helmi, R. R. Z., & Nuryasin, I. (2024). PENERAPAN WHITEBOX TESTING PADA PENGUJIAN SISTEM MENGGUNAKAN TEKNIK BASIS PATH. *JOISIE (Journal Of Information Systems And Informatics Engineering)*, *8*(1), 101–111. https://doi.org/https://doi.org/10.35145/joisie.v8i1.4229 JOISIE

Hidayat, M. A. R., Izzati, N. N., Prirhatama, A. R. P., Wijaya, Y. P., & Kurmilasari, S. (2025). PENGUJIAN PERANGKAT LUNAK PADA WEBSITE KA'CAKE IMPLEMENTASI UNIT TESTING, INTEGRATION TESTING, SYSTEM TESTING, DAN VALIDATION TESTING UNTUK MENJAMIN KUALITAS DAN KEANDALAN SISTEM. *JATI (Jurnal Mahasiswa Teknik Informatika)*, *9*(4).

Hnatkowska, B., & Mateusz, C. (2021). ACTIVITY DIAGRAM GENERATION BASED ON USE-CASE TEXTUAL SPECIFICATION Bogumila Hnatkowska , Mateusz Cebinka. *Computing and Informatics*, *40*, 772–795. https://doi.org/10.31577/cai 2021 4 772

Humaira, H., & Rasyidah, R. (2018). Determining The Appropiate Cluster Number Using Elbow Method for K-Means Algorithm. *Proceedings of the 2nd Workshop on Multidisciplinary and Applications (WMA) 2018*. https://doi.org/10.4108/eai.24-1-2018.2292388

Illahi, K. N., Suhartini, S., & Fajriyah, F. (2023). IMPLEMENTASI METODE EXTREME PROGRAMMING PADA SISTEM INFORMASI REPOSITORI SKRIPSI DI PERPUSTAKAAN UNIVERSITAS PRABUMULIH. *TEKNIMEDIA: Teknologi Informasi Dan Multimedia*, *4*(2), 182–189.

Irianto, M. R., Maududie, A., & Arifin, F. N. (2022). Implementation of K-Means Clustering Method for Trend Analysis of Thesis Topics (Case Study: Faculty of Computer Science, University of Jember). *Berkala Sainstek*, *10*(4), 210–226. https://doi.org/10.19184/bst.v10i4.29524

Iskandar, D., & Kurniawati, A. (2025). Analisis Perbandingan Teknik Word2vec

dan Doc2vec dalam Mengukur Kemiripan Dokumen Menggunakan Cosine Similarity. *Jurnal Teknologi Informasi Dan Ilmu Komputer*, *12*(1), 133–144. https://doi.org/10.25126/jtiik.2025129143

Januzaj, Y., Beqiri, E., & Luma, A. (2023). Determining the Optimal Number of Clusters using Silhouette Score as a Data Mining Technique. *International Journal of Online and Biomedical Engineering*, *19*(4), 174–182. https://doi.org/10.3991/ijoe.v19i04.37059 Ylber

Jonathan, R., & Setiyawati, N. (2023). IMPLEMENTASI FRAMEWORK FLASK PADA PEMBANGUNAN APLIKASI MONITORING DAN STORE VISIT IT SUPPORT PADA PT . XYZ. *Jurnal Informatika*, *23*(01), 91–101.

Kambey, G. E. I., Sengkey, R., & Jacobus, A. (2020). Penerapan Clustering pada Aplikasi Pendeteksi Kemiripan Dokumen Teks Bahasa Indonesia. *Jurnal Teknik Informatika*, *15*(2), 75–82.

Karnalim, O. (2020). TF-IDF Inspired Detection For Cross-Language Source Code Plagiarism and Collusion. *Computer Science*, *21*(1), 97–121.

Knittel, J., Koch, S., & Ertl, T. (2021). Efficient Sparse Spherical k-Means for Document Clustering. *DocEng : Proceedings of the 21st ACM Symposium on Document Engineering*, *6*, 1–4. https://doi.org/10.1145/3469096.3474937

Lan, F. (2022). Research on Text Similarity Measurement Hybrid Algorithm with Term Semantic Information and TF-IDF Method. *Advances in Multimedia*, *2022*(1), 1–11. https://doi.org/10.1155/2022/7923262

Londjo, M. F. (2021). IMPLEMENTASI WHITE BOX TESTING DENGAN TEKNIK BASIS PATH PADA PENGUJIAN FORM LOGIN. *Jurnal Siliwangi Seri Sain Dan Teknologi*, *7*(2), 35–40. https://doi.org/https://doi.org/10.37058/jssainstek.v7i2.4086

Lumbansiantar, S., Dwiasnati, S., & Fatonah, N. S. (2023). Penerapan Metode Cosine Similarity Dalam Mendeteksi Plagiarisme Pada Jurnal. *Jurnal Ilmiah Teknik Informatika*, *12*(2), 142–150. https://doi.org/10.22441/format.2023.v12.i2.007

Martinez-Gil, J. (2024). Advanced Detection of Source Code Clones via an Ensemble of Unsupervised Similarity Measures. *Software Quality as a Foundation for Security*, *505*. https://doi.org/10.1007/978-3-031-56281-5_2

Mehsen, R. S., & Joshi, H. D. (2024). Detection of Source Code Plagiarism Utilizing an Approach Based on Machine Learning. *International Journal of Computing*, *23*(1), 78–84. https://doi.org/10.47839/ijc.23.1.3438

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., & Joulin, A. (2017). *Advances in Pre-Training Distributed Word Representations*. *1*.

Molla, M. I., Ahmad, J., & Kadir, W. M. N. W. (2024). A Comparison of Transforming the User Stories and Functional Requirements into UML Use Case Diagram. *International Journal of Innovative Computing*, *14*(1), 29–36. https://doi.org/https://doi.org/10.11113/ijic.v14n1.463

Nilma, N. (2022). PENERAPAN MODEL SOFTWARE DEVELOPMENT LIFE CYCLE PADA RANCANG BANGUN SISTEM PAYROLL PERUSAHAAN. *JURNAL PUBLIKASI TEKNIK INFORMATIKA*, *1*(2), 61–69. https://doi.org/https://doi.org/10.55606/jupti.v1i2.334

Pawestri, S., & Suyanto, Y. (2024). Analisis Perbandingan Metode Similarity untuk Kemiripan Dokumen Bahasa Indonesia pada Deteksi Kemiripan Teks Bahasa Indonesia. *Jurnal Media Informatika Budidarma*, *8*(3), 1440. https://doi.org/10.30865/mib.v8i3.7648

Pratiwi, Y., & Widianti, L. W. (2025). IMPLEMENTASI WHITEBOX TESTING DENGAN TEKNIK BASIS PATH PADA PENGUJIAN HALAMAN PENCARIAN PROGRAM PROMO. *Jurnal Kecerdasan Buatan Dan Teknologi Informasi*, *4*(2), 173–180. https://doi.org/https://doi.org/10.69916/jkbti.v4i2.280

Rafli Aditya H, M., Ilham, M., Fatmarani Surianto, D., & Muis Mappalotteng, A. (2025). Evaluasi Pengukuran Semantik Sinonim KBBI Menggunakan Pendekatan Word Embedding. *Jurnal Nasional Teknik Elektro Dan Teknologi Informasi*, *14*(2), 112–120. https://doi.org/10.22146/jnteti.v14i2.17117

Rahman, M. F. F., Darussalam, K., Saphira, R. C., & Purwani, F. (2024). IMPLEMENTASI EXTREME PROGRAMMING DALAM PENGEMBANGAN APLIKASI MOBILE PENGENALAN ORGANISASI PADA MASA ORIENTASI MAHASISWA Fakultas Sains dan Teknologi Universitas Islam Negeri Raden Fatah Palembang. *Just IT : Jurnal Sistem Informasi, Teknologi Informasi Dan Komputer*, *14*(2), 128–132.

Ramdany, S. W., Kaidar, S. A., Aguchino, B., Putri, C. A. A., & Anggie, R. (2024). Penerapan UML Class Diagram dalam Perancangan Sistem Informasi Perpustakaan Berbasis Web. *Journal of Industrial and Engineering System (JIES)*, *5*(1), 30–41.

Resta, O. A., Aditya, A., & Purwiantono, F. E. (2021). Plagiarism Detection in Students' Theses Using The Cosine Similarity Method. *SinkrOn : Jurnal Dan Penelitian Teknik Informatika*, *5*(2), 305–313. https://doi.org/10.33395/sinkron.v5i2.10909

Riyani, A., Zidny Naf'an, M., & Burhanuddin, A. (2019). Penerapan Cosine Similarity dan Pembobotan TF-IDF untuk Mendeteksi Kemiripan Dokumen. *Jlk (Jurnal Linguistik Komputasional)*, *2*(1), 23–27.

Rustam, Z., & Fijri, A. L. (2020). Breast cancer clustering using modified spherical K-Means Zuherman. *Journal of Physics: Conference Series*, *1490*(1), 1–6. https://doi.org/10.1088/1742-6596/1490/1/012028

Saeed, A. A. M., & Taqa, A. Y. (2022). A proposed approach for plagiarism detection in Article documents. *SinkrOn : Jurnal Dan Penelitian Teknik Informatika*, *7*(2), 568–578. https://doi.org/10.33395/sinkron.v7i2.11381

Samad, A., Safi, M., & Djufri, I. (2024). IMPLEMENTASI MODEL SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC) PADA SISTEM INFORMASI ADMINISTRASI BERBASIS WEB KELURAHAN TOMALOU KOTA TIDORE KEPULAUAN. *IJIS : Indonesian Journal on Information System*, *9*(2), 242–254. https://doi.org/https://doi.org/10.36549/ijis.v9i2.341

Septiani, D., & Isabela, I. (2022). Term Frequency Inverse Document Frequency (Tf-Idf) Analysis in Information Retrieval in Text Documents. *Jurnal Sistem*

Dan Teknologi Informasi Indonesia(SINTESIA), *1*(2), 81–88.

Shutaywi, M., & Kachouie, N. N. (2021). Silhouette Analysis for Performance Evaluation in Machine. *Entropy*, *23*(6), 1–17. https://doi.org/10.3390/e23060759

Silalahi, E., Silalahi, D., Irani Tarigan, M., & Veronica Sinaga, R. (2024). Deteksi Plagiarisme Sebagai Peningkatan Integritas Akademik. *Kaizen: Jurnal Pengabdian Pada Masyarakat*, *3*(1), 27–33.

Sneha, S. G., Niha, S., & Hasan, D. M. M. (2024). Evaluating Code Clone Detection and Management : A Comprehensive Comparison among different Techniques and Tools along with some Effective Future Directions. *ICCA 2024: 3rd International Conference on Computing Advancements*, 207–215. https://doi.org/10.1145/3723178.3723206

Supriyatna, A. (2018). *METODE EXTREME PROGRAMMING PADA PEMBANGUNAN WEB APLIKASI SELEKSI PESERTA PELATIHAN KERJA*. *11*(1), 1–18.

Susanty, M., & Sahrul, S. (2021). Perbandingan Pre-trained Word Embedding dan Embedding Layer untuk Named-Entity Recognition Bahasa Indonesia. *PETIR: Jurnal Pengkajian Dan Penerapan Teknik Informatika*, *14*(2), 247–257. https://doi.org/https://doi.org/10.33322/petir.v14i2.1164

Usino, W., Prabuwono, A. S., Allehaibi, K. H. S., Bramantoro, A., Hasniaty, A., & Amaldi, W. (2019). Document similarity detection using K-Means and cosine distance. *International Journal of Advanced Computer Science and Applications*, *10*(2), 165–170. https://doi.org/10.14569/ijacsa.2019.0100222

Wahyu Kurniawan, F., & Maharani, D. W. (2020). Analisis Sentimen Twitter Bahasa Indonesia Menggunakan. *E-Proceeding of Engineering*, *7*(2), 7821–7829.

Wahyudi. (2025). Pemanfaatan Natural Language Processing (NLP) untuk Otomatisasi Penulisan Naskah Video pada Industri Kreatif Rumah Produksi. *INNOVATIVE: Journal Of Social Science Research*, *5*(3), 5568–5575.

Wajhillah, R., & Wibowo, A. (2022). Information Retrieval Pemetaan Peta Jalan Penelitian Perguruan Tinggi Berbasis Dokumen Publikasi Ilmiah Dosen. *JURNAL LARIK (Ladang Artikel Ilmu Komputer)*, *2*(2), 49–56.

Wan, Y., Xiong, Q., Qiu, Z., & Xie, Y. (2022). K-Means Clustering Algorithm Based on Memristive Chaotic System and Sparrow Search Algorithm. *Symmetry*, *14*(10), 1–14. https://doi.org/doi.org/10.3390/sym14102029

Wang, F., Xiang, X., Cheng, J., & Yuille, A. L. (2017). NormFace : L 2 Hypersphere Embedding for Face Verification. *Proceedings of the 25th ACM International Conference on Multimedia (MM '17)*, 1041–1049.

Yanti, I., & Utami, E. (2025). Analisis Sentimen Pemindahan Ibu Kota Indonesia Menggunak Word Embedding Word2Vec dan Metode Long Short-Term Memory. *Jurnal Teknik Informatika (Jutif)*, *6*(1), 149–158.

Yulisasih, B. N., Herman, H., Sunardi, S., & Yuliansyah, H. (2024). Evaluation of K-Means Clustering Using Silhouette Score Method on Customer Segmentation. *ILKOM Jurnal Ilmiah*, *16*(3), 330–342. https://doi.org/10.33096/ilkom.v16i3.2325.330-342