PENGEMBANGAN SISTEM PENDETEKSI KESAMAAN KONTEKS PADA DOKUMEN AKADEMIK MAHASISWA JURUSAN ILMU KOMPUTER UNIVERSITAS LAMPUNG

(SKRIPSI)

Oleh

Fitria Az Zahra
NPM 2217051052

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG
2026

PENGEMBANGAN SISTEM PENDETEKSI KESAMAAN KONTEKS PADA DOKUMEN AKADEMIK MAHASISWA JURUSAN ILMU KOMPUTER UNIVERSITAS LAMPUNG

Oleh

Fitria Az Zahra

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

ABSTRAK

PENGEMBANGAN SISTEM PENDETEKSI KESAMAAN KONTEKS PADA DOKUMEN AKADEMIK MAHASISWA JURUSAN ILMU KOMPUTER UNIVERSITAS LAMPUNG

Oleh

Fitria Az Zahra

Kemiripan dokumen dan plagiarisme masih menjadi permasalahan yang sering dijumpai dalam penyusunan tugas akhir mahasiswa, termasuk di Jurusan Ilmu Komputer Universitas Lampung. Sistem perpustakaan digital yang tersedia umumnya masih mengandalkan pencarian berbasis kata kunci sehingga kurang mampu mengidentifikasi kesamaan makna antardokumen. Untuk mengatasi permasalahan tersebut, penelitian ini mengembangkan SimTA, yaitu sistem deteksi kemiripan dokumen berbasis web yang dibangun menggunakan *framework* Flask. Sistem ini menerapkan pendekatan *hybrid scoring* yang menggabungkan analisis leksikal menggunakan TF-IDF dan analisis semantik menggunakan *Sentence*-BERT (SBERT) dengan model pra-latih paraphrase-multilingual-MiniLM-L12-v2. SimTA mendukung unggah dokumen dalam format PDF, DOCX, dan TXT serta menghitung tingkat kemiripan menggunakan *Cosine Similarity*. Evaluasi dilakukan terhadap 140 dokumen tugas akhir mahasiswa yang terdiri dari 30 dokumen D3 Manajemen Informatika dan 110 dokumen skripsi S1 Ilmu Komputer. Kinerja sistem dievaluasi menggunakan metrik *Precision, Recall,* dan *Mean Average Precision* (MAP). Hasil pengujian menunjukkan bahwa model *hybrid* memberikan performa terbaik dengan nilai *Precision* sebesar 90%, *Recall* sebesar 93,8%, dan MAP sebesar 91,5%, lebih tinggi dibandingkan model TF-IDF yang memperoleh MAP 85,5% dan model SBERT dengan MAP 80,9%. Penemuan ini menunjukkan bahwa kombinasi pendekatan berbasis kata dan makna mampu meningkatkan akurasi dalam mendeteksi kemiripan dokumen tugas akhir mahasiswa.

Kata Kunci: *Sentence*-BERT, TF-IDF, *Cosine Similarity*, Kemiripan Dokumen Pemrosesan Bahasa Alami.

ABSTRACT

DEVELOPMENT OF A CONTEXTUAL SIMILARITY DETECTION SYSTEM FOR STUDENT ACADEMIC DOCUMENTS IN THE DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITAS LAMPUNG

By

Fitria Az Zahra

Document similarity and plagiarism remain common challenges in the preparation of student final project documents, including in the Department of Computer Science at Universitas Lampung. Existing digital library systems generally rely on keyword-based retrieval, which limits their ability to effectively identify semantic similarities between documents. To address this issue, this study developed SimTA, a web-based document similarity detection system built using the Flask framework. The system employs a hybrid scoring approach that combines lexical analysis using TF-IDF and semantic analysis using Sentence-BERT (SBERT) with the pre-trained paraphrase-multilingual-MiniLM-L12-v2 model. SimTA supports document uploads in PDF, DOCX, and TXT formats and calculates similarity scores using Cosine Similarity. The evaluation was conducted on 140 student final project documents consisting of 30 final project reports from the D3 Informatics Management program and 110 undergraduate theses from the Computer Science program. System performance was evaluated using Precision, Recall, and Mean Average Precision (MAP). The experimental results show that the hybrid model achieved the best performance, with a Precision of 90%, Recall of 93.8%, and MAP of 91.5%, outperforming the TF-IDF model, which achieved a MAP of 85.5%, and the SBERT model, which obtained a MAP of 80.9%. These findings indicate that combining keyword-based and semantic-based approaches can improve the accuracy of detecting similarities in student final project documents.

Keywords: *Sentence-BERT;* TF-IDF*; Cosine Similarity; Document Similarity; Natural Language Processing*

Judul Skripsi	:	**PENGEMBANGAN SISTEM PENDETEKSI KESAMAAN KONTEKS PADA DOKUMEN AKADEMIK MAHASISWA JURUSAN ILMU KOMPUTER UNIVERSITAS LAMPUNG**
Nama Mahasiswa	:	Fitria Az Zahra
Nomor Pokok Mahasiswa	:	2217051052
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

Dwi Sakethi, S.Si., M.Kom.
NIP. 196806111998021001

3. Ketua Program Studi S1Ilmu Komputer

Tristiyanto, S.Kom., M.I.S., Ph.D.
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: **04 Juni 2026**

PERNYATAAN

Saya yang bertanda tangan di bawah ini.

Nama : Fitria Az Zahra

NPM : 2217051052

Dengan ini menyatakan bahwa skripsi saya yang berjudul **"Pengembangan Sistem Pendeteksi Kesamaan Konteks pada Dokumen Akademik Mahasiswa 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

Fitria Az Zahra

NPM. 2217051052

RIWAYAT HIDUP

Penulis bernama Fitria Az Zahra, lahir di Kalianda pada tanggal 22 Februari 2004. Penulis menempuh pendidikan formal pertama kali di TK Dharma Wanita Persatuan Depag Lampung Selatan. Setelah menyelesaikan pendidikan taman kanak-kanak, penulis melanjutkan ke jenjang sekolah dasar di SDN Bumi Agung dan lulus pada Tahun 2016. Penulis kemudian melanjutkan pendidikan ke jenjang menengah pertama di MTsN 1 Lampung Selatan dan lulus pada Tahun 2019 serta melanjutkan ke jenjang menengah atas di SMAN 1 Kalianda dan lulus pada Tahun 2022. Pada Agustus 2022, penulis resmi terdaftar sebagai mahasiswa Jurusan Ilmu Komputer FMIPA Unila melalui jalur SNMPTN.

Selama masa perkuliahan, penulis aktif dalam kegiatan akademik dan organisasi dengan pengalaman sebagai berikut:

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 2023/2024.
3. Menjadi Bendahara Bidang Kaderisasi Himpunan Mahasiswa Jurusan Ilmu Komputer pada periode 2024/2025.
4. Menjadi Asisten Dosen Jurusan Ilmu Komputer pada mata kuliah Rekayasa Perangkat Lunak Tahun 2024.
5. Menjadi Asisten Dosen Jurusan Ilmu Komputer pada mata kuliah Analisis dan Desain Sistem Informasi Tahun 2025.
6. Menjadi Sekertaris Koordinator Divisi Acara pada acara pekan Raya Jurusan Ilmu Komputer Tahun 2023.

7. Menjadi Koordinator Divisi Acara pada acara *Computer Science Showdown* Tahun 2024.
8. Mengikuti Studi Independen Bersertifikat (MSIB) Angkatan 7 *Bangkit Academy-Cohort Cloud Computing* Tahun 2024.
9. Melaksanakan Kerja Praktik di Dinas Komunikasi dan Informatika Kota Metro pada periode 2024/2025 dengan program kerja pengembangan *website* Metro Maps.
10. Kuliah Kerja Nyata (KKN) di Desa Mojokerto Kecamatan Padang Ratu Kabupaten Lampung Tengah (2025).

MOTTO

“Maka sesungguhnya bersama kesulitan ada kemudahan. Sesungguhnya bersama kesulitan ada kemudahan.”
(QS. Al-Insyirah: 5–6)

“Sebaik-baik manusia adalah yang paling bermanfaat bagi manusia.”
(H.R. Ahmad dan Ath Thabrani)

“Tiada kekayaan yang lebih utama daripada akal, tiada keadaan yang lebih menyedihkan daripada kebodohan, dan tiada warisan yang lebih baik daripada pendidikan.”
(Ali bin Abi Thalib)

PERSEMBAHAN

Alhamdulillahirobbilalamin

Segala puji dan syukur senantiasa penulis haturkan kehadirat Allah SWT, Dzat Yang Maha Kuasa atas segala sesuatu, yang atas limpahan rahmat, karunia, serta ridho-Nya lah sehingga skripsi ini akhirnya dapat terselesaikan. Tanpa pertolongan-Nya, tiada satu pun langkah dalam perjalanan panjang ini yang mampu penulis tempuh. Shalawat beserta salam semoga senantiasa tercurahkan kepada baginda Nabi Muhammad SAW, manusia pilihan yang menjadi teladan bagi seluruh umat manusia hingga akhir zaman, yang syafaatnya senantiasa penulis rindukan.

Kupersembahkan skripsiku ini kepada

Kedua Orang Tuaku dan Keluarga Tercinta

Kepada Ibu penulis yang telah lebih dahulu dipanggil menghadap-Nya, semoga Allah SWT melapangkan dan memuliakan tempatmu di sisi-Nya. Kepada Ayah, kakak, adik, dan seluruh keluarga besar, terima kasih atas setiap doa dan dukungan yang tanpa sadar telah menguatkan setiap langkah penulis hingga sampai di titik ini. Semoga Allah SWT senantiasa melindungi dan merahmati kita semua. Aamiin.

Teman-Temanku Tersayang

Terima kasih atas setiap dukungan dan tawa yang selalu hadir di saat penulis membutuhkan.

Seluruh Keluarga Besar Ilmu Komputer 2022

Terima kasih atas setiap kebersamaan yang telah mewarnai masa studi penulis. Semoga kita dipertemukan kembali dalam kesuksesan masing-masing.

Almamater Tercinta Jurusan Ilmu Komputer FMIPA Unila

Terima kasih telah menjadi tempat penulis menimba ilmu dan membentuk diri.

SANWACANA

Puji syukur senantiasa penulis haturkan kehadirat Allah Subhanahu Wa Ta'ala, atas limpahan rahmat dan karunia-Nya sehingga penulis diberikan kekuatan dan kemampuan untuk menyelesaikan skripsi ini. Shalawat beserta salam semoga senantiasa tercurahkan kepada baginda Nabi Muhammad SAW, suri teladan bagi seluruh umat manusia hingga akhir zaman. Atas izin dan ridha-Nya, penulis akhirnya dapat menyelesaikan skripsi yang berjudul **"Pengembangan Sistem Pendeteksi Kesamaan Konteks pada Dokumen Akademik Mahasiswa Jurusan Ilmu Komputer Universitas Lampung"** sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer pada Jurusan Ilmu Komputer FMIPA Unila. Penulis menyadari sepenuhnya bahwa terselesaikannya skripsi ini bukan semata-mata hasil kerja keras penulis sendiri, melainkan tidak terlepas dari bantuan, doa, dan dukungan berbagai pihak yang telah memberikan kontribusi besar dalam perjalanan panjang ini. Oleh karena itu, dengan penuh ketulusan penulis menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Almarhumah Ibu penulis, skripsi ini penulis persembahkan sebagai bentuk cinta, rindu dan terima kasih atas segala kasih sayang serta pengorbanan yang telah diberikan. Ayah penulis terima kasih atas untaian doa yang tidak pernah terputus, dukungan moril maupun materiil serta segala bentuk pengorbanan yang telah diberikan.
2. Bapak Dr. Eng. Heri Satria, S.Si., M.Si. selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung.
3. Bapak Dwi Sakethi, S.Si., M.Kom. selaku Ketua Jurusan Ilmu Komputer FMIPA Universitas Lampung.
4. Bapak Dr. rer. nat. Akmal Junaidi, M.Sc., selaku Pembimbing Utama yang telah penuh kesabaran, meluangkan waktu serta pikiran, untuk membimbing dan mengarahkan penulis selama penyusunan skripsi ini.

5. Bapak Muhammad Galih Ramaputra, S.Kom., M.T.I. selaku Pembimbing Pembantu yang selalu meluangkan waktu untuk berdiskusi dan memberikan solusi ketika penulis menghadapi kesulitan, sehingga setiap proses dalam penyusunan skripsi ini dapat terlewati dengan lebih baik.
6. Bapak Tristiyanto, S.Kom., M.I.S., Ph.D. selaku Dosen Penguji yang telah memberikan kritik dan saran yang sangat membangun, sehingga skripsi ini menjadi lebih baik dari sebelumnya.
7. Seluruh Dosen Jurusan Ilmu Komputer yang telah memberikan ilmu pengetahuan, wawasan dan bimbingan akademik selama penulis menempuh masa perkuliahan. Semoga setiap ilmu yang telah diberikan menjadi amal jariyah serta membawa keberkahan bagi Bapak dan Ibu Dosen sekalian.
8. Seluruh staf dan karyawan Jurusan Ilmu Komputer khususnya Ibu Ade Nora Maela, Pak Dahud dan Mas Syam yang senantiasa memberikan bantuan dan kemudahan kepada penulis dalam mengurus berbagai keperluan administrasi selama menempuh masa perkuliahan.
9. Kakak dan adik penulis terima kasih atas doa, dukungan dan semangat yang selalu diberikan kepada penulis. Kehadiran kalian menjadi salah satu penyemangat yang membuat penulis terus berjuang dan melangkah maju hingga dapat menyelesaikan skripsi ini.
10. Almarhum Ayahde Udin, Ayahde Yumi, Almarhum Ayahde Saleh, Mba Rika, Mamahde Nur, Teh Nadya, A Ova serta seluruh keluarga besar. Terima kasih atas doa, dukungan dan bantuan yang diberikan kepada penulis selama proses pendidikan. Semua doa dan kebaikan yang diberikan menjadi salah satu alasan penulis dapat bertahan hingga sejauh ini.
11. Teman-teman seperjuangan penulis Apaya Kr Kr (Oca, Afina, Ayu, Nadya, Kezia) dan Deta. Terima kasih telah hadir dan menemani perjalanan perkuliahan ini dengan segala bantuan, dukungan dan kebersamaan yang berarti. Terima kasih telah menjadi ruang untuk tumbuh, saling menguatkan dan berbagi. Semoga kita semua dapat meraih impian serta kesuksesan di jalan masing-masing.

12. Theresia Tri Oktavia Irmawanti, partner skripsi sekaligus tempat berkeluh kesah dan bertukar pikiran selama proses penyusunan skripsi ini. Terima kasih telah menemani setiap proses, berbagi ide serta saling menyemangati dalam menghadapi berbagai kesulitan selama penyusunan skripsi.
13. Puteri Mutiara Dita, sahabat sejak bangku SD yang sudah tumbuh bersama, yang mengenal penulis jauh sebelum skripsi ini ada. Terima kasih sudah menjadi tempat yang tak pernah lelah mendengar setiap cerita, keluh kesah serta lika-liku perjalanan hidup penulis. Semoga kita terus bertumbuh bersama dan persahabatan ini terus terjaga sampai kapanpun.
14. Teman-teman sejak kecil penulis Anak Alim, yakni Puja, Nesa, Ani, Puteri yang telah menjadi bagian dari saksi perjalanan hidup penulis sejak kecil hingga saat ini. Terima kasih atas doa, kebersamaan, cerita dan hubungan baik yang tetap terjaga meskipun dipisahkan oleh jarak dan kesibukan masing-masing.
15. Mutiara Utami, rekan dekat sejak SMA yang selalu menjadi "911" setiap kali penulis membutuhkan bantuan. Terima kasih sudah mau mendengar cerita yang itu-itu saja, memberikan dukungan dan tidak pernah bosan dengan saling bully kecil-kecilan yang justru membuat kedekatan kita bertahan hingga hari ini. Semoga persahabatan ini terus terjaga dan semua hal baik selalu menghampirimu.
16. Teman-teman Pimpinan Himakom 2024 (RT-17) dan Bidang Kaderisasi Periode 2024. Terima kasih atas kebersamaan dan pengalaman selama menjalani kegiatan organisasi bersama. Berbagai proses, diskusi dan momen yang dilalui bersama memberikan banyak pelajaran dan pengalaman berharga bagi penulis.
17. Eca, Riri, Putri, Atir, Eric dan Rizky, keluarga KKN yang dipertemukan dalam satu desa dan satu tujuan yang sama. Terima kasih telah menjadikan masa KKN penulis penuh warna, kebersamaan dan banyak pengalaman berharga. Berbagai cerita, suka duka serta canda tawa yang dilalui bersama menjadi kenangan yang berkesan bagi penulis. Semoga kalian selalu diberikan kesehatan, kelancaran dan kesuksesan kedepannya.

18. Berliana Efendi, teman kos yang telah menjadi bagian dalam perjalanan penulis selama masa perkuliahan. Tidak terhitung berapa kali penulis datang meminta bantuan. Terima kasih sudah menjadi tempat bercerita dan memberi semangat dihari-hari yang tidak selalu mudah.
19. Atha dan Dinda selaku rekan sejak SMA penulis, terima kasih atas semangat yang selalu diberikan. Meskipun jarang bertemu, komunikasi dan kedekatan yang terjalin tetap terjaga dengan baik. Semoga pertemanan ini dapat terus berlanjut serta menghadirkan hal-hal baik dimasa mendatang.
20. Teman-teman PPS selaku rekan sejak SMA, yakni Ntul, Triana, Abir, Wapa, Mumi, Pika, Imam, Nando, Candu, Bagus dan yang lainnya. Terima kasih atas doa, dukungan dan kebersamaan yang telah menemani penulis sejak masa SMA hingga saat ini. Semoga hubungan baik dan kebersamaan ini tetap terjaga hingga nanti.
21. Seluruh keluarga besar Ilkomp 22, terima kasih atas kebersamaan dan proses yang telah dilewati dari awal perkuliahan hingga sampai pada tahap ini. Setiap momen, cerita dan kenangan yang tercipta selama masa perkuliahan akan menjadi bagian yang membekas dalam perjalanan hidup penulis.

Penulis menyadari bahwa skripsi ini masih memiliki berbagai keterbatasan dan belum sepenuhnya sempurna. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan sebagai bahan perbaikan di masa mendatang. Penulis berharap skripsi ini dapat memberikan manfaat serta menambah wawasan dan pengetahuan bagi para pembaca.

Bandar Lampung, 08 Juni 2026
Penulis,

Fitria Az Zahra
NPM. 2217051052

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Gambar **Halaman**

DAFTAR TABEL

Tabel **Halaman**

I. PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi telah memengaruhi berbagai aspek kehidupan, termasuk dalam proses pengarsipan dokumen di berbagai instansi (Wahyuni *et al.,* 2017). Di perguruan tinggi, kegiatan akademik menghasilkan banyak dokumen ilmiah yang perlu diarsipkan dan dikelola dengan baik. Manajemen dokumen digital berperan penting dalam menjaga keamanan data, meningkatkan efisiensi, serta mengoptimalkan penggunaan ruang penyimpanan (Farozi *et al.,* 2023). Secara nasional, perguruan tinggi di Indonesia menghasilkan ribuan karya tugas akhir setiap tahunnya. Data dari Pangkalan Data Pendidikan Tinggi (PDDikti) Kementerian Pendidikan Tinggi, Sains, dan Teknologi tahun 2024 menunjukkan bahwa Indonesia memiliki 4.416 perguruan tinggi dengan 33.741 program studi, 303.067 dosen, dan 9.967.487 mahasiswa (Kemdiktisaintek, 2024). Dengan jumlah mahasiswa yang besar ini, banyaknya skripsi yang dihasilkan, sehingga berpotensi menimbulkan kesamaan dan kemiripan antar dokumen skripsi (Arsyad *et al.,* 2024). Kondisi ini meningkatkan dua tantangan utama. Pertama risiko *plagiarisme*, yaitu tindakan menyalin sebagian atau seluruh karya orang lain, baik secara sengaja maupun tidak, tanpa mencantumkan sumber yang sah, sehingga karya tersebut tampak seolah-olah sebagai hasil pribadi (Fadhullah *et al.,* 2022). Kedua, kesulitan dalam pencarian referensi relevan karena mahasiswa harus menelusuri ratusan hingga ribuan dokumen untuk menemukan penelitian dengan topik atau konteks yang serupa.

Deteksi kesamaan dokumen (*document similarity detection*) dan *plagiarism detection* menjadi fokus penelitian aktif di bidang *Natural Language*

Processing (NLP) dan *Information Retrieval* (IR). Salah satu pendekatan yang sering digunakan adalah *text mining*, yaitu proses untuk mendapatkan informasi berkualitas tinggi dari data teks dengan cara mengekstrak informasi dari berbagai sumber (Deolika & Taufiq Luthfi, 2019). *Text mining* termasuk salah satu cabang dari *data mining* yang khusus digunakan untuk menganalisis data dalam bentuk dokumen teks (Rachman *et al.,* 2021). Metode konvensional yang umum digunakan adalah kombinasi *Term Frequency Inverse Document Frequency* (TF-IDF) dan *cosine similarity*. TF-IDF adalah teknik yang digunakan untuk menilai tingkat keterkaitan suatu kata (*term*) terhadap sebuah dokumen dengan cara memberikan bobot pada setiap kata (Deolika & Taufiq Luthfi, 2019). Sementara itu, *cosine similarity* adalah ukuran kesamaan yang menghitung sudut antara vektor kueri dengan vektor dokumen (Pertiwi & Taufiqurrochman, 2017).

Penelitian terbaru menunjukkan kinerja yang baik pada berbagai tugas pemahaman bahasa alami telah dicapai dengan menggabungkan informasi sintaksis dan semantik dimasukkan secara langsung ke dalam model pra-latih seperti BERT dan RoBERTa. Namun, pendekatan ini bergantung pada *fine-tuning* yang spesifik untuk setiap permasalahan, dan seperti banyak dilaporkan, model-model berbasis BERT menunjukkan performa yang lemah serta kurang efisien ketika digunakan pada tugas perbandingan kesamaan secara *unsupervised* (Peng *et al.,* 2021). Menyadari keterbatasan ini, Reimers & Gurevych (2019) mengembangkan SBERT (*Sentence*-BERT), yaitu modifikasi dari model BERT yang menggunakan *arsitektur siamese* dan *triplet networks* untuk menghasilkan *sentence embeddings* yang bermakna secara semantik. Artinya, SBERT mampu memetakan kalimat-kalimat yang memiliki makna mirip ke posisi yang berdekatan dalam ruang vektor. Dengan kemampuan ini, BERT dapat digunakan untuk tugas-tugas baru yang sebelumnya tidak efektif dilakukan dengan BERT standar. Tugas tersebut meliputi perbandingan kesamaan semantik dalam skala besar, *clustering*, dan pencarian informasi berbasis makna (*semantic search*). Model ini juga

mengurangi waktu komputasi secara drastis untuk menemukan pasangan dokumen yang paling mirip, dari 65 jam menggunakan BERT/RoBERTa standar menjadi hanya sekitar 5 detik dengan SBERT, sambil tetap mempertahankan tingkat akurasi dari BERT (Reimers & Gurevych, 2019).

Meskipun berbagai metode deteksi kesamaan dokumen telah dikembangkan, penerapannya di lingkungan akademik, khususnya di perguruan tinggi Indonesia, masih terbatas. Jurusan Ilmu Komputer Universitas Lampung belum memiliki sistem terintegrasi yang mampu secara otomatis mendeteksi kemiripan konteks dokumen akademik mahasiswa. Sistem yang ada saat ini (Digilib) hanya menyediakan pencarian berbasis kata kunci manual, yang memiliki beberapa kelemahan. Pertama, tidak dapat mendeteksi kesamaan semantik, sehingga dokumen dengan topik serupa tetapi menggunakan istilah berbeda tidak teridentifikasi. Kedua, potensi pengulangan topik penelitian, di mana mahasiswa mungkin tidak mengetahui bahwa topik yang diajukan sudah pernah diteliti dengan pendekatan serupa. Keterbatasan ini menunjukkan perlunya sistem yang tidak hanya mengandalkan pencocokan kata kunci, tetapi juga mampu memahami konteks dan makna semantik dari dokumen akademik.

Berdasarkan permasalahan tersebut, penelitian ini difokuskan pada pengembangan sistem berbasis web untuk pendeteksi kesamaan konteks dokumen akademik mahasiswa dengan pendekatan *Natural Language Processing* (NLP). Sistem memanfaatkan model *Sentence*-BERT (SBERT) untuk menghasilkan representasi vektor semantik dari kalimat atau dokumen, sehingga diharapkan mampu mengukur kesamaan konteks secara lebih akurat dibandingkan metode kata kunci konvensional. Dengan kemampuan ini, sistem diharapkan dapat mengidentifikasi dokumen dengan tingkat kesamaan tertinggi, membantu dosen memantau orisinalitas karya ilmiah, mempercepat pencarian referensi relevan bagi mahasiswa, dan mencegah pengulangan topik penelitian.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan sebelumnya, maka rumusan masalah yang menjadi fokus dalam penelitian ini adalah sebagai berikut:

1. Bagaimana mengembangkan sistem berbasis web untuk menganalisis kesamaan konteks dokumen akademik mahasiswa Jurusan Ilmu Komputer Universitas Lampung menggunakan metode TF-IDF dan *Sentence*-BERT (SBERT)?
2. Bagaimana kinerja metode TF-IDF dan SBERT dalam mendeteksi kesamaan konteks dokumen akademik mahasiswa Jurusan Ilmu Komputer Universitas Lampung berdasarkan metrik *precision, recall,* dan *Mean Average precision* (MAP)?

1.3 Batasan Masalah

Adapun batasan masalah dari penelitian ini adalah sebagai berikut:

1. Penelitian ini menggunakan dokumen tugas akhir mahasiswa Jurusan Ilmu Komputer Universitas Lampung pada Bab 1–3, yang diperoleh melalui situs Digilib Unila, sebagai dataset utama.
2. Data yang digunakan dalam penelitian ini terbatas pada dokumen tugas akhir mahasiswa dari tahun 2021-2025.
3. Penelitian difokuskan pada analisis kesamaan teks, tidak mencakup elemen non-teks seperti tabel, gambar, atau grafik.
4. Sistem yang dibangun dibatasi hanya dapat memproses *input* berupa teks dan file dokumen dengan format PDF, TXT dan Word (.docx).

1.4 Tujuan

Adapun tujuan dari penelitian ini adalah sebagai berikut:

1. Mengembangkan sistem berbasis web untuk menganalisis kesamaan konteks dokumen akademik mahasiswa terhadap dataset utama menggunakan metode TF-IDF dan *Sentence*-BERT (SBERT).
2. Mengevaluasi kinerja metode TF-IDF dan SBERT dalam mendeteksi kesamaan konteks dokumen menggunakan metrik *precision*, *recall*, dan *Mean Average precision* (MAP).

1.5 Manfaat

Adapun manfaat yang diharapkan dari penelitian ini adalah sebagai berikut:

1. Penelitian ini diharapkan dapat mendukung Jurusan Ilmu Komputer Universitas Lampung dalam pengelolaan dokumen tugas akhir secara sistematis dan terpusat.
2. Penelitian ini diharapkan dapat mempermudah dosen dalam memberikan arahan dan bimbingan yang lebih tepat sasaran, dengan akses cepat ke referensi tugas akhir yang sesuai dengan bidang riset mahasiswa.
3. Penelitian ini diharapkan dapat mempermudah pencarian referensi dokumen tugas akhir dan menghindari pengulangan topik penelitian.

II. TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penelitian ini disusun berdasarkan landasan dari berbagai penelitian sebelumnya, sehingga terdapat persamaan dan perbedaan pada objek yang diteliti. Tabel 1 menampilkan ringkasan penelitian terdahulu yang memiliki relevansi dengan penelitian ini.

Tabel 1. Penelitian Terdahulu yang Terkait.

No	Penelitian	Metode	Hasil
1	Penerapan Algoritma *Cosine Similarity* dan Pembobotan TF-IDF pada Sistem Skripsi (Wahyuni *et al.,* 2017).	TF-IDF dan *Cosine Similarity* untuk klasifikasi otomatis dokumen skripsi	Pengujian *black-box* dan oleh pakar menunjukkan sistem layak digunakan (88,3% & 87,5%), dengan tingkat ketepatan klasifikasi 98%. Kesalahan terjadi akibat kata kunci tumpang tindih, menunjukkan perlunya *term* unik yang lebih banyak.
2	*Sentence*-BERT: *Sentence Embeddins using Siamese* BERT-*Network* (Reimers & Gurevych, 2019).	SBERT dengan *arsitektur siamese* dan *triplet network* untuk menghasilkan *sentence embeddings* yang bermakna secara semantik.	Mengurangi upaya untuk menemukan pasangan dokumen yang paling mirip dari 65 jam dengan BERT/RoBERTa menjadi sekitar 5 detik dengan SBERT, sambil mempertahankan akurasi dari BERT.

No	Penelitian	Metode	Hasil
3	Penerapan *Cosine Similarity* dan Pembobotan TF-IDF untuk Mendeteksi Kemiripan Dokumen (Riyani *et al.,* 2019).	*Cosine Similarity* dan pembobotan TF-IDF dengan tahapan *Preprocessing* (*Case folding*, *Tokenizing*, *Stopword Removal*, dan *Stemming*).	Skenario dengan *Stemming* menghasilkan nilai kemiripan rata-rata lebih tinggi 10% daripada tanpa *stemming*. Menghasilkan nilai similaritas di atas 50% untuk dokumen dengan tingkat kemiripan tinggi, dan di bawah 40% untuk dokumen dengan tingkat kemiripan rendah atau tidak berplagiat.
4	Aplikasi Deteksi Dini *Plagiarisme* Penelitian Ilmiah Menggunakan Algoritma *Cosine Similarity* Berbasis Web (Fadhullah *et al.,* 2022).	*Cosine Similarity* dengan normalisasi untuk deteksi *plagiarisme* berbasis web	Aplikasi berhasil mendeteksi tingkat *plagiarisme* pada tingkat sedang (31,18%) dan berjalan dengan baik. Penelitian ini menyarankan pengembangan metode klasifikasi yang lebih akurat serta peningkatan efisiensi perhitungan.
5	Deteksi Kemiripan Dokumen Menggunakan *Cosine Similarity* Berdasarkan Representasi Teks *Count Vectorizer* dan TF-IDF (Pradana *et al.,* 2024).	*Cosine Similarity* dengan dua pendekatan representasi teks: TF-IDF dan *Count Vectorizer*	Deteksi kemiripan dengan TF-IDF mendapatkan nilai kesamaan rata-rata 7,72861, sedangkan *Count Vectorizer* mendapatkan nilai rata-rata 16,85541. Gap antara kedua metode sebesar 9,1268. *Count vectorizer* menunjukkan performa lebih tinggi karena menghitung frekuensi kata tanpa mempertimbangkan kelangkaan kata, sehingga kata-kata umum tetap berkontribusi penuh terhadap *similarity*.

Penelitian Tabel 1 mengacu pada penelitian- penelitian sebelumnya yang dijelskan sebagai berikut.

2.1.1 Penerapan Algoritma *Cosine Similarity* dan Pembobotan TF-IDF pada Sistem Klasifikasi Dokumen Skripsi

Penelitian ini dilakukan oleh Wahyuni *et al.,* (2017) dengan hasil pengujian menunjukkan sistem berjalan baik dan layak digunakan. Pengujian *black-box* menunjukkan seluruh fungsi berjalan dengan baik, sementara pengujian kelayakan oleh pakar sistem dan pakar kearsipan memperoleh persentase rata-rata 88,3% dan 87,5%. Dari 50 dokumen yang diolah, 49 dokumen berhasil diklasifikasikan dengan benar, menghasilkan tingkat ketepatan klasifikasi sebesar 98%. Kesalahan klasifikasi sebagian besar terjadi karena adanya kata kunci yang tumpang tindih antar kategori, menunjukkan pentingnya penggunaan *term* unik dan perluasan perbendaharaan kata untuk meningkatkan akurasi sistem.

2.1.2 *Sentence*-BERT: *Sentence Embedding s using* Siamese BERT-Networks

Penelitian ini dilakukan oleh Reimers & Gurevych, (2019) yang menyatakan bahwa BERT *out-of-the-box* kurang cocok untuk mengukur kesamaan kalimat menggunakan ukuran seperti *Cosine Similarity*, dengan kinerja pada tujuh tugas STS lebih rendah dibanding *embedding* GloVe. Untuk mengatasinya, dikembangkan *Sentence*-BERT (SBERT) dengan *fine-tuning* menggunakan *arsitektur siamese* atau *triplet network*, yang menunjukkan peningkatan signifikan dibanding metode *embedding* kalimat mutakhir. SBERT juga lebih cepat dalam proses komputasi misalnya, *clustering* 10.000 kalimat yang memerlukan 65 jam dengan BERT

dapat diselesaikan hanya dalam 5 detik menggunakan SBERT, sementara tetap mempertahankan akurasi tinggi.

2.1.3 Penerapan *Cosine Similarity* dan Pembobotan TF-IDF untuk Mendeteksi Kemiripan Dokumen

Penelitian ini dilakukan oleh Riyani *et al.,* (2019) menggunakan algoritma *cosine similarity* dan TF-IDF untuk mendeteksi kemiripan teks dan nilai *plagiarisme* pada dokumen skripsi mahasiswa. Hasil menunjukkan bahwa *preprocessing* seperti *tokenizing, stopword removal, case folding,* dan *stemming* meningkatkan nilai kemiripan rata-rata sekitar 10%. Selain itu, penggunaan *stemming* membuat jumlah dokumen yang menghasilkan nilai kemiripan di atas 50% meningkat, sementara dokumen dengan nilai kemiripan rendah (<40%) tetap tidak terdeteksi sebagai *plagiarisme*.

2.1.4 Aplikasi Deteksi Dini *Plagiarisme* Penelitian Ilmiah Menggunakan Algoritma *Cosine Similarity* Berbasis Web

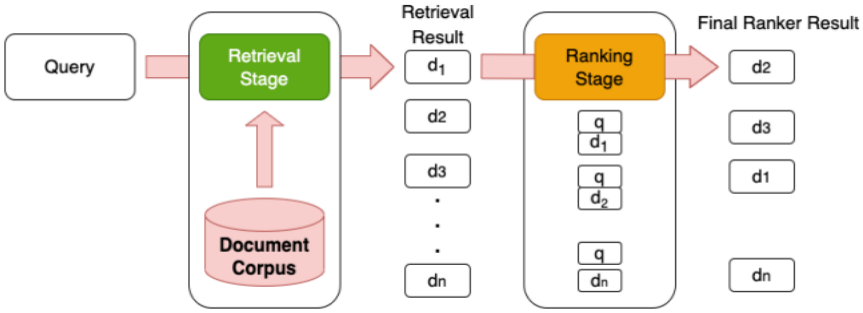
Penelitian ini dilakukan oleh Fadhullah *et al.,* (2022) hasil pengujian menunjukkan bahwa nilai *plagiarisme* secara keseluruhan berada pada tingkat sedang, yaitu sebesar 31,18% dengan menggunakan normalisasi yang diterapkan dalam penelitian ini. Hasil tersebut menunjukkan bahwa aplikasi berjalan dengan baik dan dapat dijadikan acuan untuk menentukan tingkat *plagiarisme* pada jurnal lain, dengan catatan bahwa data latih yang digunakan sudah cukup memadai untuk perhitungan yang dilakukan. Penelitian ini masih membuka peluang untuk inovasi lebih lanjut. Beberapa saran yang dapat menjadi fokus penelitian berikutnya antara lain: penggunaan metode klasifikasi lain untuk memperoleh hasil deteksi yang lebih akurat, serta peningkatan efisiensi perhitungan agar proses analisis menjadi lebih cepat dan mudah dipahami.

2.1.5 Deteksi Kemiripan Dokumen Menggunakan *Cosine Similarity* Berdasarkan Representasi Teks *Count Vectorizer* dan TF-IDF

Penelitian ini dilakukan oleh Pradana *et al.,* (2024) menggunakan algoritma *cosine similarity* dengan dua pendekatan representasi teks, yaitu TF-IDF dan *count vectorizer*, pada 1.600 dokumen abstrak skripsi mahasiswa, dengan 30 dokumen digunakan untuk pengujian. Hasil menunjukkan bahwa *count vectorizer* unggul dengan rata-rata kemiripan 16,85541 dibanding TF-IDF sebesar 7,72861, dengan selisih 9,1268, karena *count vectorizer* menghitung frekuensi kata tanpa membedakan kata umum atau jarang, sehingga kata-kata umum tetap berkontribusi penuh terhadap nilai *similarity*.

2.2 *Information Retrieval System* (IRS)

Sistem Temu Kembali Informasi (*Information Retrieval*/IR) adalah proses pencarian materi, biasanya berupa dokumen, dari data yang tidak terstruktur, sehingga dapat memenuhi kebutuhan informasi dari suatu kumpulan koleksi yang tersimpan dalam komputer. Secara umum, sistem temu-balik informasi digunakan untuk secara otomatis menemukan kembali informasi yang relevan dengan kebutuhan pengguna dari suatu kumpulan data atau dokumen (Christioko & Daru, 2018). Gambar 1 menyajikan tinjauan umum dari *Information Retrieval System*, yang memperlihatkan komponen utama serta hubungan antarbagian dalam mendukung proses temu kembali informasi.



Gambar 1. Tinjauan Umum *Information Retrieval System (K. A. Hambarde & Proença, 2023).*

Gambar 1 menunjukkan alur kerja umum dari *Information Retrieval System* modern, yang berfungsi untuk menemukan dan mengurutkan dokumen berdasarkan tingkat relevansinya terhadap sebuah *query*. Proses dimulai ketika pengguna memasukkan sebuah *query* sebagai masukan sistem. *Query* tersebut kemudian diproses pada *Retrieval Stage*, yaitu tahap yang bertugas mengambil sekumpulan dokumen kandidat dari *document corpus*, yaitu kumpulan dokumen yang tersimpan dalam basis data. Tahap ini menghasilkan daftar awal dokumen yang diperkirakan relevan, seperti d1, d2, sampai dn. Selanjutnya, daftar dokumen tersebut dikirim ke *ranking stage*, di mana setiap dokumen dipasangkan dengan *query* (misalnya pasangan q–d1, q–d2, hingga q–dn) untuk dinilai tingkat relevansinya menggunakan model ranking yang lebih mendalam. Hasil proses ini adalah urutan akhir dokumen dari yang paling relevan hingga yang kurang relevan, yang kemudian disajikan kepada pengguna sebagai *final ranker result*.

Untuk memberikan pemahaman yang lebih jelas tentang struktur dan komponen yang membentuk sistem tersebut, penjelasan berikut menguraikan elemen-elemen utama dari sebuah sistem temu kembali informasi. Sistem temu kembali informasi terdiri atas tiga elemen utama. Pertama, masukan (*input*), yaitu bagian yang berisi *query* atau kata kunci yang dimasukkan oleh pengguna sebagai permintaan untuk menemukan informasi yang relevan.

Kedua, pemroses (*processor*), yang berperan dalam mengolah *query* tersebut dengan melakukan pemrosesan lanjutan seperti analisis bahasa alami, pengindeksan dokumen, perhitungan tingkat relevansi, serta penerapan strategi pencarian yang sesuai. Ketiga, keluaran (*output*), yaitu hasil dari proses pencarian yang menampilkan dokumen atau informasi yang sesuai dengan *query*, baik dalam bentuk daftar dokumen, ringkasan, maupun format lain yang membantu pengguna memperoleh informasi yang dibutuhkan dengan lebih mudah (Septiani & Isabela, 2022).

2.3 Tugas Akhir

Tugas akhir (TA) adalah salah satu persyaratan yang harus dipenuhi mahasiswa untuk menyelesaikan jenjang pendidikan formal di perguruan tinggi (Haviluddin *et al.,* 2021). Pada jenjang Diploma (D3), salah satu bentuk tugas akhir adalah laporan akhir, sedangkan pada jenjang Sarjana (S1) bentuk tugas akhirnya biasanya berupa skripsi**.** Skripsi adalah karya ilmiah yang menjadi salah satu syarat untuk memperoleh gelar sarjana (S1). Secara umum, skripsi bertujuan untuk menjawab pertanyaan penelitian. Pertanyaan utama tersebut dirumuskan melalui hipotesis penelitian, sehingga tujuan khusus dari skripsi adalah untuk menguji dan memberikan jawaban terhadap hipotesis yang telah dibuat. Proses penyusunan skripsi melibatkan beberapa tahap pengujian, sehingga hasilnya dapat dianalisis secara logis dan disajikan secara sistematis dalam bentuk tulisan ilmiah (Yunita & Kamayani, 2023).

Struktur skripsi umumnya terdiri dari beberapa bab, yaitu: Bab 1 Pendahuluan yang memuat latar belakang, rumusan masalah, tujuan, dan manfaat penelitian; Bab 2 Tinjauan Pustaka yang berisi kajian teori dan penelitian terdahulu; Bab 3 Metodologi Penelitian yang menjelaskan metode dan langkah-langkah penelitian; Bab 4 Hasil dan Pembahasan yang menyajikan hasil penelitian dan analisisnya; serta Bab 5 Penutup yang berisi kesimpulan dan saran.

2.4 *Plagiarisme*

Plagiarisme atau plagiat merupakan tindakan mengambil atau menggunakan ide maupun karya orang lain tanpa mencantumkan sumber aslinya, kemudian mengakuinya sebagai hasil karya sendiri. Perbuatan ini termasuk pelanggaran dan dilarang dalam lingkungan akademik. Fenomena yang terjadi belakangan ini menunjukkan bahwa kasus *plagiarisme* banyak ditemukan di kalangan perguruan tinggi. Untuk memastikan bahwa karya ilmiah terbebas dari unsur *plagiarisme*, dapat dilakukan pemeriksaan menggunakan aplikasi atau perangkat lunak pendeteksi tingkat kesamaan tulisan (Silalahi *et al.,* 2024).

Plagiarisme dalam penulisan karya ilmiah dapat dikategorikan ke dalam tiga bentuk utama. Pertama, plagiat langsung, yaitu tindakan penulis menyalin atau mengutip sumber secara utuh kata demi kata tanpa mencantumkan nama penulis atau pemilik sumber asli. Dalam hal ini, pelaku plagiat secara sadar mengakui gagasan orang lain sebagai hasil pemikirannya sendiri. Kedua, plagiat karena kesalahan atau ketidakjelasan kutipan, yang terjadi ketika penulis tidak menuliskan sumber rujukan dengan tepat, sehingga menimbulkan kesan bahwa ide tersebut merupakan hasil pribadi. Ketiga, plagiat mozaik, yaitu bentuk *plagiarisme* di mana penulis mengutip dengan benar tetapi mengganti sebagian kata atau kalimat dengan kata-katanya sendiri tanpa memberikan pengakuan kepada penulis asli. Akibatnya, meskipun susunan kalimatnya berbeda, gagasan yang disampaikan tetap serupa dengan sumber aslinya, namun tidak ditulis dalam format kutipan yang semestinya (Sukowati & Suciptaningsih, 2024).

2.5 *Text Mining*

Text mining adalah metode analisis yang diterapkan pada data atau teks yang berasal dari kumpulan dokumen yang bersifat semi-terstruktur atau tidak memiliki struktur sama sekali. Berbeda dengan *data mining*, *text mining* khusus memproses data berupa teks, sehingga fokusnya adalah mengekstrak

informasi dan pola dari dokumen teks tersebut (Nur Azizah *et al.,* 2024). Agar proses analisis dapat berjalan dengan baik, teks perlu dipersiapkan terlebih dahulu melalui tahap *text preprocessing*.

Pra-pemrosesan teks (*text preprocessing*) merupakan tahap yang sangat penting dalam penerapan dan teknik *text mining*. Tahapan ini menjadi langkah awal dalam proses pengolahan teks, yang berfungsi untuk menyiapkan data agar dapat diolah oleh algoritma analisis teks selanjutnya. Dalam penelitian ini, tahapan pra-pemrosesan yang digunakan meliputi empat langkah utama, yaitu *case folding*, *tokenizing*, *stopword removal*, dan *stemming* (Alfan *et al.,* 2020). Gambaran umum mengenai tahapan pra-pemrosesan tersebut ditunjukkan pada Gambar 2, yang memperlihatkan alur proses transformasi teks sebelum masuk ke tahap analisis lebih lanjut.

Gambar 2. Pra-pemrosesan teks (Alfan *et al.,* 2017).

1. *Case folding*

   Pada tahap ini, semua huruf dalam dokumen teks diubah menjadi huruf kecil (*lowercase*). Tujuannya adalah untuk menyeragamkan penulisan kata dan menghindari perbedaan makna akibat perbedaan penggunaan huruf besar atau kecil. Sebagai contoh, kata “Wifi” akan diubah menjadi “wifi”.
2. *Tokenizing*

   Proses *tokenizing* dilakukan dengan memecah dokumen teks menjadi potongan-potongan kata yang disebut *token*. Tahapan ini penting untuk mengidentifikasi setiap kata secara terpisah agar dapat dianalisis lebih lanjut oleh sistem.

3. *Stopword Removal*

   Tahapan ini bertujuan untuk menghapus kata-kata umum yang tidak memiliki makna penting dalam konteks analisis, seperti kata sambung, kata ganti, dan preposisi. Contohnya, kata seperti "yang", "dan", "atau", "hingga" akan dihapus dari dokumen karena tidak dianggap sebagai kata kunci dalam analisis teks.

4. *Stemming*

   Tahap *stemming* berfungsi untuk mengubah kata berimbuhan menjadi bentuk dasarnya (*root word*). Misalnya, kata "berlari" akan diubah menjadi "lari". Tahapan ini sangat penting karena hasil *Stemming* yang baik akan meningkatkan akurasi proses analisis teks secara keseluruhan.

2.6 *Machine Learning*

Machine Learning (ML) merupakan cabang dari *Artificial Intelligence* (AI) yang memungkinkan komputer untuk "berpikir" dan belajar secara mandiri. Inti dari *machine learning* adalah membuat sistem mampu menyesuaikan perilakunya untuk meningkatkan kinerja, di mana peningkatan tersebut diukur berdasarkan frekuensi keberhasilan tindakan yang dilakukan (Alzubi *et al.,* 2018). Metode *machine learning* pada dasarnya memungkinkan komputer atau mesin untuk belajar secara otomatis dari data yang diberikan, baik berupa data mentah maupun dataset yang telah disiapkan. Secara umum, proses ini terdiri dari dua tahapan utama, yaitu (Pandey *et al.,* 2019):

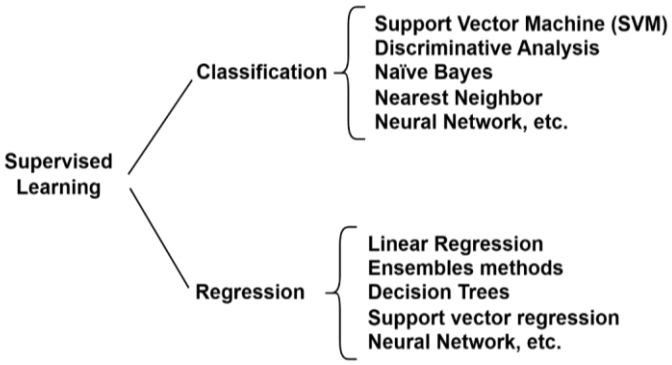
(1) Pelatihan model, dan
(2) Pengambilan keputusan atau pengujian model.

Untuk mendukung proses tersebut, digunakan dua jenis dataset, yaitu *training data* dan *test data*. *Training data* dipakai untuk melatih model, yang biasanya mencakup tahapan pra-pemrosesan data dan ekstraksi fitur sebelum proses pelatihan dilakukan. Model yang sudah terlatih kemudian digunakan untuk

memprediksi hasil pada data yang belum pernah dilihat sebelumnya, yaitu *test data*. Selain itu, tingkat akurasi model dapat dievaluasi dengan membandingkan hasil prediksi terhadap test data yang nilai sebenarnya sudah diketahui (Pandey *et al.,* 2019).

Secara umum, metode dalam *Machine Learning* dapat dikelompokkan ke dalam tiga kategori utama, yaitu *Supervised Learning* (Pembelajaran Terbimbing), *Unsupervised Learning* (Pembelajaran Tanpa Pengawasan) dan *Reinforcement Learning* (Pembelajaran Penguatan) (Pandey *et al.,* 2019).

Supervised Learning merupakan algoritma yang, sebagaimana namanya, bekerja dengan menggunakan pengawasan dalam proses pembelajaran untuk menghasilkan prediksi atau keputusan. Pada pendekatan ini, dataset dibagi menjadi dua bagian, yaitu data pelatihan (*training data*) dan data pengujian (test data). Ciri utama dari pembelajaran terbimbing adalah tersedianya nilai target atau *output* pada data pelatihan, yang digunakan sebagai dasar untuk melatih model agar dapat mengenali pola secara akurat. Metode ini merupakan salah satu teknik yang paling umum dan efektif digunakan untuk menyelesaikan dua jenis permasalahan utama, yaitu klasifikasi (*classification*) dan regresi (*regression*). Pendekatan klasifikasi diterapkan pada permasalahan dengan nilai *output* bersifat diskrit, sedangkan regresi digunakan untuk pengambilan keputusan pada permasalahan dengan nilai *output* kontinu (Pandey *et al.,* 2019). Contoh ilustrasi mengenai proses klasifikasi pada pendekatan supervised learning ditunjukkan pada Gambar 3, yang menggambarkan bagaimana model memetakan data ke dalam kelas tertentu berdasarkan pola yang dipelajari dari data pelatihan.



Gambar 3. Klasifikasi dalam *Classification in Supervised Learning* (Pandey *et al.,* 2019).

Gambar 3 menunjukkan pembagian metode dalam *supervised learning* yang terdiri dari dua cabang utama, yaitu *classification* dan *regression*. Cabang pertama adalah *classification*, yang digunakan ketika *output* yang ingin diprediksi berupa kategori atau kelas tertentu. Beberapa metode yang termasuk dalam klasifikasi antara lain *Support Vector Machine (SVM), Discriminative Analysis, Naïve Bayes, Nearest Neighbor*, dan *Neural Network.* Metode-metode ini bekerja dengan cara mempelajari pola dari data berlabel untuk kemudian mengelompokkan data baru ke dalam kelas yang sesuai. Cabang kedua adalah *Regression*, yang digunakan ketika output yang ingin diprediksi berupa nilai numerik yang kontinu. Metode yang termasuk dalam kategori ini antara lain *Linear Regression, Ensembles Methods, Decision Trees, Support Vector Regression*, dan *Neural Network*. Dalam konteks penelitian ini, pendekatan yang digunakan termasuk dalam cabang klasifikasi, di mana sistem dirancang untuk mengelompokkan dokumen akademik ke dalam kategori topik penelitian yang sesuai berdasarkan kesamaan konteks teksnya.

Sementara *unsupervised learning* merupakan kelas *machine learning* yang beroperasi pada *dataset* tanpa label, di mana tujuannya adalah untuk mengungkap struktur atau pengaturan tersembunyi dalam data. Dalam proses ini, model dilatih menggunakan fitur-fitur dari data yang tidak berlabel, dan

hasil pelatihan tersebut kemudian digunakan untuk mengelompokkan data baru atau data uji. Secara umum, metode ini sangat fokus pada penyelesaian masalah pengelompokan (*clustering*) mengatur data menjadi *cluster* berbeda dan reduksi dimensi. Beberapa algoritma paling populer yang digunakan dalam pembelajaran tanpa pengawasan meliputi *K-means clustering*, *Principal Component Analysis* (PCA), serta berbagai model seperti Hierarchical Model dan Hidden Markov Model (Pandey *et al.,* 2019).

Reinforcement Learning merupakan pendekatan pembelajaran yang bersifat dinamis dan banyak diterapkan pada bidang robotika, otomatisasi, permainan (gaming), serta sistem navigasi. Metode ini bekerja melalui mekanisme *trial and error*, di mana sebuah agen belajar berinteraksi dengan lingkungan untuk mencapai tujuan tertentu tanpa diberikan informasi eksplisit mengenai seberapa dekat ia dengan target. Agen membentuk perilaku berdasarkan *input* yang diterima serta perubahan keadaan (*state transition*) dari kondisi sebelumnya, kemudian memilih tindakan yang menghasilkan umpan balik (*feedback*) paling positif atau paling mendekati tujuan. Pendekatan ini dapat dipandang sebagai perpaduan antara *supervised* dan *unsupervised learning*: proses belajar dimulai dengan tindakan acak, lalu diperbaiki secara bertahap melalui umpan balik. Secara khusus, pembelajaran penguatan berlandaskan pada dua mekanisme utama, yaitu proses pencarian melalui *trial and error* serta hasil yang bersifat tertunda (*delayed outcome*), di mana konsekuensi suatu tindakan baru terlihat setelah beberapa langkah ke depan (Pandey *et al.,* 2019).

2.7 *Natural Language Processing*

Pemrosesan Bahasa Alami (*Natural Language Processing*/NLP) merupakan cabang penting dari kecerdasan buatan (AI) yang berfokus pada kemampuan mesin untuk memahami, menafsirkan, dan menghasilkan bahasa manusia.

NLP bertujuan menjembatani perbedaan antara cara manusia berkomunikasi dengan cara komputer memproses teks dan ucapan agar dapat dipahami secara bermakna. Penerapan NLP sangat luas, termasuk pada asisten virtual, moderasi konten media sosial, analisis sentimen pelanggan, dan layanan penerjemahan bahasa (Sateesh Kumar Rongali, 2025).

Teknologi dasar NLP untuk bahasa Indonesia mencakup berbagai metode dan pendekatan. Dua contoh utama adalah:

2.7.1 *Stemming*

Metode ini bertujuan untuk mengubah kata ke bentuk dasarnya dengan cara menghilangkan imbuhan, termasuk awalan, sisipan, dan akhiran. *Stemming* merupakan langkah penting dalam banyak aplikasi NLP, seperti pencarian informasi dan analisis sentimen, karena membantu menyederhanakan data teks dan meningkatkan efisiensi pemrosesan (Amien, 2023).

2.7.2 *Part-of-Speech (POS) Tagging*

Sistem POS digunakan untuk mengidentifikasi kategori gramatikal setiap kata dalam teks, seperti kata benda, kata kerja, kata sifat, dan lainnya. Informasi ini sangat berguna dalam berbagai tugas NLP, termasuk parsing, analisis sintaksis, dan pemahaman teks (Amien, 2023).

2.8 *Term Frequency-Inverse Document Frequency* (TF-IDF)

Metode *Term Frequency-Inverse Document Frequency* (TF-IDF) merupakan salah satu teknik yang digunakan dalam pemrosesan teks dan pemodelan bahasa alami. Tujuan utamanya adalah menilai tingkat kepentingan suatu kata (*term*) dalam sebuah dokumen dibandingkan dengan keseluruhan yang ada (Septiani & Isabela, 2022). Metode TF-IDF (*Term Frequency–Inverse*

Document Frequency) mempertimbangkan dua faktor utama dalam menentukan tingkat kepentingan suatu kata dalam dokumen.

Term Frequency (TF) berfungsi untuk mengukur seberapa sering suatu kata muncul dalam sebuah dokumen. Umumnya, TF dihitung dengan membagi jumlah kemunculan kata tersebut dengan total jumlah kata dalam dokumen. Dalam beberapa kasus, perhitungan TF dapat dimodifikasi menggunakan skema pembobotan yang lebih kompleks untuk memperoleh hasil yang lebih akurat (Septiani & Isabela, 2022). Sedangkan, *Inverse Document Frequency* (IDF) digunakan untuk menilai seberapa penting suatu kata dalam keseluruhan kumpulan dokumen. Kata yang jarang muncul di seluruh koleksi dokumen akan memiliki nilai IDF yang lebih tinggi. Perhitungannya dilakukan dengan membagi jumlah total dokumen dengan jumlah dokumen yang memuat kata tersebut, kemudian hasilnya diubah menggunakan logaritma untuk menyesuaikan skala nilai (Septiani & Isabela, 2022).

Dalam penerapannya, nilai TF dan IDF dikalikan untuk menghasilkan bobot kata (*term* weight) pada setiap kata dalam dokumen. Nilai bobot ini menunjukkan tingkat kepentingan kata tersebut di dalam satu dokumen dibandingkan dengan keseluruhan kumpulan dokumen yang ada (Septiani & Isabela, 2022).

Rumus Metode *Term Frequency-Inverse Document Frequency* (TF-IDF) (Wibowo *et al.,* 2024). Perhitungan *Term Frequency* (TF) pada dokumen teks diasumsikan bahwa setiap dokumen memiliki tingkat kepentingan yang sama. Dengan kata lain, setiap kemunculan kata dalam suatu dokumen dianggap memiliki kontribusi yang sebanding terhadap analisis. Adapun rumus yang digunakan untuk menghitung nilai TF dapat dilihat pada persamaan berikut.

$$\mathrm{TF}\,(\mathrm{d}, \mathrm{t}) = \mathrm{f}\,(\mathrm{d}, \mathrm{t}) \qquad (1)$$

Pada rumus persamaan 1, f merepresentasikan *frekuensi kemunculan* suatu kata dalam sebuah dokumen. Nilai ini digunakan untuk mengetahui seberapa

sering kata tersebut muncul dalam teks. Selanjutnya, t merupakan *term*, yaitu frasa atau kata tertentu yang dianalisis dalam dokumen. Sementara itu, d adalah *dokumen* tempat term tersebut ditemukan.

Selanjutnya, apabila suatu *term* jarang muncul dalam kumpulan dokumen, maka nilai *Inverse Document Frequency* (IDF) akan semakin tinggi. Hal ini menunjukkan bahwa *term* tersebut dianggap lebih spesifik dan memiliki tingkat kepentingan yang lebih besar. Rumus untuk menghitung nilai IDF dapat dilihat pada persamaan (2) berikut.

$$\text{IDF} = \text{LOG}\ (\frac{D}{dft}) \qquad (2)$$

Pada rumus persamaan 2, D adalah jumlah keseluruhan dokumen dalam korpus yang digunakan sebagai acuan. Nilai ini menjadi dasar untuk mengetahui seberapa luas distribusi sebuah kata dalam kumpulan dokumen. Sementara itu, df(t) adalah jumlah dokumen yang memuat *term* (t) atau kata yang sedang dianalisis. Semakin sedikit dokumen yang mengandung term tersebut, semakin tinggi nilai IDF-nya, sehingga term dianggap lebih penting atau lebih mampu membedakan isi dokumen.

Selanjutnya, dilakukan perhitungan bobot kata dalam dokumen yang diperoleh dari hasil kombinasi antara nilai *Term Frequency* (TF) dan *Inverse Document Frequency* (IDF). Perhitungan ini menghasilkan nilai bobot yang mencerminkan tingkat kepentingan suatu kata dalam dokumen. Rumus perhitungannya dapat dilihat pada persamaan (3) berikut.

$$\text{W (d, t)} = \text{TF x IDF} \qquad (3)$$

Pada metode TF-IDF, W merupakan bobot yang diberikan pada setiap dokumen yang memuat *keyword* atau kata kunci tertentu. Bobot ini dihitung berdasarkan dua komponen utama, yaitu TF (*Term Frequency*) dan IDF (*Inverse Document Frequency*). TF menggambarkan seberapa sering sebuah

term muncul dalam suatu dokumen, sedangkan IDF menunjukkan seberapa penting term tersebut di dalam keseluruhan korpus dokumen. Kombinasi TF dan IDF menghasilkan bobot W yang mencerminkan tingkat relevansi term tersebut terhadap dokumen, di mana nilai yang lebih tinggi menunjukkan bahwa term tersebut memiliki kontribusi informasi yang lebih signifikan.

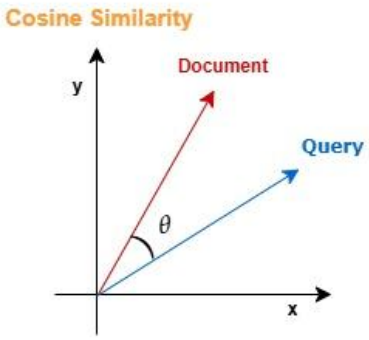
2.9 *Cosine Similarity*

Perhitungan *cosine similarity* merupakan salah satu komponen utama yang banyak dimanfaatkan dalam berbagai aplikasi *data mining*. Secara umum, metode ini bekerja dengan menghitung sudut antara dua objek (misalnya Dokumen 1 dan Dokumen 2) yang direpresentasikan dalam bentuk dua vektor, dengan menggunakan kata kunci (*keywords*) dari masing-masing dokumen sebagai acuannya (Lumbansiantar *et al.,* 2023).

Ukuran tingkat kemiripan antara dua vektor dalam suatu ruang berdimensi dapat diperoleh melalui nilai *cosine similarity* , yaitu nilai kosinus dari sudut antara kedua vektor tersebut. Nilai *cosine similarity* menunjukkan hubungan antara dua vektor berdasarkan arah, bukan besarannya. Karena nilai kosinus sudut berkisar antara 0 hingga 1 pada ruang positif, maka semakin kecil sudut antara dua vektor, semakin besar nilai *cosine similarity* -nya. Jika nilai *cosine similarity* = 1, maka kedua vektor dianggap sangat mirip (arahnya sama), sedangkan nilai yang mendekati 0 menunjukkan bahwa kedua vektor tidak mirip (Amalia *et al.,* 2021) .

Dalam konteks ruang berdimensi positif, nilai *cosine similarity* dibatasi antara 0 dan 1. Nilai 1 menunjukkan kemiripan penuh, sedangkan nilai mendekati 0 menunjukkan ketidaksamaan. Konsep ini umum digunakan pada ruang berdimensi tinggi, seperti pada bidang *Information Retrieval* (IR). Dalam kasus tersebut, setiap kata atau istilah (*term*) dianggap sebagai satu dimensi, dan setiap dokumen direpresentasikan dalam bentuk vektor, di mana setiap

dimensi berisi frekuensi kemunculan suatu istilah dalam dokumen (Amalia *et al.,* 2021). Ilustrasi mengenai konsep perhitungan *cosine similarity* tersebut dapat dilihat pada Gambar 4, yang menunjukkan hubungan sudut antara dua vektor untuk menentukan tingkat kemiripan antar dokumen.



Gambar 4. *Cosine Similarity* (Lumbansiantar *et al.,* 2023).

Gambar 4 menggambarkan konsep *cosine similarity* secara visual melalui representasi ruang vektor dua dimensi. Pada ilustrasi tersebut ditampilkan dua vektor, yaitu vektor *document* (panah merah) dan vektor *query* (panah biru), yang keduanya berawal dari titik origin (0,0) pada sumbu koordinat x dan y. Sudut yang terbentuk di antara kedua vektor tersebut dinyatakan dengan simbol θ (theta). *Cosine Similarity* mengukur tingkat kedekatan dua vektor berdasarkan besar kecilnya sudut θ tersebut. Semakin kecil sudut yang terbentuk, nilai cosinus akan semakin mendekati 1, yang menunjukkan bahwa kedua dokumen memiliki tingkat kemiripan yang tinggi. Sebaliknya, jika sudut semakin besar hingga mendekati 90 derajat, nilai cosinus akan mendekati 0, yang mengindikasikan bahwa kedua dokumen tidak memiliki kemiripan.

Dalam penelitian ini, *cosine similarity* dimanfaatkan untuk mengukur tingkat kemiripan antar dokumen akademik yang telah direpresentasikan dalam bentuk vektor, baik menggunakan metode TF-IDF maupun SBERT. Metode ini dipilih karena tidak bergantung pada panjang dokumen, sehingga memungkinkan perbandingan yang lebih objektif meskipun terdapat perbedaan jumlah kata pada setiap dokumen.

Berikut ini merupakan persamaan *cosine similarity* dalam menghitung kemiripan antar dua buah vektor (Amalia *et al.,* 2021).

$$\text{Sim} = \text{Cos}\,(\theta) = \frac{A \bullet B}{\|A\|\,\|B\|} = \frac{\sum_{j=1}^{t}(A_i x\, B_i)}{\sqrt{\sum_{i=1}^{n}(A_i)^2}\; x \sqrt{\sum_{i=1}^{n}(B_i)^2}} \quad (4)$$

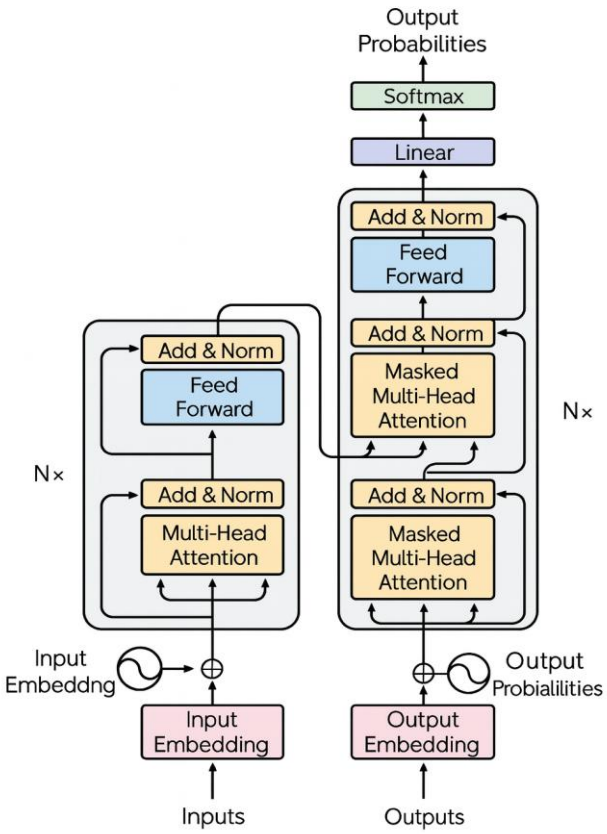
Pada rumus persamaan 4, A dan B merupakan dua buah vektor yang merepresentasikan dokumen atau kalimat yang akan dibandingkan tingkat kesamaannya. Setiap vektor terdiri dari sejumlah elemen, di mana A_i adalah bobot *term* ke-i pada vektor A, dan B_i adalah bobot *term* ke-i pada vektor B. Nilai i menunjukkan jumlah *term* yang terdapat dalam representasi kalimat atau dokumen, sedangkan n adalah jumlah total vektor yang digunakan dalam proses pembandingan.

2.10 *Tranformer*

Transformer merupakan salah satu jenis *deep neural network* (DNN) yang dikembangkan untuk mengatasi keterbatasan arsitektur *sequence-to-sequence* (seq2seq), seperti ketergantungan jangka pendek pada data sekuensial serta proses komputasi yang harus dilakukan secara berurutan sehingga menghambat pelatihan model secara paralel. Model *Transformer* memanfaatkan mekanisme *multi-head self-attention* untuk mengekstraksi fitur dari data, sehingga menunjukkan potensi yang besar dalam berbagai tugas *Natural Language Processing* (NLP). Berbeda dengan metode rekuren tradisional, *Transformer* menggunakan mekanisme perhatian (*attention*) untuk mempelajari informasi dari seluruh bagian sekuens secara simultan melalui komponen *encoding* dan *decoding*. Salah satu keunggulan utama *Transformer* dibandingkan LSTM dan jaringan saraf rekuren lainnya adalah kemampuannya dalam menangkap makna konteks secara lebih menyeluruh melalui mekanisme *attention*. Selain itu, *transformer* memiliki kecepatan komputasi yang lebih tinggi karena memungkinkan pemrosesan paralel dan dapat dioptimalkan dengan penggunaan *Graphics Processing Units* (GPU),

sehingga mampu menangani data dalam jumlah besar dengan waktu pemrosesan yang lebih singkat. Berbagai keunggulan tersebut mendorong banyak penelitian lanjutan yang mengeksplorasi penerapan *transformer* pada beragam tugas dan bidang, yang pada akhirnya menghasilkan berbagai pengembangan model *transformer* dalam ranah kecerdasan buatan (Islam *et al.,* 2023).

Sebagai perwujudan dari mekanisme dan keunggulan tersebut, Gambar 5 memperlihatkan struktur model *transformer* yang memanfaatkan lapisan *stacked self-attention* serta lapisan *point-wise fully connected* pada bagian encoder dan decoder, yang masing-masing ditunjukkan pada sisi kiri dan kanan gambar.



Gambar 5. Arsitektur Model *Transformer* (Vaswani *et al.,* 2023).

Gambar 5 menggambarkan arsitektur model Transformer yang terdiri atas dua komponen utama, yaitu *encoder* (bagian kiri) dan *decoder* (bagian

kanan), yang saling terintegrasi dalam memproses serta menghasilkan keluaran dari teks masukan.

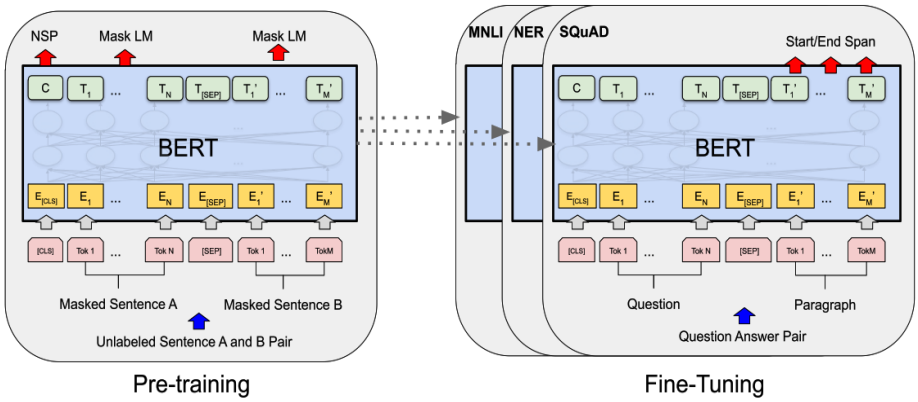
Pada bagian *encoder*, proses diawali dari teks masukan (*inputs*) yang diubah menjadi representasi vektor melalui lapisan *input embedding* . Representasi ini kemudian dikombinasikan dengan informasi posisi kata (*positional encoding*, dilambangkan dengan simbol $\oplus$) agar model dapat mengenali urutan kata dalam kalimat. Selanjutnya, vektor tersebut diproses dalam blok utama *Encoder* yang mencakup dua sub-lapisan, yaitu *multi-head attention* untuk menangkap hubungan antar kata, serta *feed forward* untuk mengolah informasi lebih lanjut. Setiap sub-lapisan dilengkapi dengan mekanisme *add & norm* guna menjaga stabilitas selama proses pelatihan. Blok ini diulang sebanyak N kali agar model semakin memahami konteks kalimat secara mendalam.

Pada bagian *decoder*, alurnya relatif serupa, namun terdapat tambahan sub-lapisan berupa *masked multi-head attention*. Lapisan ini berfungsi membatasi akses model hanya pada kata-kata sebelumnya saat memprediksi kata berikutnya, sehingga mencegah kebocoran informasi. *Decoder* juga menerima masukan dari keluaran sebelumnya (*output embedding*) yang telah dikombinasikan dengan informasi posisi. Setelah melalui rangkaian blok yang diulang sebanyak N kali, hasilnya diteruskan ke lapisan *linear* dan kemudian ke *softmax* untuk menghasilkan probabilitas setiap kemungkinan kata sebagai keluaran akhir (*output probabilities*).

Dengan arsitektur tersebut, model Transformer mampu memahami konteks kalimat secara keseluruhan dan menghasilkan keluaran yang relevan. Oleh karena itu, arsitektur ini menjadi dasar bagi pengembangan model bahasa modern seperti BERT dan SBERT yang digunakan dalam penelitian ini.

2.11 *Bidirectional Encoder Representations from Transformers* (BERT)

BERT (*Bidirectional Encoder Representations from Transformers*) merupakan model bahasa yang bersifat inovatif dalam bidang *Natural Language Processing* (NLP) dan berbasis pada arsitektur encoder Transformer untuk menghasilkan representasi teks yang kontekstual. Melalui tahapan tokenisasi, pembentukan *embedding*, serta penerapan mekanisme *attention* pada setiap lapisan Transformer, BERT mampu mempelajari hubungan antar kata secara mendalam dan bersifat dua arah (*bidirectional*). Keunggulan utama BERT terletak pada kemampuannya memahami konteks kata dengan mempertimbangkan informasi dari sisi kiri dan kanan secara bersamaan, sehingga menghasilkan representasi vektor yang kaya makna. Selain itu, BERT berperan penting dalam pengembangan pendekatan *transfer learning* pada NLP, karena model ini dapat dilatih terlebih dahulu pada data berskala besar dan kemudian disesuaikan untuk berbagai tugas spesifik. Oleh sebab itu, BERT membawa perubahan signifikan dalam paradigma pemrosesan bahasa alami dan mendukung pengembangan aplikasi lanjutan seperti klasifikasi teks, analisis sentimen, serta pemahaman bahasa yang lebih kontekstual (Aljabar & Karomah, 2024). Untuk memperjelas tahapan serta alur kerja model BERT dalam proses *pre-training* dan *fine-tuning*, ilustrasi mengenai kerangka kerja tersebut disajikan pada Gambar 6.



Gambar 6. Alur umum proses *pre-training* dan *fine-tuning* pada model BERT (Devlin *et al.,* 2019).

Kerangka kerja BERT terdiri atas dua tahap utama, yaitu *pre-training* dan *fine-tuning*. Pada tahap *pre-training*, model dilatih menggunakan data tidak berlabel melalui berbagai tugas pelatihan awal. Selanjutnya, pada tahap *fine-tuning*, model BERT diinisialisasi dengan parameter hasil *pre-training* dan seluruh parameter tersebut disesuaikan kembali menggunakan data berlabel sesuai dengan tugas lanjutan (*downstream tasks*). Meskipun setiap tugas lanjutan menggunakan model hasil *fine-tuning* yang berbeda, seluruh model tersebut berasal dari parameter awal yang sama. Salah satu karakteristik utama BERT adalah penggunaan arsitektur yang terpadu untuk berbagai jenis tugas, dengan perbedaan yang sangat minimal antara arsitektur saat *pre-training* dan arsitektur akhir pada tahap *fine-tuning* (Devlin *et al.,* 2019).

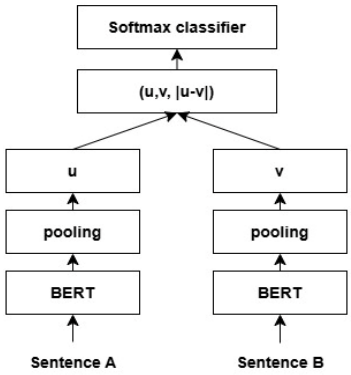
Secara arsitektural, BERT dibangun menggunakan encoder Transformer bidireksional berlapis banyak, yang mengacu pada arsitektur Transformer asli sebagaimana diperkenalkan oleh Vaswani et al. (2017) dan diimplementasikan dalam pustaka *tensor2tensor*. Mengingat penggunaan Transformer telah menjadi pendekatan yang umum dan implementasi BERT sangat mirip dengan arsitektur aslinya, penjelasan mendetail mengenai struktur model tidak dibahas secara mendalam (Devlin *et al.,* 2019).

2.12 *Sentence*-BERT

Sentence-BERT (SBERT), yaitu pengembangan dari arsitektur BERT yang memanfaatkan pendekatan siamese dan triplet network untuk menghasilkan *sentence embedding* yang memiliki makna semantik yang kuat. Dengan kemampuan ini, BERT dapat diterapkan pada jenis tugas baru yang sebelumnya tidak memungkinkan, seperti perbandingan kesamaan semantik dalam skala besar, proses *clustering*, serta *information retrieval* berbasis *semantic search*. Pendekatan ini secara signifikan mengurangi waktu pencarian pasangan kalimat paling mirip dari sekitar 65 jam pada model BERT atau RoBERTa menjadi hanya sekitar 5 detik pada SBERT tanpa

mengurangi tingkat akurasi yang dimiliki BERT (Reimers & Gurevych, 2019).

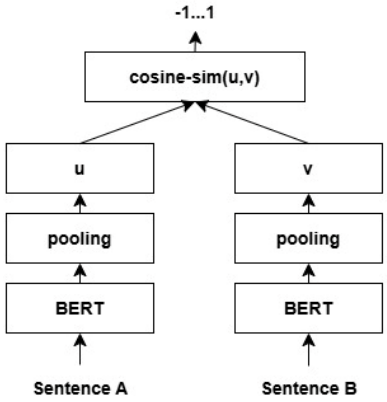
Dengan memanfaatkan *arsitektur siamese network*, SBERT mampu menghasilkan *sentence embedding* berukuran tetap untuk setiap kalimat. *embedding* ini kemudian dapat dibandingkan menggunakan berbagai ukuran kesamaan, seperti *cosine similarity* atau jarak Manhattan dan Euclidean, sehingga kalimat yang memiliki kedekatan makna dapat diidentifikasi dengan mudah. Perhitungan ukuran kesamaan tersebut sangat efisien pada perangkat keras modern, memungkinkan SBERT digunakan secara efektif dalam *semantic similarity search* maupun *clustering* (Reimers & Gurevych, 2019). Gambaran mengenai arsitektur SBERT yang digunakan dalam proses klasifikasi dapat dilihat pada Gambar 7, yang menunjukkan bagaimana pasangan kalimat diproses secara paralel untuk menghasilkan representasi vektor yang bermakna secara semantik.



Gambar 7. Arsitektur SBERT dengan fungsi objektif klasifikasi (Reimers & Gurevych, 2019).

Gambar 7 menunjukkan mekanisme kerja arsitektur SBERT (*Sentence-BERT*) dalam membandingkan dua kalimat secara bersamaan. Proses dimulai dengan dua kalimat masukan, yaitu *Sentence A* dan *Sentence B*, yang diproses secara terpisah tetapi menggunakan model yang sama. Setiap kalimat terlebih dahulu dimasukkan ke dalam model BERT untuk menghasilkan representasi vektor dari setiap kata. Selanjutnya, keluaran tersebut diringkas melalui

proses *pooling*, yaitu tahap penyederhanaan di mana sekumpulan vektor kata diubah menjadi satu vektor tunggal yang merepresentasikan makna keseluruhan kalimat. Vektor hasil dari *Sentence A* dilambangkan sebagai u, sedangkan dari *Sentence B* sebagai v. Kedua vektor tersebut kemudian dibandingkan dengan menghitung selisihnya, yaitu |u − v|, sehingga terbentuk representasi gabungan berupa (u, v, |u − v|). Representasi ini mencerminkan tingkat kesamaan atau perbedaan antara kedu a kalimat. Selanjutnya, hasil gabungan tersebut diproses oleh lapisan *Softmax classifier* untuk menghasilkan keputusan akhir, seperti apakah kedua kalimat memiliki makna yang sama, bertentangan, atau tidak berkaitan. Sementara itu, proses SBERT ketika digunakan pada tahap pencarian kesamaan tidak lagi melibatkan classifier, melainkan langsung menghitung kedekatan antar *embedding* , sebagaimana ditunjukkan pada Gambar 8.



Gambar 8. Arsitektur SBERT pada tahap inferensi untuk menghitung skor kesamaan (Reimers & Gurevych, 2019).

Gambar 8 memperlihatkan arsitektur *Sentence*-BERT (SBERT) pada tahap inferensi yang digunakan untuk menghitung skor kesamaan antar kalimat. Secara umum, alur prosesnya serupa dengan yang ditunjukkan pada Gambar 7, namun terdapat perbedaan pada tahap akhir. Dua kalimat masukan, yaitu *Sentence A* dan *Sentence B*, diproses secara terpisah menggunakan model BERT untuk menghasilkan representasi vektor dari setiap kata. Selanjutnya, keluaran tersebut diringkas melalui proses *pooling* sehingga masing-masing

kalimat direpresentasikan oleh satu vektor tunggal yang menggambarkan makna keseluruhannya. Vektor untuk *Sentence A* dilambangkan sebagai u, sedangkan untuk *Sentence B* sebagai v. Perbedaan utama terletak pada tahap akhir, di mana kedua vektor tersebut tidak digabungkan untuk proses klasifikasi. Sebaliknya, kemiripan antara u dan v dihitung secara langsung menggunakan metode *cosine similarity*, yaitu *cosine-sim(u, v)*. Nilai yang dihasilkan berada dalam rentang -1 hingga 1. Nilai yang mendekati 1 menunjukkan bahwa kedua kalimat sangat mirip atau memiliki makna yang hampir sama, nilai mendekati 0 menunjukkan tidak adanya keterkaitan, sedangkan nilai mendekati -1 menunjukkan makna yang saling bertentangan.

SBERT menambahkan sebuah operasi *pooling* pada keluaran BERT atau RoBERTa untuk menghasilkan *sentence embedding* berukuran tetap. Dalam prosesnya, penelitian ini mengevaluasi tiga strategi *pooling*, yaitu menggunakan keluaran token CLS, menghitung rata-rata seluruh vektor keluaran melalui MEAN strategy, serta mengambil nilai maksimum dari setiap dimensi vektor sepanjang urutan melalui MAX strategy (Reimers & Gurevych, 2019).

2.13 Google Colaboratory

Google Colab merupakan layanan *Cloud Computing* yang disediakan oleh Google untuk mendukung kegiatan pengembangan dan penelitian ilmiah. *Colaboratory* atau "*Colab*" adalah produk dari *Google Research* yang memungkinkan pengguna menulis serta menjalankan kode Python langsung melalui browser. Platform ini sangat cocok digunakan untuk keperluan *Machine Learning*, analisis data, maupun kegiatan pendidikan. Selain itu, *Google Colab* dapat digunakan secara kolaboratif oleh beberapa pengembang aplikasi, sehingga sangat mendukung kerja sama antaranggota tim. Meskipun demikian, layanan ini masih tergolong asing bagi masyarakat umum karena umumnya digunakan oleh mereka yang membutuhkan lingkungan

pemrograman, seperti pengembang (*developer*) atau *programmer* (Wilyani *et al.,* 2024).

Google Colaboratory (Colab) memiliki sejumlah fitur unggulan yang mendukung kegiatan komputasi dan analisis data berbasis cloud. Pertama, Python di Cloud, yaitu pengguna dapat menulis serta menjalankan kode Python langsung melalui browser tanpa perlu menginstal Python maupun pustaka pendukung di komputer pribadi. Kedua, layanan ini bersifat gratis, sehingga dapat digunakan tanpa biaya langganan. Ketiga, tersedia akses GPU (*Graphics Processing Unit*) secara cuma-cuma yang sangat bermanfaat untuk proses pelatihan model pembelajaran mesin yang membutuhkan kemampuan komputasi tinggi. Keempat, integrasi dengan Google Drive memungkinkan pengguna untuk menyimpan, mengakses, dan membagikan file secara langsung di penyimpanan awan milik mereka. Kelima, format notebook interaktif yang digunakan memungkinkan penggabungan antara kode program, teks penjelas, gambar, dan hasil eksekusi dalam satu dokumen, sehingga sangat membantu untuk dokumentasi dan kolaborasi hasil analisis. Keenam, tersedia berbagai pustaka populer seperti NumPy, Pandas, TensorFlow, dan PyTorch yang dapat diimpor dengan mudah ke dalam lingkungan kerja Colab. Ketujuh, fitur kolaborasi waktu nyata memungkinkan beberapa pengguna untuk mengerjakan satu notebook secara bersamaan. Terakhir, Google Colab bersifat fleksibel dan mudah digunakan, menjadikannya pilihan yang ideal bagi pemula dalam mempelajari Python maupun bagi peneliti yang ingin melakukan eksplorasi data tanpa perlu konfigurasi sistem lokal yang rumit (Andarsyah & Yanuar Amri, 2024).

2.14 L2 Normalisasi (Normalisasi Euclidean)

Salah satu metode normalisasi yang umum digunakan adalah L2 normalisasi, yang didasarkan pada konsep L2-norm atau yang dikenal juga sebagai norma

Euclidean. L2-norm merupakan norma dengan nilai p = 2 dan merupakan norma yang paling umum digunakan untuk mengukur panjang atau besaran suatu vektor. Normalisasi Euclidean didefinisikan sebagai fungsi yang memberikan panjang ketat positif atau ukuran untuk semua vektor dalam ruang vektor. Secara matematis, L2-norma dari suatu vektor $\vec{u}$ dihitung menggunakan persamaan berikut (Jatmika *et al.,* 2018):

$$||\vec{u}|| = \sqrt{u_1^2 + u_2^2 + u_3^2 + \cdots + u_n^2} \quad (5)$$

Dalam persamaan 5 tersebut, ‖ $\vec{u}$ ‖melambangkan nilai panjang atau besaran (magnitude) dari vektor $\vec{u}$ yang ingin dihitung. Simbol $u_1, u_2, u_3, \ldots, u_n$ merepresentasikan setiap elemen atau komponen yang terdapat di dalam vektor, di mana $u_1$adalah elemen pertama, u_2 adalah elemen kedua, dan seterusnya hingga elemen ke-n. Sedangkan n melambangkan jumlah total dimensi dari vektor tersebut. Proses perhitungan dilakukan dengan cara mengkuadratkan setiap elemen vektor terlebih dahulu, kemudian menjumlahkan seluruh hasil kuadrat tersebut, dan terakhir mengambil akar kuadrat dari hasil penjumlahan. Hasil akhir dari perhitungan ini adalah sebuah nilai skalar positif yang merepresentasikan panjang atau jarak vektor dari titik asal dalam ruang berdimensi-n. Semakin besar nilai ‖ $\vec{u}$ ‖, maka semakin jauh posisi vektor tersebut dari titik asal, dan sebaliknya jika nilainya mendekati 1 maka vektor tersebut sudah mendekati bentuk vektor satuan (unit vector).

Berdasarkan rumus tersebut, L2-norma diperoleh dari akar kuadrat dari jumlah seluruh elemen vektor yang dikuadratkan. Selanjutnya, proses normalisasi dilakukan dengan membagi setiap vektor dengan nilai L2-norm-nya, sehingga menghasilkan vektor satuan (*unit vector*) dengan panjang bernilai 1, yang dirumuskan sebagai berikut:

$$\hat{v} = \frac{\vec{u}}{||\vec{u}||_p} \quad (6)$$

Dalam persamaan 6 tersebut, $\hat{v}$ melambangkan vektor hasil normalisasi atau yang dikenal sebagai vektor satuan (unit vector), yaitu vektor baru yang diperoleh setelah proses normalisasi dilakukan. Simbol $\vec{u}$ merepresentasikan vektor asli sebelum dinormalisasi, yang berisi nilai-nilai elemen dari data yang akan diproses. Sedangkan $\| \vec{u} \|_p$ melambangkan nilai L2-norma dari vektor $\vec{u}$, yaitu nilai panjang atau besaran vektor yang dihitung menggunakan rumus norma Euclidean dengan nilai $p = 2$. Proses normalisasi dilakukan dengan cara membagi setiap elemen vektor asli $\vec{u}$ dengan nilai L2-norma $\| \vec{u} \|_p$, sehingga menghasilkan vektor baru $\hat{v}$ yang memiliki panjang bernilai 1. Dengan kata lain, arah vektor tetap dipertahankan namun skalanya disesuaikan menjadi seragam. Tujuan dari normalisasi ini adalah untuk memastikan bahwa setiap vektor berada pada skala yang sama, sehingga perbandingan jarak maupun kemiripan antar vektor dapat dilakukan secara konsisten dan tidak dipengaruhi oleh perbedaan besaran antar vektor.

2.15 Evaluasi Model

Untuk mengukur kinerja model yang diuji, penelitian ini menggunakan beberapa metrik evaluasi yang umum digunakan dalam ML, seperti *Mean Average precision* (MAP), *recall*, dan *precision*.

recall didefinisikan sebagai proporsi dokumen yang berhasil ditemukan kembali oleh suatu proses pencarian informasi dibandingkan dengan seluruh dokumen yang relevan (Martin & Nilawati, 2019). Berikut rumus menentukan *recall* dalam sistem temu kembali informasi (Kurniadi *et al.,* 2020):

$$recall = \frac{jumlah\ dokumen\ relevan\ yang\ terpanggil\ (a)}{jumlah\ dokumen\ relevan\ yang\ ada\ di\ database\ (a+c)} \text{ x } 100\% \qquad (7)$$

Pada rumus persamaan 7, *recall* digunakan untuk mengukur sejauh mana sistem mampu menemukan seluruh dokumen relevan dari *database*. Nilai *recall* dihitung dengan membagi jumlah dokumen relevan yang berhasil dipanggil oleh sistem (a) dengan total seluruh dokumen relevan yang terdapat di dalam *database*, yaitu gabungan antara dokumen relevan yang terpanggil dan dokumen relevan yang tidak terpanggil $(a+c)$, kemudian dikalikan 100%. Semakin tinggi nilai *recall*, semakin besar kemampuan sistem dalam menemukan dokumen relevan yang tersedia. Sebaliknya, nilai *recall* yang rendah menunjukkan bahwa masih ada banyak dokumen relevan yang belum berhasil ditemukan oleh sistem.

Sedangkan *precision* adalah ukuran yang menunjukkan proporsi dokumen yang ditemukan oleh suatu proses pencarian informasi yang benar-benar relevan dengan kebutuhan pencarian, atau dapat juga diartikan sebagai rasio antara jumlah dokumen relevan yang berhasil ditemukan dengan total dokumen yang ditemukan (Martin & Nilawati, 2019). Berikut rumus menentukan *precision* dalam sistem temu kembali informasi (Kurniadi *et al.,* 2020):

$$precision = \frac{jumlah\ dokumen\ relevan\ yang\ terpanggil\ (a)}{jumlah\ dokumen\ terpanggil\ dalam\ pencarian\ (a+b)}\ x\ 100\% \qquad (8)$$

Pada rumus persamaan 8, *precision* didefinisikan sebagai perbandingan antara jumlah dokumen relevan yang berhasil ditemukan (a) dengan total keseluruhan dokumen yang diambil oleh sistem, yaitu penjumlahan dokumen relevan dan tidak relevan (a+b). Formula ini pada dasarnya mengukur tingkat ketepatan atau akurasi sistem dalam menyaring dan menampilkan informasi yang benar-benar sesuai dengan kebutuhan pencarian pengguna. Struktur hubungan antara dokumen relevan, tidak relevan, ditemukan, dan tidak ditemukan tersebut dapat dilihat pada Tabel 2, yang menyajikan matriks *recall* dan *precision* sebagai dasar perhitungan evaluasi performa sistem temu kembali informasi.

Tabel 2. Matriks *Recall* dan *Precision* (Martin & Nilawati, 2019).

	Relevant	***Not Relevant***	***Total***
Retrieved	a (*hits*)	b (*noise*)	a+b
Not Retrieved	c (*misses*)	d (*reject*)	c+d
Total	a+c	b+d	a+b+c+d

Keterangan:

a (*hits*) adalah Dokumen yang relevan dan berhasil ditemukan sistem.

b (*noise*) adalah Dokumen tidak relevan yang ikut terambil (kesalahan sistem).

c (*misses*) adalah Dokumen relevan yang tidak ditemukan sistem (terlewat).

d (*reject*) adalah Dokumen tidak relevan yang memang tidak diambil (benar ditolak).

Secara sederhana, mAP merupakan nilai rata-rata dari *Average precision* (AP) dan digunakan untuk mengukur kualitas performa model berdasarkan hasil pelatihan data. Berikut rumus dari mAP (Hayati & Alifi, 2025) :

$$\text{AP@K}=\frac{1}{N}\sum_{k=1}^{K} precision(k)x\ rel(k) \tag{9}$$

Pada rumus persamaan 9, AP@K digunakan untuk menghitung rata-rata nilai *precision* pada posisi dokumen relevan yang muncul di dalam Top K hasil rekomendasi. Nilai N menunjukkan jumlah seluruh dokumen relevan, precision(k) adalah nilai *precision* pada posisi ke-k, sedangkan rel(k) bernilai 1 jika dokumen pada posisi ke-k relevan dan bernilai 0 jika tidak relevan.

$$\text{MAP@K}=\frac{1}{Q}\sum_{q=1}^{Q} AP@Kq \tag{10}$$

Pada rumus persamaan 10, MAP@K digunakan untuk menghitung rata-rata nilai *Average precision* pada seluruh *query* yang diuji. Nilai Q menunjukkan jumlah *query* yang diujikan, sehingga MAP@K diperoleh dengan menjumlahkan nilai AP@K dari setiap *query*, kemudian dibagi dengan total *query*. Dengan demikian, setiap *query* terlebih dahulu dihitung nilai AP@K-nya, kemudian seluruh nilai tersebut dirata-ratakan untuk menghasilkan nilai MAP@K.

2.16 Teknologi Pengembangan *Website*

Teknologi *website* merupakan gabungan berbagai komponen seperti teks, gambar, suara, dan animasi yang menjadikannya sebagai media informasi yang menarik serta banyak diminati untuk digunakan dalam berbagi informasi. Melalui teknologi ini, data diolah menjadi informasi dengan proses identifikasi, pengumpulan, pengelolaan, dan penyediaan data agar dapat diakses secara bersamaan oleh pengguna (Pamilih *et al.,* 2018).

Kemunculan teknologi *website* yang didukung oleh layanan jaringan internet merupakan salah satu pencapaian terbesar dalam bidang teknologi informasi, hasil dari berbagai penelitian yang dilakukan oleh para ahli dan peneliti di bidang tersebut. Perkembangan ini bermula dari penelitian yang dilakukan oleh sekelompok ilmuwan di CERN (Organisasi Penelitian Nuklir Eropa) pada tahun 1990, yang mengembangkan proyek halaman web statis atau dikenal sebagai Web 1.0 menggunakan HTML (*Hyper Text Markup Language*). Selanjutnya, dilakukan pengembangan terhadap standar protokol komunikasi di dunia web yang disebut HTTP (*HyperText Transfer Protocol*). Kedua inovasi ini menjadi fondasi utama terbentuknya jaringan internet modern yang kita kenal saat ini sebagai (WWW) (Wijaya *et al.,* 2017).

Dalam penelitian ini, teknologi *website* digunakan sebagai dasar pengembangan sistem informasi repositori skripsi. Teknologi yang diterapkan mencakup bahasa pemrograman dan *framework* yang mendukung

pengembangan sisi antarmuka (*front-end*), pengolahan logika aplikasi (*back-end*), serta pengelolaan basis data. Adapun teknologi *website* yang digunakan dalam penelitian ini meliputi HTML, CSS, Python, Flask, dan *database Management System* (DBMS), yang akan dijelaskan pada subbab berikutnya.

2.16.1 HTML

HTML merupakan bahasa yang digunakan untuk membangun struktur dasar sebuah halaman web. Pada tahap ini, diperkenalkan berbagai elemen fundamental HTML yang membentuk kerangka halaman web. Materi yang dibahas meliputi penggunaan tag-tag HTML yang umum, seperti tag <html> yang menandai awal dan akhir dokumen, serta tag <head> yang digunakan untuk menambahkan metadata, judul halaman, dan tautan ke file CSS eksternal. Selain itu, juga dijelaskan fungsi tag <body> sebagai tempat untuk menampilkan konten utama halaman seperti teks, gambar, dan berbagai media lainnya. Pembahasan selanjutnya mencakup penggunaan tag heading <h1> hingga <h6> untuk menentukan ukuran serta hierarki teks, dan tag <p> untuk membentuk paragraf. Tag penting lainnya seperti <a> untuk membuat tautan, <img> untuk menyisipkan gambar, serta elemen formulir seperti <*input*> dan <button> untuk membuat form interaktif juga turut dibahas (Mardiansyah *et al.,* 2025).

2.16.2 CSS

CSS merupakan bahasa yang digunakan untuk menata dan memperindah tampilan halaman web. Melalui CSS, pengembang web dapat mengatur berbagai elemen visual seperti warna latar belakang, ukuran dan jenis huruf, serta tata letak halaman. Materi pembahasan meliputi berbagai properti penting dalam CSS, antara lain *color* untuk mengubah warna teks, *background-color* untuk menentukan warna latar belakang, dan *font-family* untuk meng atur jenis huruf yang digunakan pada halaman. Selain itu, dibahas juga pengaturan ukuran

dan jarak antar elemen menggunakan properti seperti *width*, *height*, *margin*, dan *padding*, yang berfungsi untuk mengatur ruang di sekitar elemen-elemen web. Pembahasan selanjutnya mencakup properti tata letak seperti *display*, *float*, dan *position*, yang memungkinkan penataan elemen secara fleksibel dalam berbagai bentuk layout, seperti kolom tunggal, grid, atau pola lainnya. Selain itu, juga dijelaskan penggunaan teknik desain responsif dengan *media Query*, sehingga tampilan halaman web dapat menyesuaikan secara optimal pada berbagai ukuran layar, baik perangkat mobile maupun desktop (Mardiansyah *et al.,* 2025) .

2.16.3 Python

Bahasa pemrograman Python merupakan salah satu bahasa yang banyak digunakan dalam bidang analisis data. Popularitasnya disebabkan oleh kemudahan dalam mempelajari dan menggunakannya, sehingga dapat diakses oleh berbagai kalangan. Selain itu, Python bersifat *open source*, sehingga dapat digunakan secara bebas di berbagai sistem operasi. Bahasa ini juga dilengkapi dengan beragam *Library* yang memiliki fungsi spesifik, seperti NumPy untuk komputasi numerik, Pandas untuk pengolahan data, Matplotlib untuk visualisasi data, serta Scikit-learn untuk penerapan *Machine Learning*. Lebih lanjut, Python memiliki kemampuan integrasi yang tinggi dengan berbagai teknologi lain, seperti basis data, *big data tools*, dan *framework web*, yang mendukung proses pengolahan serta analisis data dari berbagai sumber secara efisien (Sambi Ua *et al.,* 2023).

2.16.4 Flask

Flask merupakan salah satu *web framework* yang digunakan dalam bahasa pemrograman Python. *Framework* ini tergolong sebagai *micro-framework* karena tidak memerlukan alat (*tools*) atau *library*

tambahan serta tidak memiliki basis data bawaan. Flask memanfaatkan Jinja Template Engine dan Werkzeug WSGI Toolkit sebagai komponen utamanya. Secara struktural, Flask terbagi menjadi dua kategori utama, yaitu Static File, yang berisi seluruh kode pendukung tampilan *website* seperti CSS, JavaScript, dan file gambar, serta Template File, yang memuat seluruh template Jinja termasuk halaman HTML yang digunakan dalam pengembangan aplikai web (Fitri & Setiyawati, 2021).

2.16.5 *Database Management System* (DBMS)

database Management System (DBMS) atau sistem manajemen basis data adalah perangkat lunak yang dirancang untuk mengelola dan mengatur data dalam sebuah basis data. Basis data sendiri merupakan kumpulan data yang tersusun secara terstruktur, tersimpan dengan rapi, dan dapat diakses dengan mudah. DBMS memiliki berbagai fungsi penting, termasuk penyimpanan, pengelolaan, dan akses data secara efisien, sehingga peranannya sangat krusial dalam bidang teknologi maupun bisnis (Siregar *et al.,* 2024).

Seiring meningkatnya ketergantungan dunia bisnis pada data, pemahaman tentang DBMS menjadi semakin penting. Pengetahuan mengenai fungsi, jenis, dan komponen utama DBMS membantu organisasi dalam mengelola data secara lebih efektif dan aman. Bagi para praktisi data atau individu yang sering bekerja dengan data, penguasaan SQL menjadi hal yang esensial. SQL adalah bahasa pemrograman yang digunakan untuk mengakses basis data relasional, memudahkan pengolahan data, dan hampir didukung oleh semua server basis data modern. Beberapa sistem basis data populer yang mendukung SQL antara lain *MySQL*, SQL Server, dan Oracle, meskipun fitur SQL dapat berbeda-beda di tiap sistem tersebut (Siregar *et al.,* 2024).

2.17 *Systems Development Life Cycle* (SDLC)

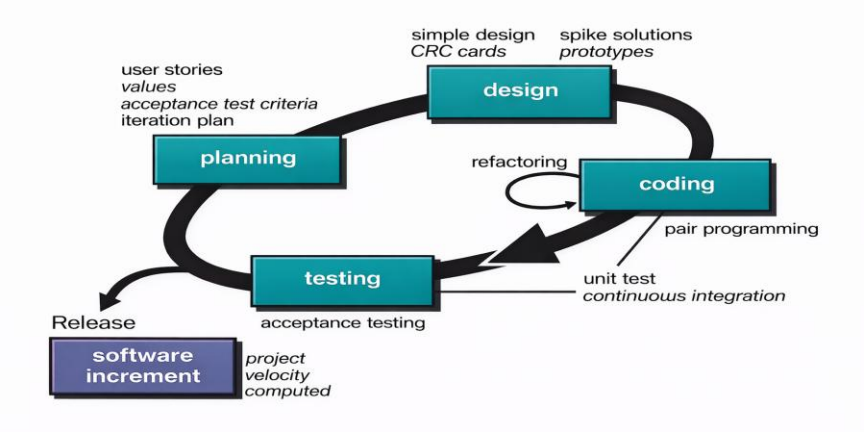
SDLC (*Systems Development Life Cycle* atau Siklus Hidup Pengembangan Sistem) merupakan suatu proses dalam rekayasa sistem dan rekayasa perangkat lunak yang mencakup kegiatan perancangan, pengembangan, serta modifikasi sistem, beserta model dan metodologi yang digunakan di dalamnya. Konsep ini umumnya diterapkan pada sistem komputer atau sistem informasi. SDLC berfungsi sebagai kerangka kerja yang sistematis dalam pengembangan perangkat lunak, yang meliputi beberapa tahapan utama, yaitu perencanaan (*planning*), analisis (*analysis*), perancangan (*design*), implementasi (*implementation*), pengujian (*testing*), dan pemeliharaan (*maintenance*) (Pricillia & Zulfachmi, 2021).

Dalam penerapannya, SDLC dapat diimplementasikan melalui berbagai model atau metode pengembangan perangkat lunak. Salah satu metode yang termasuk dalam pendekatan SDLC adalah *extreme programming* (XP), yang menekankan proses pengembangan secara iteratif, adaptif terhadap perubahan, dan berfokus pada keterlibatan pengguna secara aktif.

2.17.1 *Extreme Programming* (XP)

Extreme Programming (XP) merupakan salah satu metode rekayasa perangkat lunak yang berorientasi pada pendekatan berorientasi objek. Metode ini paling efektif diterapkan pada tim pengembangan berukuran kecil hingga menengah, terutama pada kondisi di mana kebutuhan atau harapan pengguna belum terdefinisi secara jelas atau berpotensi mengalami perubahan. XP sangat sesuai digunakan pada proyek pengembangan perangkat lunak yang menuntut adanya penyesuaian dan perubahan secara berulang selama proses pengembangan. Tahapan-tahapan dalam metode *extreme programming*, yang meliputi perencanaan, perancangan, pengkodean,

dan pengujian, disusun secara sistematis sebagaimana ditampilkan pada Gambar 9 (Illahi *et al.,* 2023).



Gambar 9. Tahapan Metode *Extreme Programming* (Illahi *et al.,* 2023).

a. Tahap Perencanaan (*Planning*)

Tahap perencanaan diawali dengan proses pengumpulan data terkait kondisi sistem yang berjalan. Pada tahap ini dilakukan identifikasi permasalahan serta analisis kebutuhan sistem untuk memperoleh gambaran mengenai keluaran (*output*), fitur, dan fungsi yang diharapkan oleh pengguna (Illahi *et al.,* 2023).

b. Tahap Perancangan (*Design*)

Pada tahap perancangan, hasil analisis kebutuhan diterjemahkan ke dalam bentuk pemodelan sistem. Proses ini mencakup perancangan struktur sistem, pemodelan basis data untuk menggambarkan hubungan antar data, serta perancangan tata letak antarmuka pengguna. Pemodelan sistem dilakukan menggunakan *Unified Modeling Language* (UML), yang meliputi *Use Case Diagram*, *class diagram*, dan *Activity Diagram* (Illahi *et al.,* 2023).

c. Tahap Pengkodean (*Coding*)

Tahap pengkodean merupakan fase implementasi, di mana pengembang mulai menerjemahkan hasil perancangan dan *user stories* yang telah disepakati ke dalam bentuk kode program. Dalam metode *Extreme Programming*, praktik *pair programming* sering diterapkan, yaitu dua orang pengembang bekerja secara bersamaan untuk menulis serta meninjau kode secara bergantian. Pendekatan ini bertujuan untuk meminimalkan kesalahan, meningkatkan kualitas kode, serta memastikan bahwa hasil pengembangan sesuai dengan standar yang telah ditetapkan (Fachrurozi *et al.,* 2025).

d. Tahap Pengujian (*Testing*)

Pada tahap pengujian, pengguna sistem menentukan aspek dan metode pengujian dengan mempertimbangkan seluruh fitur dan fungsi yang tersedia. Selanjutnya dilakukan evaluasi terhadap kinerja sistem berdasarkan hasil pengujian yang diperoleh. Apabila hasil pengujian belum memenuhi kriteria yang telah ditetapkan, maka proses pengembangan sistem akan diulang melalui tahapan *software increment* hingga sistem sesuai dengan kebutuhan pengguna (Illahi *et al.,* 2023).

2.18 Pemodelan Sistem

Pemodelan sistem merupakan proses untuk merepresentasikan rancangan sistem yang bertujuan menggambarkan kebutuhan dan keinginan pengguna dalam pembangunan suatu sistem. Pemodelan sistem juga berfungsi sebagai *blueprint* atau cetak biru pengembangan sistem informasi, sehingga dapat membantu pengembang dalam memahami struktur, alur, serta komponen sistem yang akan dibangun (Latipah, 2022). Salah satu alat yang umum

digunakan untuk tujuan ini adalah *Unified Modelling Language* (UML), yang akan dijelaskan pada subbab berikut.

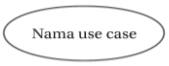
2.18.1 *Unified Modelling Language* (UML)

Unified Modelling Language (UML) merupakan alat yang digunakan untuk memvisualisasikan dan mendokumentasikan hasil analisis serta desain sistem dalam bentuk model visual. UML berfungsi sebagai kumpulan konvensi pemodelan yang digunakan untuk mendeskripsikan atau menggambarkan sistem perangkat lunak yang berorientasi objek. Selain itu, UML juga dapat dipahami sebagai bahasa pemodelan standar yang digunakan dalam pengembangan sistem berparadigma objek. Secara umum, UML memiliki tiga konsep utama, yaitu *structural classification*, *dynamic behavior*, dan *model management*. Konsep-konsep ini menjadi dasar dalam pembuatan berbagai jenis diagram. UML mendefinisikan beberapa jenis diagram, antara lain *Use Case Diagram*, *Class Diagram*, *Statechart Diagram*, *Activity Diagram*, *Sequence Diagram*, *Collaboration Diagram*, *Component Diagram*, dan *Deployment Diagram* (Ronal *et al.,* 2022).

2.18.1.1 *Use Case Diagram*

Use Case atau diagram *use case* merupakan model yang digunakan untuk menggambarkan perilaku dari sistem informasi yang akan dikembangkan. Dalam hal ini, *use case* menjelaskan bentuk interaksi yang terjadi antara satu atau lebih aktor dengan sistem informasi tersebut (Siswidiyanto *et al.,* 2020). Untuk memahami elemen-elemen yang digunakan dalam penyusunannya, Tabel 3 menampilkan simbol-simbol utama pada diagram Use Case beserta penjelasannya (Ramdany *et al.,* 2024).

Tabel 3. Simbol- simbol *Use Case Diagram* (Ramdany *et al.,* 2024)

No	Simbol	Nama	Keterangan
1		*Use Case*	Fungsionalitas yang disediakan oleh sistem digambarkan sebagai sekumpulan unit yang saling berinteraksi atau bertukar pesan antara satu sama lain maupun dengan aktor. Umumnya, penamaan pada *Use Case* diawali dengan kata kerja untuk menunjukkan tindakan atau proses yang dilakukan sistem.
2		Aktor	Aktor merupakan entitas yang dapat berupa orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang sedang dikembangkan, namun berada di luar sistem tersebut.
3		Asosiasi atau *association*	Komunikasi antara aktor dan *Use Case* menunjukkan adanya partisipasi atau interaksi di antara keduanya, di mana aktor berperan dalam menjalankan atau memicu suatu *Use Case* dalam sistem.
4		Ekstensi atau *extend*	Hubungan antara *Use Case* tambahan dengan *Use Case* utama menggambarkan bahwa *Use Case* tambahan tersebut dapat berdiri sendiri meskipun tanpa keberadaan *Use Case* utama.
5		Generalisasi atau *generalization*	Hubungan generalisasi dan spesialisasi merupakan relasi antara dua *Use Case*, di mana satu *Use Case* memiliki fungsi yang lebih umum, sedangkan *Use Case* lainnya memiliki fungsi yang lebih spesifik atau khusus dari *Use Case* tersebut.

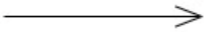
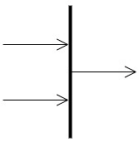
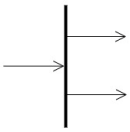
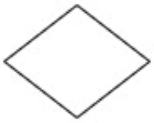
No	Simbol	Nama	Keterangan
6		Mengguna kan/includ e/uses	Relasi *include* pada *Use Case* menggambarkan hubungan antara suatu *Use Case* tambahan dengan *Use Case* lain yang diperlukan untuk menjalankan fungsinya atau menjadi prasyarat agar *Use Case* tersebut dapat dijalankan. Terdapat dua pandangan umum terkait konsep *include* dalam *Use Case*: 1. *Include* berarti *Use Case* yang disertakan akan selalu dijalankan setiap kali *Use Case* utama dijalankan. 2. *Include* berarti *Use Case* tambahan akan terlebih dahulu memeriksa apakah *Use Case* yang disertakan sudah dijalankan sebelum *Use Case* utama dieksekusi.

2.18.1.2 *Activity Diagram*

Pada *Activity Diagram*, setiap proses menerima masukan berupa sumber daya dari sisi kiri dan menghasilkan keluaran di sisi kanan. Diagram ini berfungsi untuk menggambarkan aliran aktivitas dalam suatu sistem informasi secara fungsional. Secara menyeluruh, *Activity Diagram* menjelaskan titik awal dan akhir dari suatu alur kerja, aktivitas-aktivitas yang terjadi selama proses berlangsung, serta urutan kejadian dari aktivitas tersebut. Selain itu, diagram ini juga mendukung pemodelan proses yang berjalan secara paralel. Bagi pengguna yang terbiasa dengan pendekatan analisis dan perancangan sistem tradisional, *Activity Diagram* menggabungkan konsep dasar dari *data flow diagram* (DFD) dan *flowchart* sistem (Ramdany *et al.,*

2024). Untuk memahami elemen penyusunnya, Tabel 4 menampilkan simbol-simbol yang digunakan dalam *Activity Diagram* beserta fungsi masing-masing.

Tabel 4. Simbol-simbol *Activity Diagram* (Ramdany *et al.,* 2024).

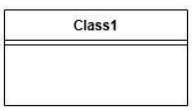
No	Bentuk Simbol	Nama Simbol	Fungsi Simbol
1		*Activity*	Menyatakan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain.
2		*Control Flow*	Menunjukkan Urutan Eksekusi
3		*Object Flow*	Menunjukkan aliran objek dari sebuah action atau activity ke action
4		*Start Point*	Menyatakan bahwa sebuah object dibentuk atau diawali
5		*End Point*	Menyatakan bahwa sebuah object dibentuk atau diakhiri
6		Join/ Penggabungan	Menyatakan untuk menggabungkan kembali activity atau action yang parallel
7		*Fork*	Menyatakan untuk memecah behavior menjadi activity atau action yang parallel
8		*Decision*	Menunjukkan penggambaran suatu keputusan/tindakan

No	Bentuk Simbol	Nama Simbol	Fungsi Simbol
			yang harus diambil pada kondisi tertentu

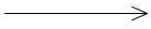
2.18.1.3 *Class Diagram*

Class Diagram merupakan representasi hubungan antar kelas beserta penjelasan rinci dari setiap kelas dalam model perancangan suatu sistem. Diagram ini menampilkan aturan serta tanggung jawab tiap entitas yang membentuk perilaku sistem. Dengan demikian, *Class Diagram* dapat dipahami sebagai visualisasi struktur program yang menggambarkan jenis-jenis komponen yang menyusun sistem. Selain itu, *Class Diagram* juga berfungsi untuk menunjukkan alur data dalam sistem yang sedang dirancang. Diagram ini terdiri atas kumpulan beberapa kelas beserta relasinya. Setiap kelas digambarkan dalam bentuk persegi panjang yang berisi tiga bagian utama, yaitu nama kelas di bagian atas, atribut kelas di bagian tengah, dan metode atau fungsi di bagian bawah. Secara konseptual, *Class Diagram* merupakan spesifikasi yang dapat menghasilkan objek saat diinstansiasi dan menjadi komponen utama dalam pengembangan serta desain sistem berorientasi objek (Wahyu *et al.,* 2024). Untuk memperjelas elemen-elemen pembentuknya, Tabel 5 menyajikan simbol-simbol yang digunakan dalam Class Diagram beserta penjelasannya.

Tabel 5. Simbol-simbol *Class Diagram* (Wahyu *et al.,* 2024).

No	Simbol	Nama	Keterangan
1		*Class*	*Class* merupakan komponen dasar dalam pemrograman berorientasi objek. Secara visual, *class* biasanya digambarkan sebagai sebuah

No	Simbol	Nama	Keterangan
			kotak yang terbagi menjadi tiga bagian utama.
2		*Association*	*Association* merupakan jenis hubungan yang menggambarkan adanya interaksi antara satu *class* dengan *class* lainnya. Hubungan ini biasanya digambarkan dengan garis yang memiliki panah terbuka di ujungnya, yang menunjukkan arah pertukaran pesan atau interaksi antar *class*.
3		*Aggregation*	*Aggregation* menggambarkan hubungan antara suatu objek yang terdiri atas bagian-bagian tertentu, di mana objek tersebut mewakili keseluruhan dari bagian-bagian yang terlibat. Hubungan ini sering disebut sebagai relasi "keseluruhan-bagian" dalam pemodelan sistem.
4		*Composition*	Apabila suatu *class* tidak dapat berdiri sendiri dan harus menjadi bagian dari *class* lain, maka *class* tersebut memiliki hubungan *Composition* dengan *class* induknya. Relasi ini menunjukkan ketergantungan penuh, di mana keberadaan *class* tersebut bergantung pada *class* tempat ia terhubung. Dalam diagram, hubungan *Composition* digambarkan

No	Simbol	Nama	Keterangan
			dengan garis yang memiliki ujung berbentuk belah ketupat
5		*Generalization*	*Generalization* merupakan hubungan antar *class* yang menunjukkan keterkaitan dari *class* yang bersifat spesifik menuju *class* yang lebih umum.
6		*Dependency*	Terkadang, sebuah *class* memanfaatkan *class* lain dalam prosesnya. Hubungan seperti ini disebut *Dependency*. Secara umum, *Dependency* digunakan untuk menunjukkan bahwa suatu operasi dalam satu *class* bergantung atau menggunakan fungsi dari *class* lain. Hubungan ini biasanya digambarkan dengan panah putus-putus.

2.18.1.4 *Entity Relationship Diagram*

Entity Relationship Diagram (ERD) merupakan diagram yang digunakan dalam proses perancangan basis data untuk menggambarkan hubungan atau relasi antar entitas beserta atribut yang dimilikinya. Dengan demikian, ERD berfungsi sebagai model yang menjelaskan keterkaitan antardata dalam suatu basis data berdasarkan objek-objek yang saling berhubungan (Akbar & Haryanti, 2021).

Dalam bidang rekayasa perangkat lunak, *Entity-Relationship Model* (ERM) adalah representasi data yang bersifat abstrak dan konseptual.

Pendekatan *Entity-Relationship* merupakan salah satu metode pemodelan basis data yang umum digunakan untuk menghasilkan skema konseptual dari struktur data suatu sistem. Pemodelan ini umumnya diterapkan pada sistem yang menggunakan basis data relasional melalui pendekatan perancangan secara top-down. Diagram yang digunakan untuk menggambarkan model *Entity-Relationship* tersebut dikenal sebagai *Entity-Relationship* Diagram, ER Diagram, atau ERD (Akbar & Haryanti, 2021). Komponen utama dalam sebuah *Entity Relationship Diagram* (ERD) meliputi (Palinggi *et al.,* 2024):

a. Entitas

   Entitas merupakan objek atau konsep yang datanya perlu disimpan di dalam basis data. Entitas digambarkan menggunakan bentuk persegi panjang dan diberi nama sesuai dengan objek yang direpresentasikan.

b. Atribut

   Atribut adalah sifat atau karakteristik yang menjelaskan suatu entitas. Setiap entitas memiliki satu atau lebih atribut yang berfungsi memberikan informasi mengenai entitas tersebut. Atribut dituliskan di dalam persegi panjang entitas.

c. Relasi

   Relasi menggambarkan hubungan antara dua entitas atau lebih. Relasi divisualisasikan dengan garis yang menghubungkan entitas-entitas terkait dan diberi nama sesuai dengan jenis hubungannya.

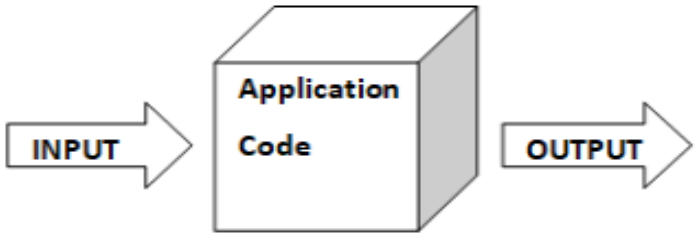
2.19 Pengujian Sistem

Pengujian sistem merupakan aspek yang perlu diperhatikan dan sangat dibutuhkan untuk memastikan kualitas suatu sistem, baik yang sedang dalam tahap pengembangan maupun yang sudah digunakan, agar dapat berfungsi dengan benar. Meskipun pengujian dilakukan pada waktu

tertentu, tujuan utamanya adalah untuk mengidentifikasi kesalahan atau cacat sejak awal sehingga dapat segera diperbaiki (Ijudin & Saifudin, 2020).

2.19.1 *White Box Testing*

White Box Testing merupakan suatu teknik pengujian perangkat lunak yang berfokus pada struktur internal aplikasi. Melalui pendekatan ini, penguji memeriksa logika internal, struktur kode, serta alur kontrol dalam program. Teknik ini juga dikenal dengan sebutan *Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing,* atau *Structural Testing*. Dalam *White Box Testing*, penguji perlu memiliki pemahaman menyeluruh terhadap bahasa pemrograman dan kode sumber yang digunakan. Hal ini karena pengujian dilakukan dengan melihat secara langsung mekanisme kerja internal aplikasi, sehingga membutuhkan informasi detail mengenai cara aplikasi dibangun (Verma *et al.,* 2017). Hal tersebut digambarkan secara visual pada Gambar 10.



Gambar 10. Representasi dari Pengujian *White Box* (Verma *et al.,* 2017).

a. *Unit Testing*

Unit Testing adalah pengujian yang dilakukan pada tingkat unit atau modul terkecil dari perangkat lunak untuk memastikan setiap bagian bekerja sesuai fungsinya dan telah dikembangkan sesuai dengan harapan. Pengujian ini biasanya dilakukan secara otomatis

menggunakan *framework* seperti PHP Unit, JUnit, *Pytest*, Mocha, dan lain-lain (Amin *et al.,* 2025).

b. *Integration Testing*

Integration Testing bertujuan untuk mengevaluasi bagaimana modul atau komponen dalam sistem berinteraksi satu sama lain. Pengujian ini penting untuk mendeteksi kesalahan yang mungkin muncul akibat integrasi antar modul yang sebelumnya telah diuji secara individual melalui *unit testing*. Perbedaan utama antara *unit testing* dan *integration testing* terletak pada subjek pengujian, *unit testing* fokus pada satu bagian tertentu, sedangkan *Integration testing* menguji beberapa komponen sekaligus dan memastikan semua komponen tersebut bekerja secara sinergis (Amin *et al.,* 2025).

c. *System Testing*

System testing merupakan pengujian menyeluruh terhadap perangkat lunak untuk memastikan seluruh fitur berfungsi sesuai spesifikasi yang telah ditetapkan (Amin *et al.,* 2025). *System Testing* mencakup:

1) Pengujian fungsional bertujuan memastikan setiap fitur bekerja sesuai kebutuhan.
2) Pengujian non-fungsional meliputi aspek performa, keamanan, dan kompatibilitas.

III. METODOLOGI PENELITIAN

3.1 Waktu dan Tempat

Penelitian ini dilaksanakan di Jurusan Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam (FMIPA), yang berlokasi di Jalan Prof. Dr. Ir. Sumantri Brojonegoro No. 1, Gedong Meneng, Kecamatan Rajabasa, Kota Bandar Lampung. Kegiatan penelitian berlangsung mulai dari Oktober 2025 hingga Maret 2026. Rincian waktu pelaksanaan penelitian secara lebih lengkap disajikan pada Tabel 6.

Tabel 6. Waktu Penelitian

Nama Kegiatan	**Oktober 2025**				**November 2025**				**Desember 2025**				**Januari 2026**				**Februari 2026**				**Maret 2026**	
	1	2	3	4	1	1	2	3	4	2	3	4	1	2	3	4	1	2	3	4	1	2
Penulisan Bab 1-3																						
Pengumpulan Dataset																						
Perencanaan																						
Perancangan (Usecase diagram, *Activity Diagram*, ERD, *Wireframe*)																						
Pengkodean (Integrasi Model NLP																						

ke Sistem Web)																				
Pengujian Sistem																				
Penulisan Bab 4-5																				

3.2 Alat

Alat yang digunakan dalam penelitian ini adalah sebagai berikut:

3.2.1 Perangkat Keras (*Hardware*)

Spesifikasi perangkat keras yang digunakan dalam penelitian ini adalah sebagai berikut:

1. *Processor*: 11th Gen Intel® Core™ i3-1115G4 @ 3.00GHz (2 Core, 4 Th*read*)
2. RAM: 8 GB
3. Storage: 256 GB SSD
4. System Type: 64-bit Operating System, x64-based *Processor*

3.2.2 Perangkat Lunak (*Software*)

Perangkat lunak yang digunakan dalam penelitian ini meliputi sistem operasi, platform pengembangan, serta *Library* pendukung:

1. Sistem Operasi: Windows 10 (64-bit)
2. *Google Colaboratory*: Platform cloud yang digunakan untuk menjalankan model SBERT dengan dukungan GPU gratis
3. Python: Versi 3.8 atau lebih tinggi sebagai bahasa pemrograman utama
4. Visual Studio Code: Digunakan sebagai *Text* editor untuk pengembangan kode

5. Figma: Digunakan untuk mendesain *User Interface (UI)* dan *User Experience (UX)* dari sistem berbasis web.
6. Draw.io (diagrams.net): Digunakan untuk membuat diagram pendukung dokumentasi sistem seperti *Class Diagram*, *Activity Diagram*, dan *Use Case Diagram*
7. Web Browser: Google Chrome digunakan untuk menjalankan aplikasi berbasis web
8. *MySQL*: Digunakan untuk menyimpan dan mengelola data dalam sistem pendeteksi kesamaan konteks dokumen

Library dan *Framework* Python yang digunakan:

1. Flask: *Framework* untuk pengembangan aplikasi web
2. *Sentence-Transformer*s: *Library* untuk implementasi SBERT (*Sentence*-BERT)
3. PyPDF2 dan python-docx: *Library* untuk ekstraksi teks dari dokumen PDF dan doc/docx.
4. Sastrawi: *Library* untuk melakukan *Stemming* bahasa Indonesia
5. NumPy: *Library* untuk komputasi numerik
6. Pandas: *Library* untuk pengolahan dan analisis data dan analisis data
7. Scikit-learn: *Library* yang digunakan untuk perhitungan *Cosine Similarity.*

3.3. Jenis dan Sumber Data

3.3.1 Data Primer

Data primer dalam penelitian ini diperoleh dari dataset dokumen tugas akhir mahasiswa Jurusan Ilmu Komputer yang diunduh melalui Digilib Universitas Lampung (http://digilib.unila.ac.id). Dataset ini digunakan sebagai sumber utama dalam proses pengujian dan

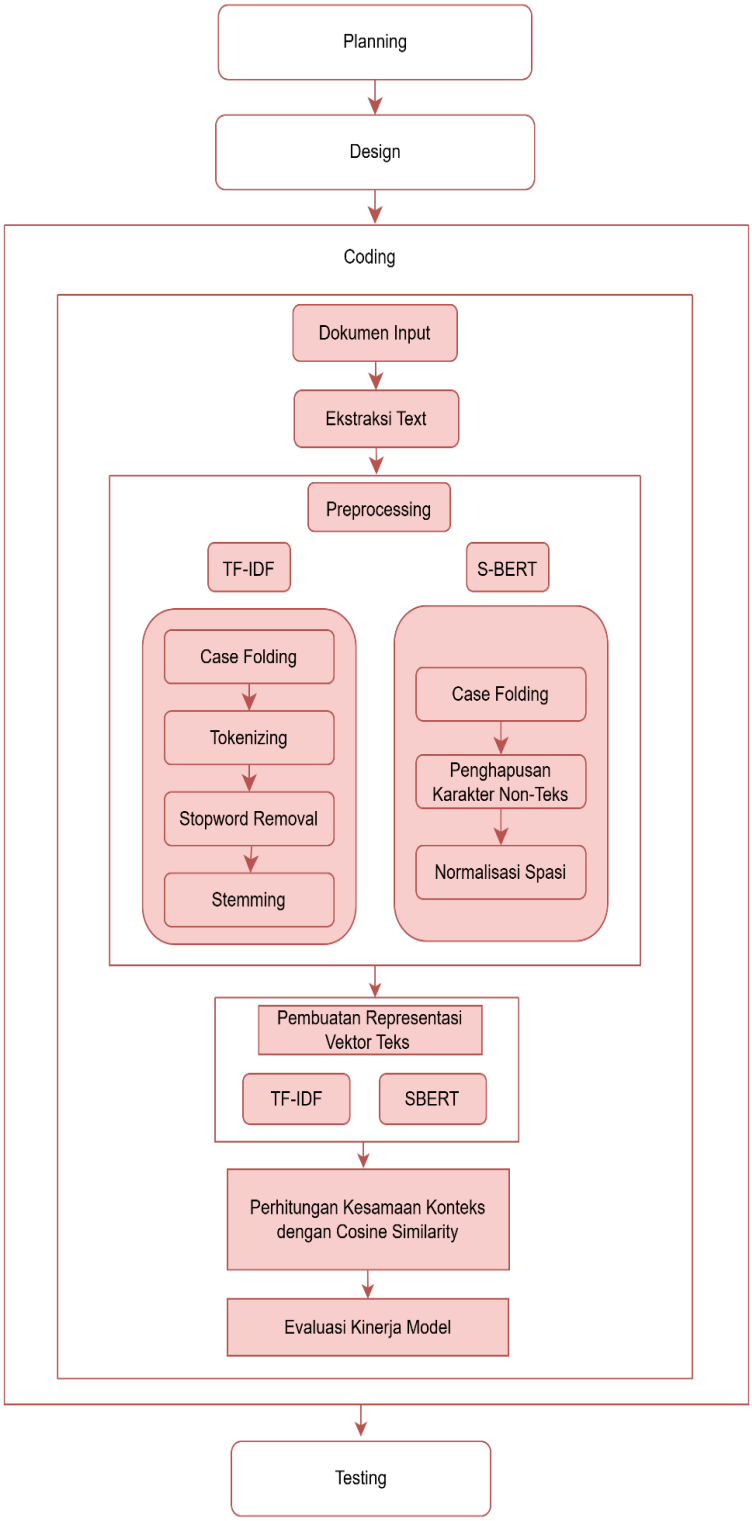
penerapan sistem pendeteksi kesamaan konteks pada dokumen akademik. Dokumen yang digunakan berupa skripsi mahasiswa tahun 2024 dengan format PDF, mencakup Bab 1 (Pendahuluan), Bab 2 (Tinjauan Pustaka), dan Bab 3 (Metodologi Penelitian).

3.3.2 Data Sekunder

Data sekunder dalam penelitian ini diperoleh melalui studi literatur dari berbagai sumber yang relevan, seperti buku, jurnal ilmiah, artikel penelitian, dan situs resmi yang membahas topik terkait pengembangan sistem informasi serta pendeteksian kesamaan dokumen. Data ini mencakup referensi mengenai teori rekayasa perangkat lunak, metode pengembangan sistem dengan model *Prototyping*, serta dokumentasi terkait teknologi yang digunakan, seperti Python, Flask, dan *MySQL*.

3.4 Metode Pengembangan Sistem

Metode pengembangan sistem yang digunakan dalam penelitian ini adalah *Extreme Programming* (XP). Metode ini dipilih karena mampu mendukung pengembangan perangkat lunak secara iteratif dan adaptif, terutama dalam kondisi di mana kebutuhan pengguna dapat berubah atau belum sepenuhnya terdefinisi. XP menekankan kolaborasi aktif antara pengembang dan pengguna, serta penyesuaian berulang pada sistem melalui tahapan perencanaan, perancangan, pengkodean, dan pengujian untuk memastikan kualitas hasil akhir. Gambar 11 menunjukkan alur metode *Extreme Programming* yang terdiri atas tahapan perencanaan (*Planning*), perancangan (*Design*), pengkodean (*Coding*), dan pengujian (*Testing*).



Gambar 11. Metode Pengembangan Sistem.

3.4.1 Perencanaan (*Planning*)

Tahap perencanaan sistem dimulai dengan pengumpulan data terkait kondisi sistem yang berjalan. Pada tahap ini dilakukan identifikasi permasalahan serta analisis kebutuhan sistem untuk memperoleh gambaran mengenai keluaran (*output*), fitur, dan fungsi yang diharapkan oleh pengguna. Hasil dari tahap perencanaan ini kemudian dijabarkan menjadi kebutuhan fungsional dan kebutuhan non-fungsional sistem. Pada sistem pendeteksi kesamaan konteks dokumen akademik ini terdapat dua peran utama, yaitu Admin dan *user*.

3.4.1.1 Kebutuhan Fungsional

a. Kebutuhan Fungsional untuk Admin
    1. Admin dapat melakukan login ke sistem.
    2. Admin dapat mengelola dokumen referensi yang digunakan sebagai pembanding (unggah dan hapus dokumen PDF).
b. Kebutuhan Fungsional untuk *user*
    1. *User* dapat mengunggah dokumen tugas akhir dalam format text, pdf, atau docx (Bab 1–3).
    2. *User* dapat melihat hasil kesamaan konteks berupa nilai *similarity* dan ranking hasil perbandingan.

3.4.1.1 Kebutuhan Non-Fungsional

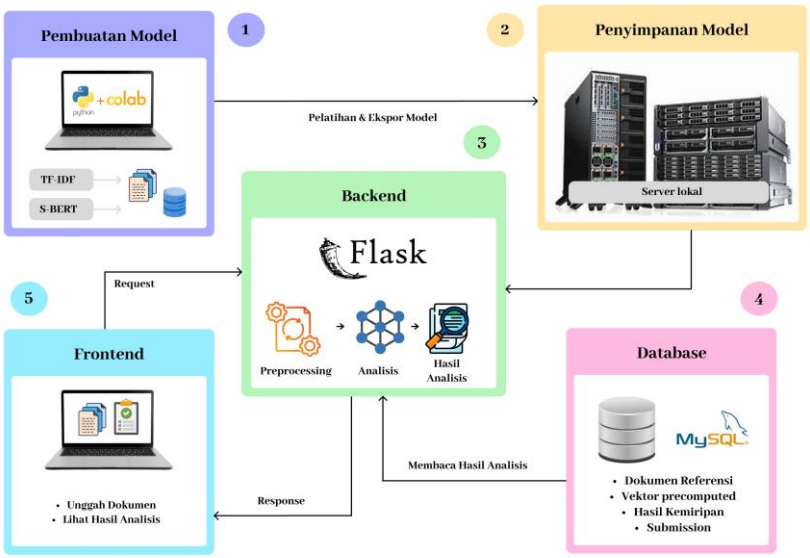
a. Sistem harus berjalan stabil tanpa mengalami kegagalan saat proses unggah maupun perhitungan.
b. Antarmuka harus mudah digunakan oleh Admin dan *User*.
c. Sistem harus mudah diperbaiki dan dikembangkan.

3.4.2 Perancangan (*Design*)

Pada tahap perancangan, hasil analisis kebutuhan diterjemahkan ke dalam bentuk pemodelan sistem. Proses ini mencakup perancangan struktur sistem, pemodelan basis data untuk menggambarkan hubungan antar data, serta perancangan tata letak antarmuka pengguna. Pemodelan sistem dilakukan menggunakan *Unified Modeling Language* (UML), yang meliputi *Use Case Diagram*, *Activity Diagram* dan ERD, sehingga memudahkan pengembang dalam memahami alur, struktur, dan komponen sistem yang akan dibangun.

3.4.2.1 Arsitektur Sistem

Arsitektur sistem dirancang untuk memastikan seluruh komponen yang telah dikembangkan dapat terintegrasi secara menyeluruh sehingga sistem mampu beroperasi secara terpadu dan saling berinteraksi secara optimal. Proses integrasi dilakukan setelah model *hybrid similarity* (TF-IDF dan SBERT) melalui tahapan pelatihan, pengujian, serta validasi hingga mencapai performa yang diharapkan. Setiap komponen sistem, mulai dari proses pembangunan dan penyimpanan model, *backend*, *database*, hingga *frontend*, disusun dalam satu alur kerja yang terintegrasi guna memenuhi kebutuhan fungsional sistem dalam mendeteksi kesamaan dokumen. Gambaran alur integrasi serta hubungan antar komponen tersebut disajikan pada Gambar 12.



Gambar 12. Arsitektur Sistem.

a. Pembuatan Model (*Model Development*)

Tahap pengembangan model dimulai untuk membangun sistem yang mampu mendeteksi kesamaan dokumen menggunakan pendekatan *hybrid similarity*. Model ini dikembangkan di Google Colab dengan bahasa pemrograman Python, karena platform ini menyediakan komputasi berbasis cloud dan dukungan GPU sehingga proses pengolahan data dan eksperimen NLP dapat berjalan lebih cepat.

Model terdiri dari tiga komponen utama, yaitu TF-IDF, *Sentence*-BERT (SBERT), dan metode *hybrid similarity*. TF-IDF digunakan untuk mengukur kesamaan dokumen berdasarkan frekuensi kata (*lexical similarity*) dengan memanfaatkan kelas Tfidf*Vectorizer* dari pustaka *scikit-learn*, dan melalui tahapan *preprocessing* seperti *case folding*, *tokenization*, penghapusan *stopwords* Bahasa Indonesia dan Inggris, serta *stemming* menggunakan Sastrawi. SBERT berfungsi untuk menangkap kesamaan makna (*semantic similarity*)

menggunakan model *pre-trained paraphrase-multilingual-MiniLM-L12-v2* dari pustaka *sentence-transformers*, dengan *preprocessing* sederhana agar konteks semantik dokumen tetap terjaga.

Sistem menggunakan dokumen referensi dari Program Studi S1 Ilmu Komputer sebagai *corpus* data, dan model diimplementasikan pada server berbasis Flask, di mana *vectorizer* dapat dilatih ulang secara dinamis sesuai dokumen referensi yang tersimpan di basis data setiap program studi.

b. *Model Storage (Local Server)*

Sistem dirancang agar setiap program studi memiliki *vectorizer* sendiri yang dilatih menggunakan dokumen referensi yang tersimpan di basis data. Pada tahap perancangan, model SBERT dimuat ke dalam memori saat server flask diinisialisasi, sedangkan untuk setiap program studi dibuat TF-IDF *vectorizer* berdasarkan seluruh dokumen referensi di tabel masing-masing. *vectorizer* yang sudah dilatih kemudian disimpan di cache memori agar dapat digunakan kembali selama sistem berjalan. Pendekatan ini memastikan kosakata (*vocabulary*) TF-IDF selalu sesuai dengan korpus dokumen terbaru.

c. *Backend* (Web Server)

Backend sistem dibangun menggunakan Flask, sebuah *framework* Python yang berfungsi sebagai pusat pemrosesan dan integrasi seluruh modul dalam sistem. Arsitektur *Backend* mengikuti pola *Model-View-Controller* (MVC), sehingga logika, pengelolaan data, dan antarmuka pengguna dapat dipisahkan secara terstruktur. *Backend* bertanggung jawab atas manajemen dokumen referensi, pemrosesan dokumen uji, deteksi kesamaan, serta interaksi dengan basis data.

Dalam manajemen dokumen referensi, sistem dirancang untuk menerima dan memvalidasi berkas yang diunggah oleh administrator. Proses ekstraksi teks dilakukan dari berbagai format seperti PDF, DOCX, dan TXT menggunakan pustaka PyPDF2 (dengan fallback ke pdfplumber) serta python-docx. Dokumen hasil ekstraksi disimpan dalam tabel terpisah sesuai program studi. Selanjutnya, sistem melakukan *pre-computing*, meliputi *preprocessing* serta pembentukan vektor TF-IDF dan *embedding* SBERT untuk setiap dokumen referensi.

Pada pemrosesan dokumen uji, *backend* menerima dokumen yang diunggah pengguna, melakukan validasi format dan ukuran (maksimal 10 MB), mengekstrak teks, dan menyimpan data ke basis data sebelum diproses lebih lanjut. Deteksi kemiripan menggunakan pendekatan *Fast Similarity* (*Precomputed Vectors*), di mana dokumen uji diproses melalui dua jalur *preprocessing*, TF-IDF dan SBERT. Vektor TF-IDF dibuat menggunakan *vectorizer* sesuai program studi, sementara *embedding* SBERT dihasilkan dari model *sentence-transformers*. Nilai kemiripan dihitung menggunakan *cosine similarity* dan dikombinasikan melalui metode *hybrid similarity* dengan bobot seimbang antara TF-IDF dan SBERT.

d. *Database* (*MySQL*)

Basis data *MySQL* dirancang sebagai penyimpanan terpusat bagi seluruh data dalam sistem, menggunakan model relasional untuk memisahkan data berdasarkan program studi sekaligus mendukung mekanisme *pre-computing*, sehingga proses deteksi kesamaan dokumen menjadi lebih cepat. Skema basis data terdiri dari beberapa tabel utama yang saling terintegrasi.

Tabel *users* menyimpan data pengguna, termasuk peran administrator, afiliasi program studi (D3 Manajemen Informatika, S1 Ilmu Komputer, dan S1 Sistem Informasi), serta informasi waktu pembuatan dan pembaruan data.

Dokumen referensi dikelola dalam tabel terpisah untuk setiap program studi, yaitu reference_documents_mi, reference_documents_ik, dan reference_documents_si. Selain itu, tabel ini juga menyimpan hasil *pre-computing*, berupa teks yang telah melalui tahap *preprocessing* dan representasi vektor TF-IDF serta *embedding* SBERT dalam format JSON. Penyimpanan vektor ini bertujuan untuk mengurangi kebutuhan perhitungan ulang saat deteksi kesamaan dilakukan. Relasi antara dokumen referensi dan pengguna yang mengunggahnya diatur menggunakan foreign key, serta dilengkapi atribut is_active untuk menandai status dokumen.

Dokumen uji mahasiswa disimpan pada tabel *submission*s, yang memuat informasi program studi, metadata dokumen, status pemrosesan, serta waktu pemrosesan. Hasil perhitungan kesamaan disimpan pada tabel similarity_results, yang mencakup relasi antara dokumen uji dan dokumen referensi, skor *hybrid* (rentang 0–1), persentase kesamaan (0–100%), komponen skor TF-IDF dan SBERT untuk analisis, metode yang digunakan, serta waktu pencatatan hasil.

e. *Frontend*

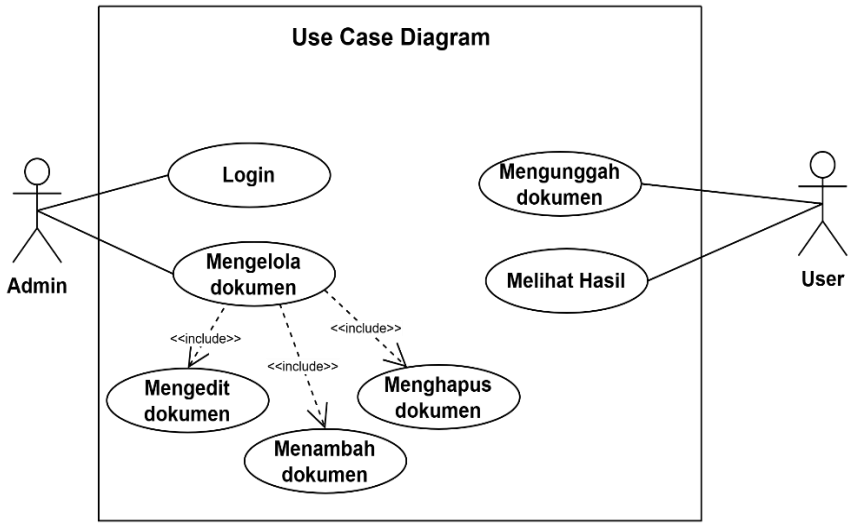
Frontend dirancang sebagai antarmuka pengguna berbasis web yang berfungsi sebagai media interaksi antara pengguna dan sistem. Antarmuka ini dikembangkan menggunakan HTML, CSS, dan JavaScript untuk menghasilkan tampilan yang responsif pada berbagai perangkat. Server menyiapkan halaman web secara dinamis sebelum

dikirim ke browser pengguna, menggunakan template engine Jinja2 yang terintegrasi dengan *framework* Flask.

Pada sisi pengguna (mahasiswa), sistem menyediakan halaman unggah dokumen yang dilengkapi dengan validasi format berkas (PDF, DOCX, dan TXT) serta batas ukuran maksimum 10 MB. Pengguna juga diwajibkan memilih program studi sebelum dokumen diproses. Setelah unggah selesai, sistem secara otomatis menghitung tingkat kesamaan dan menampilkan halaman hasil yang memuat sepuluh dokumen referensi dengan skor kemiripan tertinggi. Informasi yang ditampilkan meliputi peringkat, judul, penulis, tahun, skor *hybrid* dalam bentuk persentase, serta rincian komponen skor TF-IDF dan SBERT sebagai bentuk transparansi perhitungan. Selain itu, sistem menyediakan halaman detail kesamaan yang menampilkan perbandingan teks antara dokumen uji dan dokumen referensi secara berdampingan, disertai fitur penyorotan (*highlight*) pada bagian teks yang mirip.

Pada sisi administrator, sistem menyediakan halaman login dengan mekanisme manajemen sesi. Fitur pengelolaan dokumen referensi mencakup operasi *create*, *read*, *update* dan *delete* (CRUD) berdasarkan program studi. *Dashboard* administrator menampilkan informasi statistik sistem, seperti jumlah dokumen referensi pada setiap program studi, jumlah dokumen uji (*submission*), serta visualisasi distribusi dokumen berdasarkan tahun.

3.4.2.2 Rancangan *Use Case Diagram*

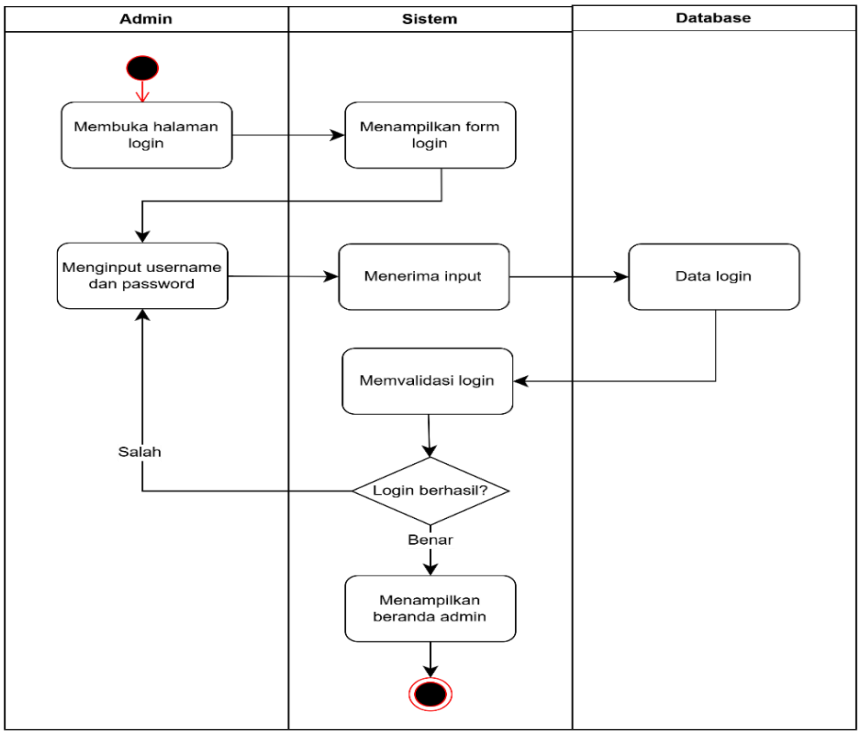


Gambar 13. *Use Case Diagram*.

Gambar 13 menampilkan *use case diagram* yang menggambarkan fungsionalitas sistem dari sudut pandang pengguna. Sistem melibatkan dua aktor, yaitu admin dan *user*. Admin memiliki akses penuh, meliputi login, dan mengelola dokumen, sedangkan *user* hanya dapat mengunggah dokumen dan melihat hasil. Pada use case mengelola dokumen, terdapat tiga fungsi turunan dengan relasi <<include>>, yaitu mengedit, menambah, dan menghapus dokumen. Hal ini menunjukkan bahwa ketiga fungsi tersebut merupakan bagian dari proses pengelolaan dokumen dan hanya dapat dilakukan oleh Admin.

3.4.2.3 Rancangan *Activity Diagram*

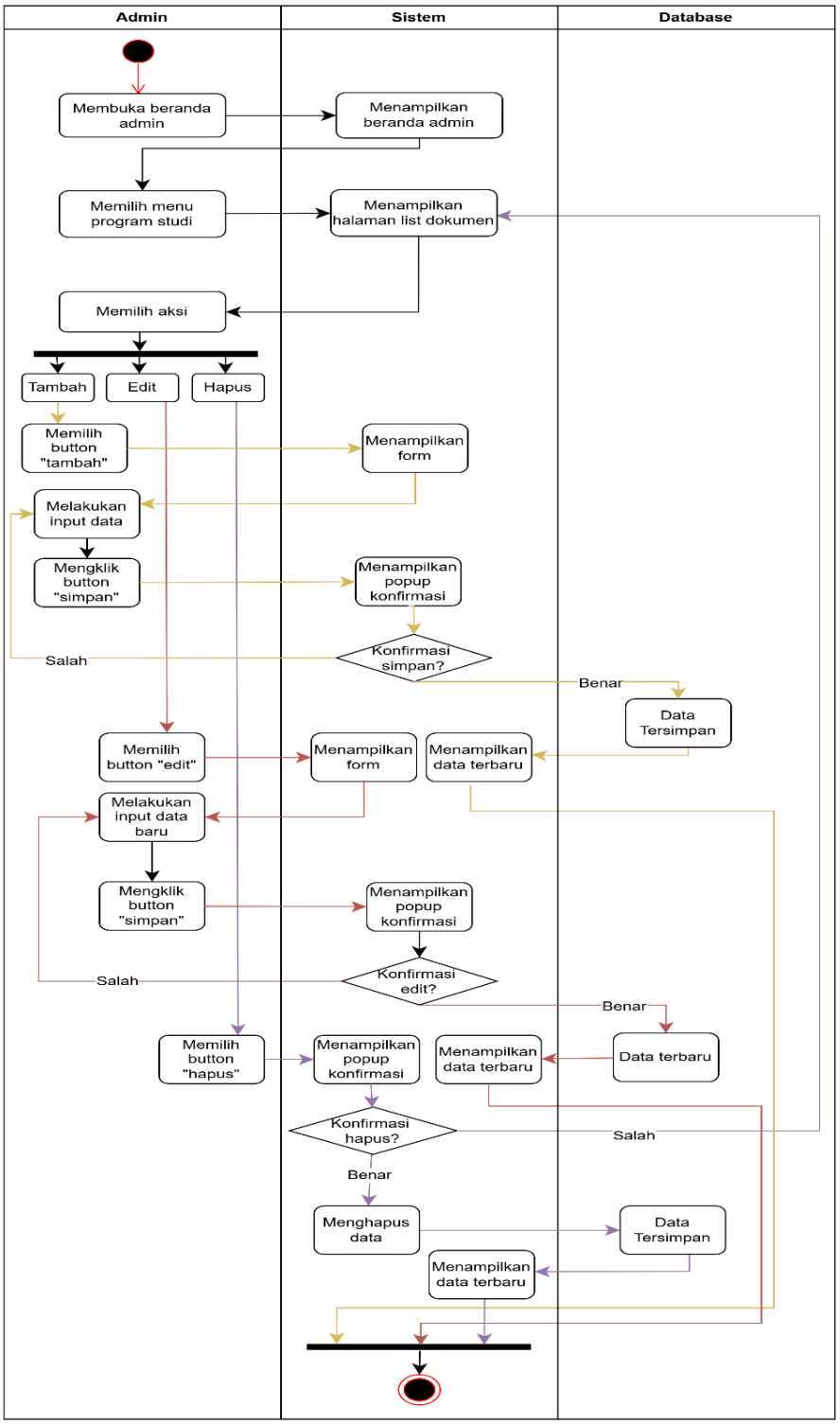
a. *Activity Diagram* Admin Melakukan *Login*



Gambar 14. *Activity Diagram* Admin Melakukan *Login*.

Gambar 14 menyajikan *activity diagram* yang menggambarkan alur aktivitas admin dalam melakukan proses login ke dalam sistem. Diagram ini terdiri dari tiga komponen utama, yaitu Admin, Sistem, dan *database*. Proses diawali ketika admin mengakses halaman login, kemudian sistem merespons dengan menampilkan form login. Selanjutnya, admin memasukkan *username* dan *password*, lalu sistem menerima *input* tersebut dan secara bersamaan mengambil data kredensial yang tersimpan di *database* untuk proses verifikasi. Hasil verifikasi menentukan alur berikutnya, jika data tidak valid, admin dikembalikan ke halaman login, sedangkan jika valid, sistem menampilkan halaman beranda admin dan proses login selesai.

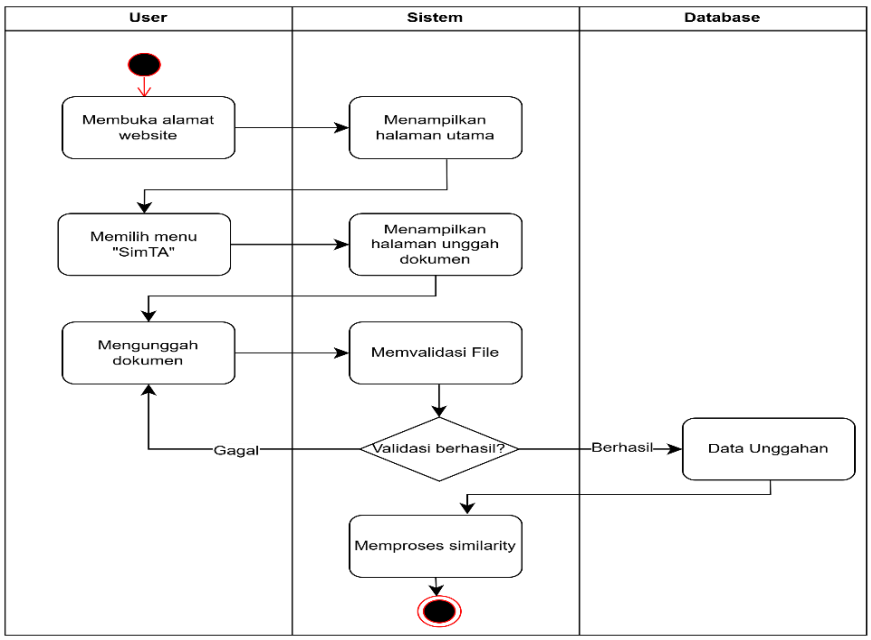
b. *Activity Diagram* Admin Mengelola Dokumen



Gambar 15. *Activity Diagram* Admin Mengelola Dokumen.

Gambar 15 menunjukkan *activity diagram* yang menggambarkan alur aktivitas admin dalam mengelola dokumen, dengan melibatkan tiga komponen yaitu Admin, Sistem, dan *database*. Proses dimulai ketika admin membuka beranda dan memilih menu program studi, sehingga sistem menampilkan daftar dokumen. Selanjutnya, admin dapat memilih salah satu aksi, yaitu tambah, edit, atau hapus dokumen. Pada aksi tambah, admin mengisi form lalu menyimpan data, yang kemudian dikonfirmasi sebelum disimpan ke *database*. Pada aksi edit, sistem menampilkan data yang dapat diperbarui, kemudian dilakukan proses simpan dengan konfirmasi. Sedangkan pada aksi hapus, sistem menampilkan konfirmasi sebelum data dihapus dari *database*. Jika konfirmasi ditolak, proses dibatalkan dan admin kembali ke halaman daftar dokumen.

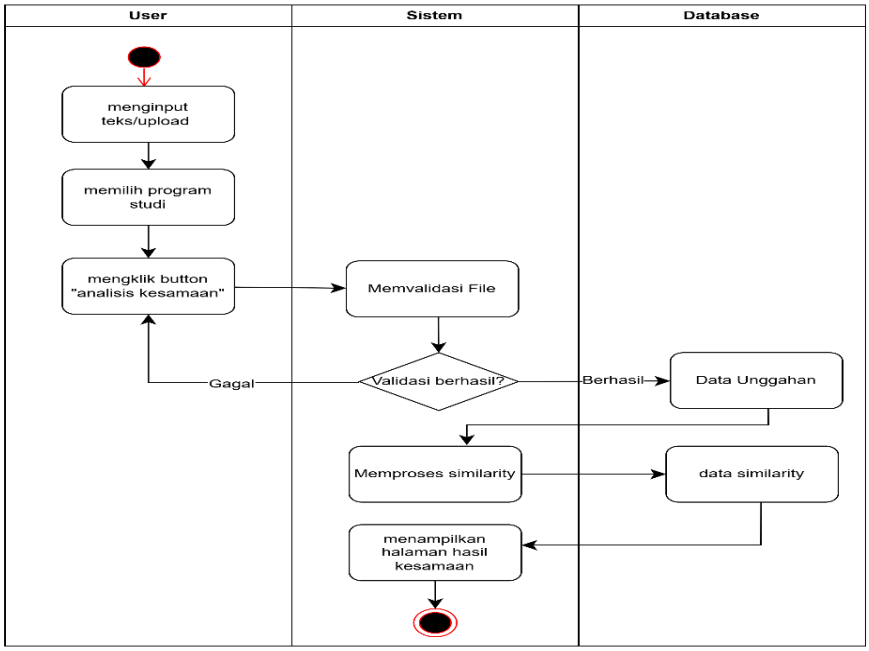
c. *Activity Diagram User* Mengunggah dokumen



Gambar 16. *Activity Diagram User* Mengunggah Dokumen.

Gambar 16 menunjukkan *activity diagram* proses *user* dalam mengunggah dokumen ke sistem yang melibatkan *user*, Sistem, dan *database*. Proses dimulai dari *user* membuka *website*, memilih menu "SimTA", lalu mengunggah dokumen. Sistem kemudian melakukan validasi file. Jika gagal, *user* mengulang proses unggah, sedangkan jika berhasil, data disimpan ke *database* dan sistem langsung menghitung *similarity* untuk mendeteksi kemiripan dokumen.

d. *Activity Diagram User* Melihat Hasil

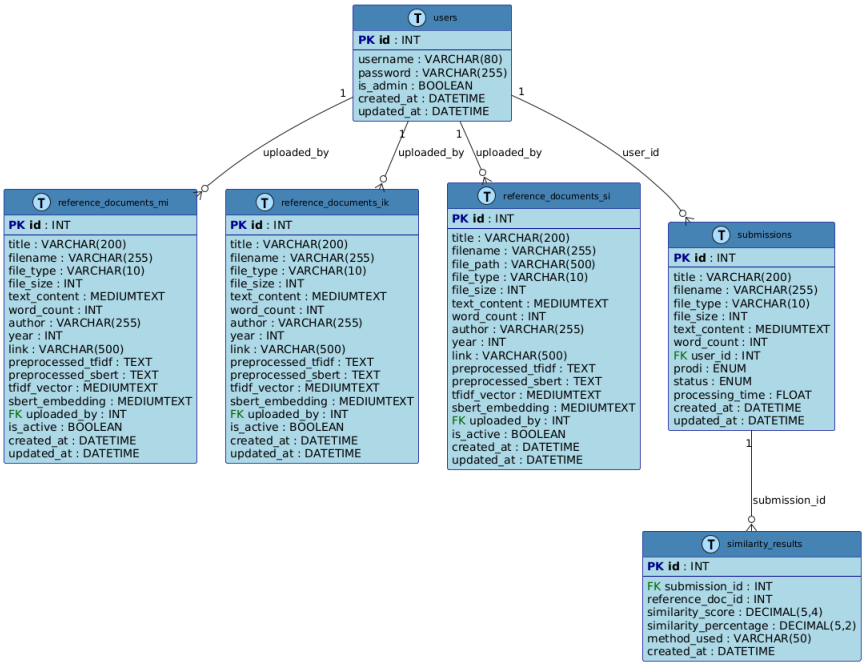


Gambar 17. *Activity Diagram User* Melihat Hasil.

Gambar 17 menunjukkan *Activity Diagram* yang menggambarkan alur *user* dalam melihat hasil analisis kesamaan dokumen, dengan melibatkan *user*, Sistem, dan *database*. Proses dimulai saat *user* menginput teks atau mengunggah dokumen, memilih program studi, lalu menekan tombol "Analisis Kesamaan". Sistem kemudian melakukan validasi file. Jika validasi gagal, *user* harus mengulang

proses. Jika berhasil, data disimpan ke *database*, sistem menghitung *similarity*, dan hasilnya ditampilkan kepada *user*.

3.4.2.4 Rancangan *Entity Relationship Diagram*



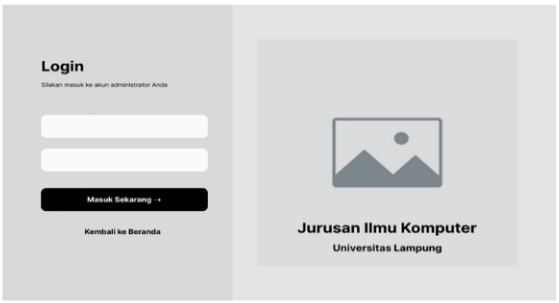
Gambar 18. *Entity Relationship Diagram*.

Gambar 18 menampilkan *Entity Relationship Diagram* (ERD) yang menggambarkan struktur *database* sistem secara keseluruhan, yang terdiri dari enam tabel yang saling berelasi. Tabel *users* berperan sebagai tabel utama yang menyimpan data pengguna, seperti *user* name, *password*, status admin, serta waktu pembuatan dan pembaruan akun. Tabel ini terhubung dengan tiga tabel dokumen referensi, yaitu reference_documents_mi, reference_documents_ik, dan reference_documents_si, yang masing-masing menyimpan dokumen berdasarkan program studi dengan struktur data yang serupa. Tabel submissions digunakan untuk menyimpan dokumen yang diunggah oleh *user* untuk dianalisis, serta memiliki relasi ke tabel *users* melalui *foreign key user*_id. Sementara itu, tabel similarity_results

menyimpan hasil perhitungan kemiripan dokumen, yang terhubung dengan tabel submissions melalui *foreign key* submission_id, serta memuat informasi skor dan metode yang digunakan, seperti TF-IDF dan SBERT.

3.4.2.5 *Wireframe* Sistem (*Interface*)

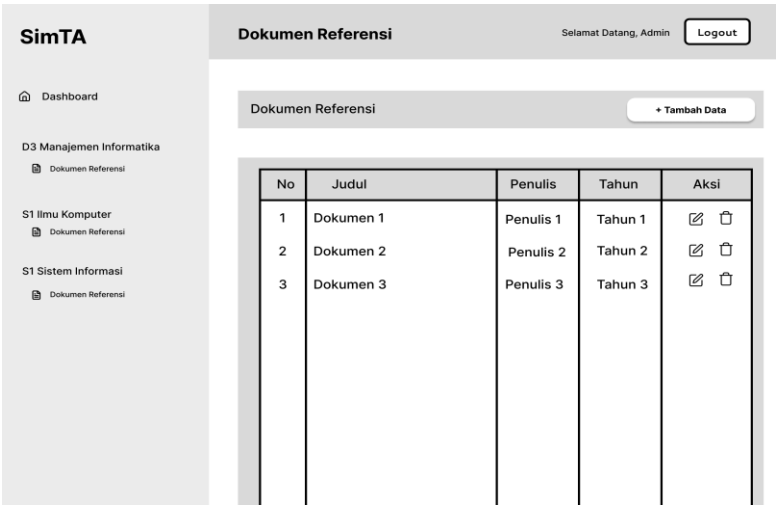
a. *Wireframe* Halaman Admin Melakukan *Login*



Gambar 19. *Wireframe* Halaman Admin Melakukan *Login*.

Gambar 19 menampilkan *Wireframe* halaman login yang digunakan admin untuk mengakses sistem. Halaman ini terbagi menjadi dua bagian utama secara horizontal. Bagian kiri berisi form login yang mencakup judul, teks instruksi, kolom input *username* dan *password*, tombol "Masuk Sekarang", serta tautan "Kembali ke Beranda". Sementara itu, bagian kanan menampilkan logo atau ilustrasi identitas beserta teks "Jurusan Ilmu Komputer Universitas Lampung" sebagai identitas sistem.

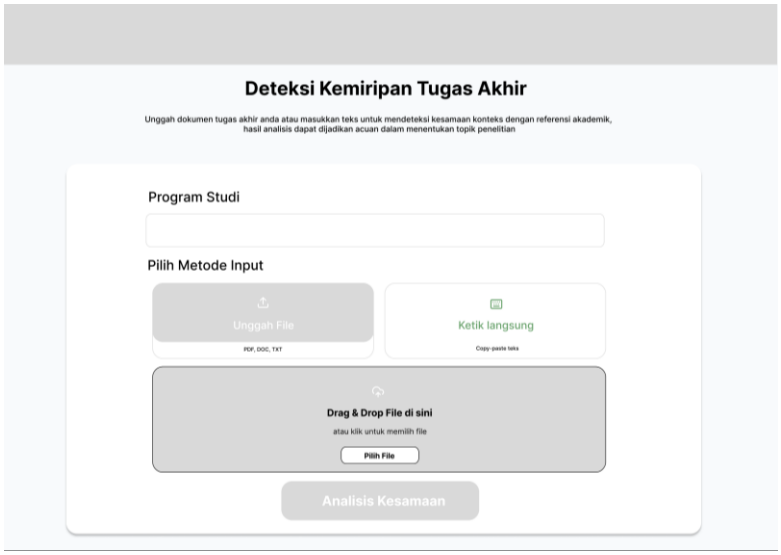
b. *Wireframe* Halaman Admin Mengelola Dokumen



Gambar 20. *Wireframe* Halaman Admin Mengelola Dokumen.

Gambar 20 menampilkan *wireframe* halaman pengelolaan dokumen referensi yang hanya dapat diakses oleh admin. Halaman ini terdiri dari dua bagian utama, yaitu sidebar navigasi di sebelah kiri dan area konten utama di sebelah kanan. Sidebar berisi menu *Dashboard* serta kategori dokumen berdasarkan program studi. Pada bagian kanan atas terdapat informasi admin yang sedang login beserta tombol logout. Sementara itu, area konten utama menampilkan tabel daftar dokumen referensi dengan kolom No, Judul, Penulis, Tahun, dan Aksi. Pada kolom aksi tersedia fitur edit dan hapus, serta tombol "+ Tambah Data" di bagian atas tabel untuk menambahkan dokumen baru.

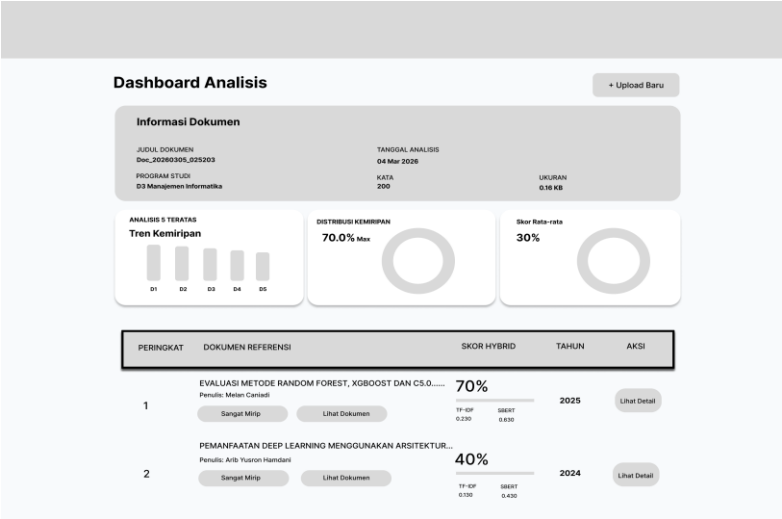
c. *Wireframe* Halaman *User* Mengunggah Dokumen



Gambar 21. *Wireframe* Halaman *User* Mengunggah Dokumen.

Gambar 21 menampilkan *wireframe* halaman utama yang digunakan oleh *user* untuk mengunggah. Halaman ini berjudul "Deteksi Kemiripan Tugas Akhir" dan dilengkapi dengan deskripsi singkat mengenai fungsinya. Terdapat tiga komponen input utama, yaitu pilihan Program Studi, metode input (unggah file atau ketik langsung), serta area *drag and drop* untuk mengunggah file atau memilih file secara manual. Setelah seluruh input diisi, *user* dapat menekan tombol "Analisis Kesamaan" untuk memulai proses deteksi kemiripan dokumen dengan dokumen referensi yang tersedia dalam sistem.

d. *Wireframe* Halaman *User* Melihat Hasil



Gambar 22. *Wireframe* Halaman *User* Melihat Hasil.

Gambar 22 menampilkan *wireframe* halaman *Dashboard* Analisis yang menyajikan hasil deteksi kesamaan dokumen yang telah diunggah oleh *user*. Halaman ini terdiri dari beberapa bagian utama. Bagian pertama adalah Informasi Dokumen yang memuat detail dokumen, seperti judul, program studi, tanggal analisis, jumlah kata, dan ukuran file. Bagian kedua menampilkan ringkasan hasil analisis dalam bentuk tiga kartu, yaitu tren kemiripan (grafik batang 5 dokumen teratas), distribusi kemiripan dengan nilai maksimum, serta skor rata-rata kesamaan. Bagian ketiga berupa tabel peringkat dokumen referensi yang menampilkan dokumen paling mirip secara berurutan, dilengkapi informasi judul, penulis, tingkat kemiripan, skor (hybrid, TF-IDF, dan SBERT), tahun terbit, serta tombol "Lihat Detail". Selain itu, tersedia tombol "+ *Upload* Baru" untuk melakukan analisis dokumen baru.

3.4.3 Pengkodean (*Coding*)

Tahap pengkodean merupakan fase implementasi dalam pengembangan sistem, yaitu tahap ketika hasil perancangan dan *user stories* yang telah dirumuskan sebelumnya diterjemahkan ke dalam bentuk kode program yang dapat dijalankan. Pada fase ini, seluruh kebutuhan fungsional dan nonfungsional mulai direalisasikan melalui proses pembangunan modul-modul sistem secara terstruktur.

Implementasi sistem dibagi ke dalam dua fase utama. Fase pertama adalah *Model Development* yang dilakukan di Google Colab untuk membangun, melatih, menguji, dan memvalidasi model *hybrid similarity*. Fase kedua adalah *Application Development*, yang mencakup pengembangan *Backend* dan *Frontend*. *Backend* dibangun menggunakan *framework* Flask untuk mengelola logika bisnis, pemrosesan data, serta integrasi dengan basis data, sedangkan *Frontend* dikembangkan sebagai antarmuka pengguna berbasis web untuk mendukung interaksi antara pengguna dan sistem. Pembagian ini bertujuan untuk menjaga modularitas, memudahkan proses pemeliharaan, serta memastikan setiap komponen sistem dapat dikembangkan dan diuji secara terpisah sebelum diintegrasikan secara menyeluruh.

3.4.3.1 Pengembangan Logika Sistem (*Backend*)

Backend bertugas mengatur logika sistem, memproses dokumen, menghitung kesamaan isi, menyimpan data, dan mengelola login pengguna. *Backend* dibangun menggunakan Python dengan *framework* Flask, sehingga memudahkan pengembangan serta pengelolaan basis data melalui *MySQL*. Di sini, *Backend* juga menjalankan fungsi utama sistem, mulai dari unggah dokumen, mengekstrak teks, membersihkan dan mempersiapkan data,

menghitung kesamaan menggunakan *Sentence*-BERT (SBERT), hingga menyimpan hasil analisis ke basis data.

a. Dokumen *Input*

Tahap awal alur kerja sistem dimulai ketika pengguna mengunggah dokumen tugas akhir yang akan dianalisis tingkat kesamaan konteksnya. Sistem mendukung unggahan dalam format teks (.txt), PDF (.pdf), serta Microsoft Word (.docx) dengan batas ukuran maksimum 10 MB. Dokumen yang diunggah harus memiliki konten tekstual yang dapat diekstrak, serta idealnya mencakup Bab 1 (Pendahuluan), Bab 2 (Tinjauan Pustaka), dan Bab 3 (Metodologi Penelitian).

Proses unggah dilakukan melalui antarmuka web yang menyediakan fitur *drag-and-drop* maupun tombol “Pilih File”. Sistem kemudian melakukan validasi terhadap format dan ukuran file, memberikan konfirmasi apabila unggahan berhasil, atau menampilkan pesan kesalahan apabila ditemukan ketidaksesuaian. Setelah lolos validasi, dokumen disimpan sementara di server dengan nama file unik berbasis *timestamp* untuk mencegah konflik. Dokumen tersebut selanjutnya digunakan sebagai masukan pada tahap ekstraksi teks dan pemrosesan lanjutan.

b. Ekstraksi Teks

Setelah dokumen berhasil diunggah dan divalidasi, langkah selanjutnya adalah melakukan ekstraksi teks dari file yang telah disimpan. Tahap ini bertujuan mengubah dokumen yang masih berbentuk data tidak terstruktur menjadi teks mentah (*plain text*) yang dapat diproses lebih lanjut oleh algoritma *Natural Language Processing* (NLP). Proses ekstraksi dilakukan menggunakan fungsi

khusus yang disesuaikan dengan masing-masing format file untuk memastikan seluruh konten tekstual dapat terbaca secara optimal.

Untuk dokumen berformat PDF, sistem menggunakan pustaka PyPDF2 sebagai metode utama. Ekstraksi dilakukan dengan membaca setiap halaman menggunakan objek Pdf*Read*er, kemudian menggabungkan teks hasil ekstraksi dari tiap halaman ke dalam satu variabel string. Apabila terjadi kesalahan atau teks tidak dapat diekstrak secara optimal, sistem secara otomatis menggunakan metode *fallback* dengan pustaka pdfplumber. Pendekatan ini meningkatkan keandalan proses ekstraksi karena beberapa file PDF memiliki struktur internal yang berbeda-beda.

Pada dokumen berformat Word (.docx), proses ekstraksi dilakukan menggunakan pustaka python-docx. Sistem membaca seluruh paragraf dalam dokumen melalui objek Document, kemudian mengambil teks dari setiap paragraf yang tidak kosong dan menggabungkannya menjadi satu string utuh. Metode ini memungkinkan sistem memperoleh isi dokumen tanpa elemen format seperti gaya huruf atau tata letak.

Sementara itu, untuk file berformat .txt, ekstraksi dilakukan menggunakan fungsi bawaan Python tanpa memerlukan pustaka tambahan. File dibuka dengan mode baca ("r") menggunakan *encoding* UTF-8 dan opsi errors="ignore" untuk menghindari kegagalan pembacaan akibat karakter yang tidak dikenali. Seluruh isi file kemudian dibaca dan disimpan sebagai string.

Hasil akhir dari proses ekstraksi ini berupa teks murni yang mencakup seluruh konten dokumen, termasuk judul, paragraf isi, serta elemen tekstual lainnya. Teks tersebut disimpan dalam bentuk string dan selanjutnya digunakan sebagai masukan pada tahap *preprocessing*

untuk proses analisis kesamaan dokumen menggunakan model *hybrid similarity*.

c. Pra-Pemrosesan Data Teks (*Text Pre-processing*)

Teks yang telah diekstraksi kemudian diproses melalui tahap pra-pemrosesan (*Text Preprocessing*) untuk mempersiapkan data agar dapat diolah oleh model *Natural Language Processing* (NLP). Pada sistem ini, *preprocessing* dirancang dalam dua mode berbeda, karena karakteristik metode TF-IDF dan SBERT tidak sama. Pemisahan ini bertujuan agar masing-masing metode memperoleh representasi teks yang paling optimal sesuai pendekatannya.

Mode 1 merupakan *preprocessing* untuk TF-IDF yang bersifat agresif karena metode ini berbasis frekuensi kata (*bag-of-words*), sehingga teks perlu dibersihkan secara menyeluruh agar hanya menyisakan kata-kata kunci yang bermakna. Proses diawali dengan *case folding*, yaitu mengubah seluruh huruf menjadi huruf kecil untuk menjaga konsistensi penulisan. Selanjutnya dilakukan pembersihan angka dan tanda baca menggunakan *regular expression* serta *string.punctuation*, kemudian teks dipecah menjadi token melalui proses *tokenization*. Tahap berikutnya adalah *stopword removal*, yaitu menghapus kata-kata umum yang tidak memiliki kontribusi signifikan terhadap makna dokumen.

Daftar stopword bahasa Indonesia diperoleh menggunakan pustaka Sastrawi melalui *StopWordRemoverFactory*, lalu digabungkan dengan stopword bahasa Inggris dari NLTK agar sistem mampu menangani dokumen campuran bahasa. Setelah itu dilakukan *Stemming* menggunakan *StemmerFactory* dari Sastrawi untuk mengembalikan kata berimbuhan ke bentuk dasarnya, sehingga

variasi kata seperti “berjalan”, “berjalan-jalan”, dan “perjalanan” direduksi menjadi “jalan”. Sebagai contoh, teks “Sistem Informasi Akademik berbasis Web menggunakan *Framework* Laravel” akan diproses menjadi “sistem informasi akademik basis web guna *framework* laravel”. Hasil akhir dari tahap ini adalah teks yang telah dibersihkan secara agresif dan siap dikonversi menjadi representasi vektor menggunakan metode TF-IDF.

Mode 2 merupakan *preprocessing* untuk SBERT yang bersifat minimal. Berbeda dengan TF-IDF yang berbasis frekuensi kata, SBERT merupakan model berbasis representasi semantik sehingga konteks kalimat harus tetap dipertahankan. Oleh karena itu, *preprocessing* pada tahap ini hanya meliputi *case folding* dengan mengubah seluruh huruf menjadi huruf kecil, penghapusan karakter non-teks menggunakan *regular expression* tanpa menghilangkan struktur kalimat secara berlebihan, serta normalisasi spasi agar tidak terdapat spasi ganda. Pada mode ini tidak dilakukan *Stopword Removal* maupun *Stemming*, karena penghapusan kata atau perubahan bentuk kata dapat mengurangi informasi kontekstual yang penting bagi model.

Sebagai contoh, teks “Sistem Informasi Akademik berbasis Web menggunakan *Framework* Laravel” akan diproses menjadi “sistem informasi akademik berbasis web menggunakan *framework* laravel”. Pemisahan mode *preprocessing* ini dilakukan karena perbedaan karakteristik kedua metode, di mana TF-IDF yang berbasis frekuensi kata memerlukan pembersihan agresif untuk menekankan kata kunci yang relevan, sedangkan SBERT yang berbasis pemahaman semantik memerlukan *preprocessing* minimal agar makna kalimat tetap utuh. Dengan pendekatan ini, kedua metode dapat bekerja secara optimal sebelum digabungkan dalam perhitungan *hybrid similarity*.

d. Pembuatan Representasi Vektor Teks (*Text Embedding*)

Tahap ini merupakan proses mengubah teks yang telah dibersihkan menjadi representasi numerik (vektor) agar dapat diolah oleh komputer untuk perhitungan kesamaan. Penelitian ini menggunakan dua pendekatan representasi vektor yang berbeda untuk kemudian dibandingkan kinerjanya, yaitu TF-IDF sebagai metode baseline dan SBERT sebagai metode yang diusulkan.

1). Pembentukan Vektor TF-IDF

Pendekatan pertama menggunakan metode TF-IDF (*Term Frequency–Inverse Document Frequency*) sebagai *baseline*. Pada tahap ini, teks yang telah melalui *preprocessing* dikonversi menjadi vektor menggunakan Tfidf*Vectorizer* dari scikit-learn. Metode ini memberikan bobot pada setiap kata berdasarkan frekuensi kemunculannya dalam sebuah dokumen serta tingkat penyebarannya di seluruh korpus. Dimensi vektor bergantung pada jumlah kata unik (*vocabulary*) setelah *preprocessing*.

Misalnya, jika terdapat 5.000 kata unik, maka setiap dokumen direpresentasikan sebagai vektor berdimensi 5.000, di mana setiap nilai menunjukkan bobot TF-IDF dari kata tersebut. Selanjutnya, vektor TF-IDF yang telah diperoleh dinormalisasi menggunakan metode L2 normalisasi (*Euclidean norm*) sehingga setiap vektor memiliki nilai panjang (*magnitude*) sebesar 1. Proses normalisasi ini bertujuan agar perbandingan antar dokumen tidak dipengaruhi oleh perbedaan panjang teks, tetapi lebih berfokus pada distribusi dan bobot kata pada masing-masing dokumen. Tahap ini menghasilkan matriks TF-IDF yang berisi representasi vektor untuk dokumen *input* dan seluruh dokumen referensi dalam *database*.

2). Pembentukan Embedding SBERT

Pendekatan kedua menggunakan model *Sentence*-BERT (SBERT) dengan varian *paraphrase-multilingual-MiniLM-L12-v2*, yaitu model *pre-trained* dari *Library Sentence-Transformer*s. Pada penelitian ini, model digunakan secara langsung (*off-the-shelf*) tanpa proses *fine-tuning*. Setiap dokumen yang telah melalui tahap *preprocessing* diproses menggunakan fungsi model.encode(). Model kemudian menghasilkan vektor berdimensi tetap 384, terlepas dari panjang teks atau jumlah *vocabulary*. Dengan demikian, baik dokumen pendek berisi 100 kata maupun dokumen panjang berisi 1.000 kata akan direpresentasikan oleh vektor berdimensi sama.

Setelah proses *encoding* menggunakan model SBERT (*Sentence-BERT*) Sentence-BERT *paraphrase-multilingual-MiniLM-L12-v2*, setiap dokumen direpresentasikan ke dalam bentuk vektor embedding berdimensi 384. Vektor embedding tersebut kemudian dinormalisasi menggunakan metode L2 normalisasi yang mengacu pada konsep *L2-norm* atau norma Euclidean, sehingga setiap vektor memiliki panjang (*magnitude*) bernilai 1. Proses normalisasi ini bertujuan untuk menjaga konsistensi dalam perhitungan kemiripan antar dokumen, karena perbandingan yang dilakukan tidak dipengaruhi oleh besar kecilnya nilai vektor, melainkan berdasarkan arah dan distribusi vektornya. Selanjutnya, vektor yang telah dinormalisasi digunakan dalam proses perhitungan kemiripan menggunakan metode *cosine similarity* yang akan dibahas lebih lanjut pada subbab berikutnya.

e. Perhitungan Kesamaan Konteks (*Context Similarity Calculation*)

Setelah dokumen input dan seluruh dokumen referensi dikonversi ke dalam bentuk vektor numerik, tahap berikutnya adalah menghitung

tingkat kesamaan menggunakan pendekatan *hybrid similarity*. Pendekatan ini mengombinasikan dua metode, yaitu TF-IDF dan SBERT, sehingga mampu menangkap kesamaan berbasis kata kunci maupun kesamaan berbasis makna semantik.

1). Perhitungan *Cosine Similarity* TF-IDF

Pada tahap awal, sistem menghitung skor kesamaan mentah (*raw similarity Score*) untuk setiap metode dengan menggunakan pendekatan *Cosine Similarity*. Pada metode TF-IDF, perhitungan ini dilakukan melalui fungsi cosine_similarity() dari pustaka *scikit-learn*, yang mengukur nilai kosinus sudut antara vektor dokumen kueri dan seluruh vektor dokumen referensi dalam matriks TF-IDF. Hasil pengukuran berupa nilai dalam rentang 0 hingga 1 yang mencerminkan tingkat kemiripan berdasarkan kesamaan kata kunci. Perhitungan *cosine similarity* antara vektor dokumen kueri Q dan vektor dokumen referensi D dilakukan menggunakan persamaan 11 berikut:

$$\text{cosine\_similarity (Q,D)} = \frac{\text{Q.D}}{||Q|| \text{ x } ||\text{D}|} \qquad (11)$$

Pada persamaan 11 tersebut, Q.D merupakan hasil perkalian *Dot Product* antara vektor kueri dan vektor referensi. Simbol Q, menyatakan norma atau panjang vektor Q sedangkan D menyatakan norma vektor D.

2). Perhitungan *Cosine Similarity* SBERT

Pada pendekatan SBERT, sistem memanfaatkan fungsi util.cos_sim() dari pustaka *sentence-transformers*, atau dapat juga menggunakan fungsi cosine_similarity() dari *scikit-learn* sebagai alternatif, yang menghasilkan nilai kesamaan serupa. Kedua metode ini memberikan

skor *cosine similarity* secara langsung tanpa memerlukan proses modifikasi tambahan. Perbedaannya terletak pada representasi data, TF-IDF mencerminkan kesamaan leksikal, sedangkan SBERT mencerminkan kesamaan semantik. Rumus *cosine similarity* yang digunakan sama dengan TF-IDF, namun pada SBERT diterapkan pada ruang *embedding* semantik berdimensi 384 seperti pada persamaan 12:

$$\text{cosine\_similarity}\,(E_{Q,}E_D) = \frac{E_Q.E_D}{||E_{Q,}||\,\text{x}\,||E_D} \qquad (12)$$

Pada persamaan 12 tersebut, di mana E_Q merupakan vektor *embedding* kueri berdimensi 384 dan E_D merupakan vektor *embedding* dokumen referensi dengan dimensi yang sama.

Mengacu pada pendekatan *hybrid retrieval* yang dikemukakan oleh Hsu & Tzeng (2025), sistem *hybrid* umumnya menggabungkan skor similarity menggunakan parameter bobot tetap α sebagai berikut seperti pada persamaan 13:

$$\text{R}\,(\text{q},\text{d}) = \alpha \cdot \tilde{\text{S}}_{\,\text{dense}}\,(\text{q},\text{d}) + (1-\alpha) \cdot \tilde{\text{S}}_{\,\text{BM25}}\,(\text{q},\text{d}) \qquad (13)$$

Pada penelitian ini, persamaan tersebut diadaptasi dengan menggantikan metode dense menggunakan SBERT *cosine similarity* dan metode sparse menggunakan TF-IDF *cosine similarity*, sehingga rumus menjadi seperti pada persamaan 14:

$$\textit{Hybrid Score} = \alpha \times \text{TF-IDF}_{\text{cosine}} + (1-\alpha) \times \text{SBERT}_{\text{cosine}} \qquad (14)$$

Pada persamaan 14 tersebut, diketahui bahwa parameter α (alpha) menunjukkan bobot kontribusi metode TF-IDF, sedangkan (1 − α) merupakan bobot untuk metode SBERT. Dalam penelitian ini digunakan nilai α = 0,3 sehingga metode TF-IDF berkontribusi sebesar 30% dan metode SBERT berkontribusi sebesar 70%.

Pemilihan bobot ini didasarkan pada pertimbangan bahwa SBERT mampu menangkap kemiripan berbasis makna secara lebih mendalam dibandingkan TF-IDF yang hanya mengandalkan kecocokan berbasis kata, sehingga kontribusi yang lebih besar diberikan kepada SBERT untuk menghasilkan skor kemiripan yang lebih representatif terhadap kesamaan makna antar kalimat. Hasil akhir berupa skor *hybrid* dalam rentang 0 hingga 1 yang kemudian dikonversi ke dalam bentuk persentase agar lebih mudah diinterpretasikan.

Tahap berikutnya adalah melakukan pengurutan skor secara menurun (*descending*) untuk mengidentifikasi dokumen dengan tingkat kemiripan tertinggi. Sistem kemudian menampilkan sejumlah dokumen teratas, dalam implementasi ini sebanyak Top-10, disertai metadata pendukung seperti nama file dokumen, skor hybrid, serta rincian skor TF-IDF dan SBERT. Penyajian rincian skor masing-masing metode bertujuan untuk menjaga transparansi proses perhitungan, sehingga pengguna maupun peneliti dapat memahami secara jelas kontribusi setiap metode terhadap nilai akhir skor *hybrid*.

f. Evaluasi Kinerja Model

Tahap ini bertujuan untuk mengevaluasi kinerja model *Sentence*-BERT (SBERT) dan metode TF-IDF secara sistematis guna memastikan tingkat efektivitas serta akurasi dalam mendeteksi kesamaan konteks antar dokumen. Penelitian ini menggunakan tiga metrik utama untuk mengevaluasi kinerja sistem sebagaimana telah dijelaskan pada Bab 2, yaitu:

1). *precision*

precision merupakan metrik yang digunakan untuk menilai seberapa banyak dokumen relevan yang berhasil ditampilkan dalam 10 hasil

teratas suatu sistem temu kembali. Nilai ini diperoleh dengan membandingkan jumlah dokumen relevan yang muncul pada posisi 1 hingga 10 dengan total 10 dokumen yang disajikan. Dengan demikian, *precision*@10 menggambarkan tingkat akurasi sistem dalam menyediakan informasi yang benar-benar relevan bagi pengguna pada bagian awal hasil pencarian.

2). *recall*

recall digunakan untuk menilai sejauh mana sistem mampu menemukan dokumen-dokumen relevan dalam 10 hasil teratas dibandingkan dengan seluruh dokumen relevan yang telah ditetapkan berdasarkan ground truth. Nilai *recall* ini diperoleh dengan membagi jumlah dokumen relevan yang muncul pada posisi 1 hingga 10 dengan total dokumen relevan. Dengan demikian, *recall*@10 menggambarkan tingkat kelengkapan sistem dalam mengidentifikasi seluruh dokumen yang relevan.

3). *Mean Average precision* (MAP)

Mean Average precision (MAP) merupakan metrik evaluasi yang digunakan untuk menilai kualitas peringkat hasil pencarian dalam sistem temu kembali informasi. MAP digunakan untuk mengukur sejauh mana sistem mampu menempatkan dokumen relevan pada posisi teratas hasil pencarian. Perhitungan MAP dilakukan dengan menghitung nilai *Average precision* (AP) pada setiap kueri, kemudian dirata-ratakan. Nilai *Average precision* (AP) diperoleh dengan menghitung rata-rata nilai *precision* setiap kali dokumen relevan ditemukan pada posisi tertentu. *precision* pada posisi ke-k menunjukkan proporsi dokumen relevan di antara k dokumen teratas yang dihasilkan oleh sistem. Secara matematis, MAP dihitung dengan merata-ratakan nilai AP dari seluruh kueri yang diuji, sehingga dapat

digunakan sebagai indikator kinerja sistem dalam menghasilkan urutan dokumen yang relevan secara keseluruhan.

3.4.3.2 Pembuatan Antarmuka (*Frontend*)

Perancangan antarmuka pengguna pada sistem SimTA dilakukan dengan mempertimbangkan kemudahan interaksi dan alur kerja masing-masing jenis pengguna. Sistem dirancang memiliki dua kelompok halaman utama, yaitu halaman untuk pengguna umum dan panel administrator. Halaman pengguna umum mencakup beranda, unggah dokumen, dan hasil analisis. Pada halaman unggah dokumen, dirancang komponen yang memungkinkan pengguna memilih program studi, memasukkan dokumen melalui unggah berkas atau input teks langsung, serta menampilkan pratinjau informasi berkas dan indikator progres pemrosesan dokumen secara bertahap. Halaman hasil analisis dirancang sebagai *dashboard* yang menampilkan ringkasan tingkat kesamaan dokumen, visualisasi perbandingan skor dokumen, serta daftar dokumen referensi terdekat dengan opsi untuk melihat perbandingan teks secara berdampingan.

Panel administrator dirancang sebagai antarmuka mandiri yang memfasilitasi proses autentikasi dan manajemen dokumen referensi. Panel ini mencakup halaman login, *dashboard* statistik, serta halaman untuk melakukan daftar, penambahan, dan pembaruan dokumen referensi. Setiap halaman dan komponen dirancang dengan identitas visual yang konsisten dan mudah dipahami, sehingga memudahkan administrator dalam mengelola data serta memonitor proses analisis dokumen.

3.4.4 Pengujian (*Testing*)

Setelah sistem berhasil dikembangkan dan diintegrasikan, dilakukan rangkaian pengujian untuk memastikan bahwa setiap komponen bekerja sesuai dengan spesifikasi yang ditetapkan. Penelitian ini menerapkan metode *White Box Testing*, yang berfokus pada evaluasi struktur internal kode, alur logika, serta jalur eksekusi program. Proses pengujian dilaksanakan pada tiga level, yaitu *Unit Testing*, *Integration Testing*, dan *System Testing*.

3.4.4.1 *Unit Testing*

Unit testing direncanakan untuk memverifikasi setiap fungsi dalam sistem secara mandiri, guna memastikan bahwa seluruh komponen bekerja sesuai dengan tujuan perancangannya. Pengujian ini mencakup beberapa fungsi inti, dimulai dari fungsi *preprocessing* yang meliputi *case folding*, *tokenizing*, *stopword removal*, dan *stemming*, untuk memastikan teks diproses secara tepat; misalnya dengan menguji apakah kata "berjalan" dapat diubah menjadi "jalan". Selanjutnya, fungsi ekstraksi teks dari dokumen PDF dan Word akan diuji untuk memastikan informasi dapat diambil tanpa kehilangan konten penting. Proses pembentukan vektor juga akan diperiksa, baik pada TF-IDF maupun SBERT, untuk memastikan dimensi vektor sesuai dengan yang diharapkan, seperti vektor berdimensi 384 pada SBERT. Selain itu, fungsi perhitungan *similarity* seperti *cosine_similarity()* akan diuji untuk memastikan nilai kemiripan dihitung dengan benar dalam rentang 0 hingga 1. Setiap fungsi direncanakan diuji dengan berbagai variasi input untuk memverifikasi konsistensi dan kesesuaian *output* dengan ekspektasi.

3.4.4.2 *Integration Testing*

Integration testing direncanakan untuk menguji interaksi antar modul yang telah berhasil melewati *unit testing*, dengan tujuan memastikan setiap modul bekerja secara terpadu dalam satu sistem yang utuh. Pengujian ini mencakup beberapa skenario utama, antara lain integrasi modul unggah dokumen dengan tahapan proses *preprocessing* untuk memastikan dokumen yang diunggah dapat diproses tanpa kesalahan format, integrasi *preprocessing* dengan proses pembentukan vektor agar teks hasil *preprocessing* dapat dikonversi menjadi vektor TF-IDF maupun *embedding* SBERT secara tepat, serta integrasi vektor dengan mesin perhitungan *similarity* untuk memastikan nilai kemiripan yang dihasilkan dapat digunakan dalam pemeringkatan dokumen secara akurat. Selain itu, direncanakan pengujian integrasi *Backend* Flask dengan basis data *MySQL* untuk memastikan hasil perhitungan *similarity* dapat disimpan dan diambil kembali tanpa kehilangan data, serta integrasi *Backend* dengan antarmuka pengguna untuk menjamin data hasil perhitungan dapat ditampilkan dengan benar melalui antarmuka sistem.

3.4.4.3 *System Testing*

System testing direncanakan untuk mengevaluasi sistem secara menyeluruh dalam kondisi yang mendekati penggunaan nyata. Pengujian ini dibagi menjadi dua kategori, yaitu fungsional dan non-fungsional. Pengujian fungsional mencakup kemampuan sistem dalam menangani unggah dokumen dengan berbagai format dan ukuran file, proses ekstraksi serta *preprocessing* pada berbagai jenis konten dokumen, perhitungan *similarity* dan pemeringkatan dokumen, serta penyajian hasil analisis pada antarmuka pengguna yang menampilkan informasi seperti judul dokumen, nama penulis,

tahun, dan skor kemiripan dari tiga metode (TF-IDF, SBERT, dan *Hybrid*). Sementara itu, pengujian non-fungsional meliputi evaluasi performa sistem dengan mengukur waktu pemrosesan dokumen sejak proses unggah hingga hasil ditampilkan, validasi keamanan melalui pemeriksaan format dan ukuran file untuk mencegah unggahan yang tidak sesuai ketentuan.

V. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan dalam pengembangan Sistem Pendeteksi Kesamaan Konteks pada Dokumen Akademik Mahasiswa Jurusan Ilmu Komputer Universitas Lampung (SimTA), maka dapat dirumuskan beberapa kesimpulan sebagai berikut:

1. Sistem SimTA berhasil dikembangkan sebagai aplikasi web berbasis Flask yang menggabungkan metode TF-IDF dan Sentence-BERT untuk mendeteksi kesamaan konteks dokumen akademik. Sistem ini mampu memproses dokumen dalam format PDF, DOCX, dan TXT, serta menyajikan hasil analisis secara langsung melalui antarmuka yang mudah digunakan.
2. Berdasarkan hasil pengujian menggunakan 10 kueri pada dataset S1 Ilmu Komputer, metode *hybrid* (TF-IDF + SBERT) memberikan kinerja terbaik dibandingkan metode tunggal. Model *hybrid* mencapai nilai *precision* sebesar 90%, *recall* sebesar 93,8%, dan MAP sebesar 91,5%, lebih tinggi dibandingkan TF-IDF (*precision* 85%, *recall* 88,5%, MAP 85,5%) dan SBERT (*precision* 82%, *recall* 85,4%, MAP 80,9%).

5.2 Saran

Berdasarkan keterbatasan yang ditemukan selama penelitian dan untuk mendukung pengembangan sistem yang lebih baik, peneliti mengajukan beberapa saran sebagai berikut:

1. Model SBERT yang digunakan saat ini masih bersifat umum dan belum dilatih khusus menggunakan data akademik berbahasa Indonesia. Oleh

karena itu, pada penelitian selanjutnya disarankan untuk melakukan *fine-tuning* pada model SBERT agar dapat lebih memahami gaya penulisan dan konteks akademik dalam bahasa Indonesia, sehingga hasil perhitungan kesamaan menjadi lebih akurat.

2. Sistem ini juga dapat dikembangkan dengan menambahkan fitur rekomendasi otomatis berdasarkan hasil deteksi kesamaan. Fitur ini dapat membantu memberikan saran topik penelitian yang masih jarang diteliti, berdasarkan hubungan kemiripan antar dokumen, sehingga dapat mendorong mahasiswa menghasilkan penelitian yang lebih beragam dan inovatif.
3. Pengembangan selanjutnya dapat dilakukan dengan menambahkan fitur pemrosesan dokumen secara bertahap (*batch*) atau menggunakan sistem antrian (*queue*). Fitur ini dapat membantu sistem menangani proses unggah dan analisis banyak dokumen secara bersamaan dengan lebih cepat.
4. Penelitian selanjutnya disarankan untuk melakukan *performance testing* atau pengujian performa sistem untuk mengetahui seberapa baik sistem bekerja saat digunakan secara nyata. Pengujian ini dilakukan untuk mengukur kecepatan respons sistem ketika diakses oleh banyak pengguna secara bersamaan, serta melihat kemampuan sistem dalam menangani beban kerja yang tinggi. Pengujian tersebut penting karena proses analisis dokumen pada sistem SimTA, terutama pada tahap *encoding* SBERT dan perhitungan kemiripan dokumen, membutuhkan waktu dan kemampuan pemrosesan sistem yang cukup besar, terutama jika jumlah dokumen dalam basis data terus bertambah. Dengan adanya pengujian performa, pengembang dapat mengetahui bagian sistem yang masih perlu dioptimalkan agar sistem tetap dapat berjalan dengan cepat, stabil, dan lancar meskipun digunakan dalam skala yang lebih besar.

DAFTAR PUSTAKA

Akbar, I. S., & Haryanti, T. (2021). Pengembangan Entity Relationship Diagram *database* Toko Online Ira Surabaya. *Jurnal Ilmiah Computing Insight*, *3*(2), 28–35. https://doi.org/10.30651/comp_insight.v3i2.12002

Alfan, M., Senja, A., Ratna, I., Iqrok, N., & Ahmad, H. (2020). Improving Text Preprocessing For Student Complaint Document Classification Using Sastrawi. *IOP Conference Series: Materials Science and Engineering*, 1–6. https://doi.org/10.1088/1757-899X/874/1/012017

Aljabar, A., & Karomah, B. M. (2024). Mengungkap Opini Publik: Pendekatan BERT-based-caused untuk Analisis Sentimen pada Komentar Film. *Journal of System and Computer Engineering (JSCE)*, *5*(1), 36–43. https://doi.org/10.61628/jsce.v5i1.1060

Alzubi, J., Nayyar, A., & Kumar, A. (2018). Machine Learning from Theory to Algorithms An Overview. *Journal Of Physics: Conference Series*, *1142*(1). https://doi.org/10.1088/1742-6596/1142/1/012012

Amalia, E. L., Jumadi, A. J., Mashudi, I. A., & Wibowo, D. W. (2021). Analisis Metode Cosine Similarity Pada Aplikasi Ujian Online Otomatis (Studi Kasus JTI POLINEMA). *Jurnal Teknologi Informasi Dan Ilmu Komputer*, *8*(2), 343–348. https://doi.org/10.25126/jtiik.2021824356

Amien, M. (2023). Sejarah dan Perkembangan Teknik Natural Language Processing (NLP) Bahasa Indonesia: Tinjauan tentang sejarah, perkembangan teknologi, dan aplikasi NLP dalam bahasa Indonesia. *Elang: Journal of Interdisciplinary Research*, 99–105. https://doi.org/10.48550/arXiv.2304.02746

Amin, M., Nura, N., Rizky, A., Patria, Y., & 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), 6805–6811. https://doi.org/10.36040/jati.v9i4.13969

Andarsyah, R., & Yanuar Amri. (2024). *Sentimen Analisis Aplikasi PosAja pada Google Play Store untuk Peningkatan Pospay Superapp Menggunakan Support Vector Machine*. *16*(2), 1–7. https://ejurnal.ulbi.ac.id/index.php/informatika/article/view/3533/1276

Arsyad, A., Hamid, M., & Santosa, M. (2024). Penerapan Teks Mining dan Cosine Similarity untuk Menentukan Kesamaan Dokumen Skripsi. *IJIS Indonesian Journal on Information System*, *9*(1), 99–109. https://doi.org/10.36549/ijis.v9i1.314

Christioko, B. V., & Daru, A. F. (2018). Sistem Temu Kembali Informasi untuk Pencarian Judul Tugas Akhir Berbasis Kata Kunci. *Pengembangan Rekayasa Dan Teknologi*, *2*, 41–49. http://journals.usm.ac.id/index.php/jprt/index

Deolika, A., & Taufiq Luthfi, E. (2019). Analisis Pembobotan Kata pada Klasifikasi Text Mining. *Jurnal Teknologi Informasi*, *3*(2). https://files.core.ac.uk/download/pdf/287325339.pdf

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. http://arxiv.org/abs/1810.04805

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

Fadhullah, A. N., Fauziah, F., & Winarsih, W. (2022). Aplikasi Deteksi Dini Plagiarism Penelitian Ilmiah Menggunakan Algoritma Consine Similarity Berbasis Web. *Jurnal JTIK (Jurnal Teknologi Informasi Dan Komunikasi)*, *6*(3), 325–334. https://doi.org/10.35870/jtik.v6i3.427

Farozi, M. I., Kurnia, D., & Aprilia, R. A. L. (2023). Perencanaan Green Computing Melalui Digitalisasi Dokumen Akademik pada ITS NUS Sumatera Selatan. *MDP Student Conference*, *2*(1), 596–603. https://doi.org/10.35957/mdp-sc.v2i1.4477

Fitri, D., & Setiyawati, N. (2021). Implementasi Flask Framework pada Pembangunan Aplikasi Purchasing Approval Request. *Jurnal Janitra Informatika Dan Sistem Informasi*, *1*(1), 19–34. https://doi.org/10.25008/janitra.v1i1.120

Hambarde, K. A., & Proença, H. (2023). Information Retrieval: Recent Advances and Beyond. *IEEE Access*, *11*, 76581–76604. https://doi.org/10.1109/ACCESS.2023.3295776

Haviluddin, H., Patandianan, S. J., Putra, G. M., Puspitasari, N., & Pakpahan, H. S. (2021). Implementasi Metode K-Means Untuk Pengelompokkan Rekomendasi Tugas Akhir. *Informatika Mulawarman : Jurnal Ilmiah Ilmu Komputer*, *16*(1), 13–18. https://doi.org/10.30872/jim.v16i1.5182

Hayati, H., & Alifi, M. R. (2025). Information Retrieval Berbasis Latent Dirichlet Allocation pada Data Kekayaan Intelektual. *Jurnal Teknologi Terapan*, 201–209. https://doi.org/10.31884/jtt.v11i2.793

Hsu, H.-L., & Tzeng, J. (2025). *DAT: Dynamic Alpha Tuning for Hybrid Retrieval in Retrieval-Augmented Generation*. http://arxiv.org/abs/2503.23013

Ijudin, A., & Saifudin, A. (2020). Pengujian Black Box pada Aplikasi Berita Online dengan Menggunakan Metode Boundary Value Analysis. *Jurnal Informatika Universitas Pamulang*, *5*(1), 8–12. https://doi.org/10.32493/informatika.v5i1.3717

Illahi, K. N., Suhartini, & Fajriyah. (2023). Implementasi Metode Extreme Programming pada Sistem Informasi Repositori Skripsi di Perpustakaan Universitas Prabumulih. *TEKNIMEDIA: Teknologi Informasi Dan Multimedia*, *4*(2), 182–189. https://doi.org/10.46764/teknimedia.v4i2.128

Islam, S., Elmekki, H., Elsebal, A., Bentahar, J., Drawel, N., Rjoub, G., & Pedrycz, W. (2023). A Comprehensive Survey on Applications of Transformers for Deep Learning Tasks. *Expert Systems with Applications (Elsevier)*. https://doi.org/10.48550/arXiv.2306.07303

Jatmika, S., Indriastuti, M. T., Wafdulloh, G. A., & Malang, S. A. (2018). Implementasi Text Mining untuk Klasifikasi Buku Berdasarkan Dewey Decimal Classification (DDC) di Perpustakaan STMIK Asia Malang Berbasis Vector Space Model. *Jurnal Sistem Dan Teknologi Informasi*, *4*(2), 103–112.

Kemdiktisaintek. (2024). *Buku Statistik Pendidikan Tinggi 2024*. Pusat Data dan Teknologi Informasi. https://drive.google.com/file/d/1RcIDvDRs1VNcNAvPCOHeBxBzY3NgXizN/view

Kurniadi, D., Haviana, S. F. C., & Novianto, A. (2020). Implementasi Algoritma Cosine Similarity pada sistem arsip dokumen di Universitas Islam Sultan Agung. *Jurnal Transformatika*, *17*(2), 124–133. https://doi.org/10.26623/transformatika.v17i2.1613

Latipah, S. A. (2022). Pemodelan Sistem Informasi Form Pembelian Urgent pada PT. Kalbe Morinaga Indonesia Menggunakan Unified Modeling Language. *JIPI (Jurnal Ilmiah Penelitian Dan Pembelajaran Informatika)*, *7*(4), 1332–1341. https://doi.org/10.29100/jipi.v7i4.3199

Lumbansiantar, S., Dwiasnati, S., & Fatonah, N. S. (2023). Penerapan Metode Cosine Similarity dalam Mendeteksi Plagiarisme pada Jurnal. *Jurnal Format*, *12*(2), 142–150. https://publikasi.mercubuana.ac.id/index.php/format/article/view/18437

Mardiansyah, A., Kasah, B. N., Zamzami, H. R., Arabu, Y., Nasro, M. A., Kristanto, N., Paojiah, R., & Wulandari, Y. (2025). Pengenalan Dasar HTML dan CSS: Langkah Pertama dalam Pengembangan Web. *Abdi Jurnal Publikasi*, *3*(3), 165–170. https://jurnal.portalpublikasi.id/index.php/AJP/article/view/1597/1187

Martin, M., & Nilawati, L. (2019). *recall* dan *precision* Pada Sistem Temu Kembali Informasi Online Public Access Catalogue (OPAC) di Perpustakaan. *Paradigma - Jurnal Komputer Dan Informatika*, *21*(1), 77–84. https://doi.org/10.31294/p.v21i1.5064

Nur Azizah, N., Purnamasari, I., & Prangga, S. (2024). Pengelompokan Judul Laporan Skripsi Berbasis Text Mining dengan Metode Fuzzy K-Means. *METIK JURNAL*, *8*(1), 18–23. https://doi.org/10.47002/metik.v8i1.808

Palinggi, O., Maesaroh, S., Permana, M. B., Huda, D. F., & Priyono, K. A. (2024). Entity-Relationship Diagram Technique in *database*. *Collabits Journal*, *1*(2), 102–104. https://publikasi.mercubuana.ac.id/index.php/collabits/article/view/27252/8397

Pamilih, P., Haviluddin, Jati, H., Taruk, M., & Satoso, H. (2018). Sistem Informasi Website Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Mulawarman. *Prosiding Seminar Nasional Ilmu Komputer Dan Teknologi Informasi*, *3*(2), 5–9.

Pandey, D., Niwaria, K., & Chourasia, B. (2019). Machine Learning Algorithms: A Review. *International Research Journal Of Engineering and Technology (IRJET)*, *6*(2), 916–922.

Peng, Q., Weir, D., & Weeds, J. (2021). Structure-aware Sentence Encoder in Bert-Based Siamese Network. *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, 57–63. https://doi.org/10.18653/v1/2021.repl4nlp-1.7

Pertiwi, M., & Taufiqurrochman. (2017). Sistem Temu Kembali Informasi dalam Dokumen (Pencarian 10 Kata Kunci di E-Journal BSI). *Seminar Nasional Sains Dan Teknologi 2017*, 1–6. https://jurnal.umj.ac.id/index.php/semnastek/article/view/2058/1699

Pradana, M. G., Irzavika, N., & Maulana, N. (2024). Deteksi Kemiripan Dokumen Menggunakan Cosine Similarity Berdasarkan Representasi Teks Count Vectorizer dan TF-IDF. *Indonesian Journal of Business Intelligence*, *7*(2), 40–47. https://doi.org/10.21927/ijubi.v7i2.5170

Pricillia, T., & Zulfachmi. (2021). Perbandingan Metode Pengembangan Perangkat Lunak (Waterfall, Prototype, RAD). *Jurnal Bangkit Indonesia*, *10*(1), 6–12. https://doi.org/10.52771/bangkitindonesia.v10i1.153

Rachman, D. A. C., Goejantoro, R., & Amijaya, F. D. T. (2021). Implementasi Text Mining Pengelompokkan Dokumen Skripsi Menggunakan Metode K-Means Clustering. *EKSPONENSIAL*, *11*(2), 167. https://doi.org/10.30872/eksponensial.v11i2.660

Ramdany, S., Kaidar, S., Aguchino, B., Putri, C., & Anggie, R. (2024). Penerapan UML Class Diagram dalam Perancangan Sistem Informasi Perpustakaan Berbasis Web. *Journal of Industrial and Engineering System*, *5*(1), 30–41. https://doi.org/10.31599/2e9afp31

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 3982–3992. https://doi.org/10.48550/arXiv.1908.10084

Rini Rz, M., & Badieah. (2025). DETEKSI PLAGIARISME PADA NOVEL BERBAHASA INGGRIS MENGGUNAKAN AUTHORSHIP ATTRIBUTION BERBASIS STYLOMETRY DAN SUPPORT VECTOR MACHINE (SVM). *Jurnal Ilmiah Sultan Agung*, 135–148. https://doi.org/jurnal.unissula.ac.id/index.php/JIMU/article/view/49808/14475

Riyani, A., Zidny Muhammad, & Burhanuddin, A. (2019). Penerapan Cosine Similarity dan Pembobotan TF-IDF untuk Mendeteksi Kemiripan Dokumen. *Jurnal Linguistik Komputasional*, *2*(1), 23–27. http://download.garuda.kemdikbud.go.id/article.php?article=975400&val=15014&title=Penerapan%20Cosine%20Similarity%20dan%20Pembobotan%20TF-IDF%20untuk%20Mendeteksi%20Kemiripan%20Dokumen/1000

Ronal, Yunita, & Yuliana. (2022). Desain Unified Modeling Language (UML) Dalam Perancangan Aplikasi Hauling Trip Di Industri Tambang Batubara. *Jurnal Teknik Informatika Dan Sistem Informasi*, *9*(4), 3038–3050. https://jurnal.mdp.ac.id/index.php/jatisi/article/view/2401/1023

Sambi Ua, A., Lestriani, D., Sonia, E., Ong, J., Savinka, M., Nurhaliza, P., & Yulia, R. (2023). Penggunaan Bahasa Pemrograman Python Dalam Analisis Faktor Penyebab Kanker Paru-Paru. *Jurnal Publikasi Teknik Informatika (JUPTI)*, *2*(2), 88–99. https://journalcenter.org/index.php/jupti/article/view/1742/1363

Sateesh Kumar Rongali. (2025). Natural Language Processing (NLP) in Artificial Intelligence. *World Journal of Advanced Research and Reviews*, *25*(1), 1931–1935.

Septiani, D., & Isabela, I. (2022). Analisis Term Frequency–Inverse Document Frequency (TF-IDF) dalam Temu Kembali Informasi pada Dokumen Teks. *SINTESIA: Jurnal Sistem Dan Teknologi Informasi Indonesia*, *1*(2), 81–88. https://journal.unj.ac.id/unj/index.php/SINTESIA/article/view/39364

Silalahi, E., Silalahi, D., Plagiarisme Sebagai Peningkatan, D., Irani Tarigan, M., & Veronica Sinaga, R. (2024). Deteksi Plagiarisme sebagai Peningkatan Integritas Akademik. *Kaizen: Jurnal Pengabdian Pada Masyarakat*, (1), 27–33. https://ejournal.ust.ac.id/index.php/KAIZEN/article/view/3867

Siregar, U. K., Sitakar, T. A., Haramain, S., Lubis, Z. N. S., Nadhirah, U., & Yahfizham, Y. (2024). Pengembangan *database* Management system menggunakan My SQL. *Jurnal Sains, Teknologi & Komputer*, *1*(1), 8–12. https://doi.org/10.56495/saintek.v1i1.450

Siswidiyanto, Munif, A., Wijayanti, D., & Haryadi, E. (2020). Sistem Informasi Penyewaan Rumah Kontrakan Berbasis Web Dengan Menggunakan Metode Prototype. *Jurnal Interkom: Jurnal Publikasi Ilmiah Bidang Teknologi Informasi Dan Komunikasi*, *15*(01), 16–23.

Sukowati, I., & Suciptaningsih, O. A. (2024). Literatur Review: Plagiarisme dalam Penulisan Karya Ilmiah: Memahami, Mencegah dan Menangani. *JIIP - Jurnal Ilmiah Ilmu Pendidikan*, *7*(2), 1473–1477. https://doi.org/10.54371/jiip.v7i2.3844

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., & Kaiser, L. (2023). Attention Is All You Need. *Conference on Neural Information Processing Systems*, 1–15. https://arxiv.org/abs/1706.03762v7

Verma, A., Khatana, A., & Chaudhary, S. (2017). A Comparative Study of Black Box Testing and White Box Testing. *International Journal of Computer Sciences and Engineering*, *5*(12), 301–304. https://doi.org/10.26438/ijcse/v5i12.301304

Wahyuni, R. T., Prastiyanto, D., & Supraptono, D. E. (2017). Penerapan Algoritma Cosine Similarity dan Pembobotan TF-IDF pada Sistem Klasifikasi Dokumen Skripsi. *Jurnal Teknik Elektro*, *9*(1), 18–23. https://journal.unnes.ac.id/nju/jte/article/view/10955/6659

Wibowo, I. S., Witanti, A., & Susilawati, I. (2024). Keyword Extraction Judul Berita Online Di Indonesia Menggunakan Metode TF-IDF. *Jurnal Teknik Informatika Dan Sistem Informasi*, *11*(1), 99–111. https://jurnal.mdp.ac.id/index.php/jatisi/article/view/6718

Wijaya, H., Supriyanti, D., & Saefullah, A. (2017). Penggunaan Teknologi Web 2.0 dan Dampak Perubahannya pada Aplikasi Website berbasis Rich Internet Application (RIA). *Ultimatics*, *9*(2), 72–81. https://doi.org/10.31937/ti.v9i2.621

Wilyani, F., Arif, Q., & Aslimar, F. (2024). Pengenalan Dasar Pemrograman Python Dengan Google Colaboratory. *Jurnal Pelayanan Dan Pengabdian Masyarakat Indonesia*, *3*(1), 08–14. https://doi.org/10.55606/jppmi.v3i1.1087

Yunita, R., & Kamayani, M. (2023). Perbandingan Algoritma SVM Dan Naive Bayes Pada Analisis Sentimen Kebijakan Penghapusan Kewajiban Skripsi. *Indonesian Journal of Computer Science*, *12*(5), 2879–2890. http://ijcs.net/ijcs/index.php/ijcs/article/view/3415