ANALISIS KINERJA ALGORITMA LEVENSHTEIN DISTANCE DAN RABIN-KARP DALAM MENDETEKSI PLAGIARISME KODE PYTHON PADA PRAKTIKUM ALGORITMA DAN PEMROGRAMAN

AWANG MULYA NUGRAWAN

H071201027

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

MAKASSAR

2024

ANALISIS KINERJA ALGORITMA LEVENSHTEIN DISTANCE DAN RABIN-KARP DALAM MENDETEKSI PLAGIARISME KODE PYTHON PADA PRAKTIKUM ALGORITMA DAN PEMROGRAMAN

AWANG MULYA NUGRAWAN

H071201027

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

MAKASSAR

2024

ANALISIS KINERJA ALGORITMA LEVENSHTEIN DISTANCE DAN RABIN-KARP DALAM MENDETEKSI PLAGIARISME KODE PYTHON PADA PRAKTIKUM ALGORITMA DAN PEMROGRAMAN

AWANG MULYA NUGRAWAN
H071201027

Skripsi

Sebagai salah satu syarat untuk mencapai gelar sarjana

Program Studi Sistem Informasi

Pada

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS HASANUDDIN
MAKASSAR
2024

SKRIPSI

ANALISIS KINERJA ALGORITMA LEVENSHTEIN DISTANCE DAN RABIN-KARP DALAM MENDETEKSI PLAGIARISME KODE PYTHON PADA PRAKTIKUM ALGORITMA DAN PEMROGRAMAN

Awang Mulya Nugrawan
H071201027

Skripsi,

Telah dipertahankan di depan Panitia Ujian Sarjana Sistem Informasi Pada 17 Juli 2024 dan dinyatakan telah memenuhi syarat kelulusan

Pada

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS HASANUDDIN
MAKASSAR

Mengesahkan:
Pembimbing tugas akhir,

Edy Saputra Rusdi, S.Si., M.Si
NIP. 199104102020053001

Mengetahui:
Ketua Program Studi

Dr. Khaeruddin, M.Sc.
NIP. 196509141991031003

PERNYATAAN KEASLIAN SKRIPSI
DAN PELIMPAHAN HAK CIPTA

Dengan ini saya menyatakan bahwa, skripsi berjudul "Analisis Kinerja Algoritma Levenshtein Distance Dan Rabin-Karp Dalam Mendeteksi Plagiarisme Kode Python Pada Praktikum Algoritma Dan Pemrograman" adalah benaar karya saya dengan arahan dari pembimbing (Edy Saputra Rusdi, S.Si., M.Si) karya ilmiah ini belum diajukan dan tidak sedang diajukan dalam bentuka apapun kepada perguruan tinggi mana pun. Sumber informasi yang berasal atau dikutip dari karya yang diterbitkan maupun tidak diterbitkan dari penulis lain telah disebutkan dalam teks dan dicantumkan dalam Daftar Pustaka skripsi. Apabila di kemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan skripsi ini adalah karya orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut berdasarkan aturan yang berlaku.

Dengan ini saya melimpahkan hak cipta (hak ekonomis) dari karya tulis saya berupa skripsi ini kepada Universitas Hasanuddin.

Makassar, 17 Juli 2024

Awang Mulya Nugrawan

H071201027

UCAPAN TERIMAKASIH

Alhamdulillahi rabbil ‘alamin, segala puji dan syukur senantiasa penulis panjatkan kepada Allah SWT atas segala rahmat dan hidayah-Nya yang berlimpah sehingga penulis dapat menyelesaikan skripsi ini. Salawat serta salam penulis haturkan kepada junjungan Nabi Muhammad SAW, sebagai Nabi yang telah menjadi suri tauladan bagi seluruh umatnya sehingga penulis dapat menyelesaikan skripsi yang berjudul “Analisis Kinerja Algoritma Levenshtein Distance Dan Rabin-Karp Dalam Mendeteksi Plagiarisme Kode Python Pada Praktikum Algoritma Dan Pemrograman”. Sebagai salah satu syarat dalam penyelesaian pendidikan dan mencapai gelar Sarjana Komputer (S.Kom).

Pada kesempatan ini, penulis memberikan penghargaan dan mengucapkan terima kasih yang sebesar-besarnya kepada orang tua penulis, Ayahanda Muhammad Agung, dan Ibunda Hasmili Ganisa yang telah sabar membesarkan dan mendidik penulis, serta memberikan do’a dan kasih sayang tak terhingga.

Penulis sampaikan terima kasih dengan sepenuh hati kepada Bapak Edy Saputra Rusdi, S.Si., M.Si. selaku Dosen Pembimbing Utama yang dengan sabar, tulus, dan Ikhlas banyak memberikan ilmu dan meluangkan waktu untuk membimbing dan memberikan masukan serta motivasi dalam penulisan skripsi ini. Ucapan terima kasih juga penulis persembahkan kepada tim penguji Bapak Dr. Khaeruddin, M.Sc. dan Bapak Dr. Hendra, S.Si., M.Kom. yang telah meluangkan waktu untuk memberikan masukan dan kritikan yang membantu penyempurnaan penulisan skripsi ini.

Ucapan terima kasih juga kepada pimpinan Universitas Hasanuddin dan pimpinan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin beserta Bapak dan Ibu Dosen Departemen Matematika terutama pada Bapak dan Ibu Dosen Program Studi Sistem Informasi yang telah memberikan banyak ilmu dan pengetahuan kepada penulis selama menjadi mahasiswa. Terimakasih juga kepada teman-teman seperjuangan penulis di bangku perkuliahan (Talitha, Faizah, Ufairah, Yan, Mufti, Hamsa, Rizuki, Mita, Rani, dan teman- teman asisten lab lainnya) senantiasa memberikan bantuan dan dukungan moral kepada penulis, serta memberikan momen dan pengalaman berharga bagi penulis selama masa perkuliahan.

Terakhir kepada Seluruh pihak yang tidak dapat penulis sebut satu persatu, terima kasih untuk segala dukungan, doa, motivasi, inspirasi, dan partisipasi yang diberikankepada penulis. Akhir kata, penulis berharap semoga segala bentuk kebaikan yang telah diberikan bernilai ibadah dan mendapatkan balasan dari Allah Subhanahu Wa Ta’ala. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu pengetahuan.

Penulis,

Awang Mulya Nugrawan

ABSTRAK

AWANG MULYA NUGRAWAN. Analisis Kinerja Algoritma Levenshtein Distance Dan Rabin-Karp Dalam Mendeteksi Plagiarisme Kode Python Pada Praktikum Algoritma Dan Pemrograman (dibimbing oleh Edy Saputra Rusdi, S.Si., M.Si)

Kemajuan teknologi yang pesat, terutama di bidang teknologi dan internet, membawa dampak positif yang signifikan. Namun, terdapat juga konsekuensi negatif, seperti plagiarisme. Plagiarisme merupakan tindakan tidak etis dalam bidang akademik, melibatkan penggunaan karya orang lain tanpa izin dan mengakuinya sebagai karya sendiri. Penelitian ini berfokus pada plagiarisme kode, bentuk pelanggaran akademik yang umum terjadi di mana mahasiswa mengirimkan kode orang lain sebagai miliknya sendiri. Penelitian ini bertujuan untuk menganalisis kinerja algoritma Levenshtein Distance dan Rabin-Karp dalam mendeteksi plagiarisme kode Python selama praktikum Algoritma dan Pemrograman. Selain itu, penelitian ini mengeksplorasi potensi teknik *ensamble* yang menggabungkan kedua algoritma untuk meningkatkan akurasi deteksi dan mengembangkan aplikasi *web* interaktif menggunakan Streamlit untuk deteksi plagiarisme. Penelitian ini melibatkan pengumpulan data, pemrosesan, implementasi algoritma, pengembangan web, dan evaluasi kinerja dalam berbagai skenario plagiarisme. Hasil penelitian menunjukkan bahwa Levenshtein Distance efektif mendeteksi perubahan kecil dalam kode tetapi memiliki waktu eksekusi yang lebih lama, sedangkan Rabin-Karp lebih cepat tetapi kurang sensitif terhadap kesamaan logika kecil. Teknik *Voting Classifier ensamble* menunjukkan kinerja seimbang antara waktu dan akurasi. Aplikasi *web* yang dikembangkan memudahkan deteksi plagiarisme dan membuatnya lebih mudah diakses oleh dosen dan mahasiswa.

Kata Kunci: Deteksi Plagiarisme Kode, Levenshtein Distance, Praktikum Algoritma dan Pemrograman, Rabin-Karp, Teknik *Ensemble*

ABSTRACT

AWANG MULYA NUGRAWAN. *Performance Analysis Of Levenshtein Distance And Rabin-Karp Algorithms In Detecting Python Code Plagiarism In Algorithms And Programming Practicum* (supervised by Edy Saputra Rusdi, S.Si., M.Si)

The rapid advancement of technology, particularly in the field of technology and the internet, has brought about significant positive impacts. However, it also has negative consequences, such as plagiarism. Plagiarism is an unethical act in the academic field, involves using someone else's work without permission and claiming it as one's own. This study focuses on code plagiarism, a prevalent form of academic misconduct where students submit someone else's code as their own. This research aims to analyze the performance of Levenshtein Distance and Rabin-Karp algorithms in detecting Python code plagiarism during Algorithm and Programming courses. Additionally, it explores the potential of ensemble techniques combining both algorithms to enhance detection accuracy and develops an interactive web application using Streamlit for plagiarism detection. The research involves data collection, preprocessing, algorithm implementation, web development, and performance evaluation under various plagiarism scenarios. Results indicate that Levenshtein Distance effectively detects minor code changes but has a longer execution time, while Rabin-Karp is faster but less sensitive to small logical similarities. The Voting Classifier ensemble shows a balanced performance between time and accuracy. The developed web application facilitates easy and accessible plagiarism detection for educators and students.

Key words: Algorithm and Programming Lab, *Ensemble Technique*, Levenshtein Distance, *Code Plagiarism Detection*, Rabin-Karp

DAFTAR ISI

Halaman

DAFTAR GAMBAR

No.urut Halaman

DAFTAR TABEL

DAFTAR LAMPIRAN

Nomor urut Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Seiring dengan perkembangan teknologi yang sangat pesat saat ini, khususnya di bidang teknologi dan internet, banyak dampak positif yang dirasakan dari kemajuan teknologi ini. Namun tidak sedikit pula dampak negatif yang hampir tidak bisa dihindari, salah satunya adalah tindakan plagiarisme. Plagiarisme merupakan tindakan yang tidak etis dalam bidang akademik, yaitu menggunakan karya orang lain tanpa izin dan mengakuinya sebagai karya sendiri. Plagiarisme dapat terjadi dalam berbagai bentuk, termasuk menyalin teks, kode, atau gambar dari sumber lain tanpa memberikan kredit. Plagiarisme kode merupakan salah satu bentuk plagiarisme yang sering terjadi, yaitu menggunakan kode yang ditulis oleh orang lain tanpa izin dan mengakuinya sebagai karya sendiri. Plagiarisme kode dapat terjadi dalam berbagai bentuk, termasuk menyalin kode dari sumber lain, menggunakan kode yang dibagikan secara online, atau menggunakan kode yang ditulis oleh orang lain untuk tugas yang sama.

Di lingkungan akademis, plagiarisme yang biasanya sering terjadi pada dokumen tekstual seperti esai, laporan, dan bahkan penelitian. Akan tetapi, plagiarisme tidak hanya berlaku pada dokumen tekstual tetapi juga dokumen *source code*. Plagiarisme *source code* di dunia akademis biasanya terjadi ketika mahasiswa meniru kode milik mahasiswa lain dan melakukan *submission* seolah-olah seperti pekerjaan milik mahasiswa tersebut. Permasalahan yang biasanya timbul di lingkungan akademis yang berkaitan dengan sistem informasi adalah terjadinya plagiarisme di tugas-tugas pemrograman (Widjaja dkk, 2022).

Praktik plagiat tidaklah menjadi hal asing lagi, apalagi di kalangan mahasiswa yang hampir setiap hari mengerjakan tugas yang diberikan oleh dosen. Tak terkecuali pula, mahasiswa sering mendapat tugas untuk membuat program aplikasi dengan menggunakan bahasa pemrograman tertentu Dalam pengerjaan tugas tersebut, praktik plagiat tak terelakkan lagi untuk dilakukan mengingat waktu pengerjaan tugas yang terbatas dan tidak adanya motivasi untuk berusaha menyelesaikan tugas dengan kemampuan sendiri. Praktik plagiat dilakukan dengan cara tukar menukar kode program *(source code)* yang telah berhasil. Mahasiswa yang melakukan plagiat dapat dengan mudah menyalin atau mengganti kode program yang telah didapatkan secara cepat dengan menggunakan fitur-fitur yang disediakan oleh komputer (Setia & Lestariningsih, 2018).

Algoritma dan Pemrograman merupakan mata kuliah yang wajib diambil oleh setiap mahasiswa semester 1 Prodi Sistem Informasi Universitas Hasanuddin. Mata kuliah ini merupakan mata kuliah dasar dari banyak mata kuliah lainnya. Untuk bisa menguasai mata kuliah ini, bukan hanya mengerti teori namun harus latihan setiap hari. Dari metode pengajaran yang diterapkan, mahasiswa diberikan tugas praktikum coding untuk berlatih mandiri. Hal ini menuntut banyaknya tugas pemrograman yang

harus diberikan agar mahasiswa semakin mahir. Namun, seringkali ditemukan banyak mahasiswa yang memplagiat hasil pekerjaan mahasiswa lain. Hal ini semakin mempersulit pengajar memberikan penilaian yang objektif. Permasalahan yang sering timbul adalah terjadinya plagiarisme dalam pengerjaan soal-soal tersebut. Untuk mengatasi praktik plagiat, tidaklah cukup hanya mengingatkan kepada mahasiswa bahwa tindakan plagiat tidak baik dilakukan. Pendeteksian praktik plagiat merupakan solusi yang sebaiknya dilakukan sehingga tindakan curang tersebut dapat diminimalisasi.

Terdapat berbagai penelitian yang telah dilakukan untuk mengatasi masalah plagiarisme kode. Salah satu penelitian yang relevan adalah (Setia & Lestariningsih, 2018) yang secara khusus memfokuskan perancangan detektor plagiarisme pada bahasa pemrograman Java dengan menerapkan algoritma Rabin-Karp. Penelitian lain yang relevan adalah analisis perbandingan oleh (Purba & Situmorang, 2021) yang membandingkan algoritma Rabin-Karp dan Levenshtein Distance dalam menghitung kemiripan teks, menunjukkan kelebihan dan kekurangan masing-masing algoritma dalam berbagai kondisi. Selain itu, penelitian oleh (Dhammajoti, 2021) memperkenalkan pendekatan *ensemble learning* menggunakan Logistic Regression, Naive Bayes, dan SVM untuk analisis sentimen bahasa Indonesia, menunjukkan potensi penggunaan teknik ensemble untuk meningkatkan akurasi deteksi plagiarisme. Penelitian ini memberikan kontribusi signifikan terhadap pengembangan metode deteksi kemiripan dokumen dengan mengusulkan kombinasi dua algoritma, yaitu Levenshtein Distance dan Rabin-Karp, yang dapat meningkatkan akurasi deteksi kemiripan dokumen. Meskipun penelitian ini tidak secara langsung berkaitan dengan deteksi plagiarisme kode Python, penggunaan kombinasi algoritma ini dapat menjadi pertimbangan untuk pengembangan metode deteksi plagiarisme kode Python.

Berdasarkan uraian penelitian-penelitian terdahulu ini, penelitian ini bertujuan melakukan analisis kinerja dua algoritma, yaitu Levenshtein Distance dan Rabin-Karp dalam konteks mendeteksi plagiarisme pada praktikum algoritma dan pemrograman, khususnya dengan bahasa pemrograman Python. Selain itu, penelitian ini akan mengeksplorasi potensi teknik *ensamble* atau kombinasi dari kedua algoritma untuk meningkatkan keakuratan deteksi plagiarisme dalam konteks praktikum tersebut. Maka peneliti memutuskan melakukan penelitian berjudul “Analisis Kinerja Algoritma Levenshtein Distance Dan Rabin-Karp Dalam Mendeteksi Plagiarisme Kode Python Pada Praktikum Algoritma Dan Pemrograman”. Di harapkan bahwa hasil penelitian ini akan memberikan pemahaman yang lebih dalam tentang kelebihan dan kekurangan masing-masing algoritma, serta potensi peningkatan performa melalui pendekatan *ensamble* dalam mendeteksi plagiarisme *code* python. Selain itu Pendeteksian kemiripan kode pada mata kuliah Algoritma dan pemrograman ini dilakukan agar memudahkan para dosen dan asisten lab untuk mengoreksi jawaban dari mahasiswa dalam proses penilaian dari tugas maupun ujian.

1.2 Tujuan dan Manfaat

Berdasarkan latar belakang diatas, maka dapat disimpulkan tujuan penelitian sebagai berikut:

1. Menganalisis kinerja algoritma Levenshtein Distance dan Rabin-Karp dalam mendeteksi plagiarisme pada kode Python dalam praktikum algoritma dan pemrograman.
2. Mengeksplorasi potensi penggunaan teknik *ensamble* atau kombinasi dari kedua algoritma untuk meningkatkan keakuratan deteksi plagiarisme pada kode Python.
3. Mengembangkan *web* interaktif untuk mendeteksi plagiarisme kode Python antara satu *file* Python mahasiswa dengan mahasiswa lainnya menggunakan Streamlit.

Berdasarkan tujuan penelitian tersebut, maka manfaat dari penelitian ini yaitu:

1. Memberikan pemahaman yang lebih dalam tentang kelebihan dan kekurangan masing-masing algoritma dalam mendeteksi plagiarisme *code* python.
2. Mengidentifikasi potensi peningkatan performa melalui pendekatan *ensamble* dalam mendeteksi plagiarisme *code* python.
3. Menyediakan solusi praktis untuk mendeteksi plagiarisme pada kode Python di lingkungan pendidikan dengan menggunakan *web* interaktif.

1.3 Landasan Teori

1.3.1 Plagiarisme *Code*

Secara etimologis plagiat berasal dari bahasa Inggris Plagiarism yang apabila dirunut sebenarnya berasal dari bahasa Yunani yaitu Plagiarius berarti penculik atau pencuri karya tulis. Kemudian di kamus Longman Dictionary of English Language and Culture, plagiarism didefinisikan sebagai pengambilan gagasan dari karya orang lain kemudian menggunakan gagasan tersebut dalam karyanya sendiri tanpa memberi penghargaan terhadap penulis aslinya.

Plagiat adalah teknik penyalinan atau meniru karya orang lain yang diklaim menjadi hasil karya sendiri. Tidak adanya motivasi ataupun kemudahan dalam proses penyalinan dengan harapan tidak diketahui orang lain menjadi alasan utama terjadinya praktik plagiat. *Code Plagiarism* adalah istilah yang digunakan untuk mendeskripsikan adanya kemiripan isi *source code* antar 2 *code* yang berbeda (Setia & Lestariningsih, 2018). *Similarity source code* dapat terjadi secara keseluruhan *code* maupun sebagian, serta dengan sedikit perubahan pada bagian yang tidak signifikan. Beberapa jenis plagiat yang dikenal selama ini, yaitu:

a. *Word-for-word plagiarism*: menyalin setiap kata secara langsung tanpa diubah sedikitpun.
b. *Plagiarism of the form of a source*: menyalin dan atau menulis ulang kode-kode program tanpa mengubah struktur dan jalannya program.

c. *Plagiarism of authorship*: mengakui hasil karya orang lain sebagai hasil karya sendiri dengan cara mencantumkan nama sendiri menggantikan nama pengarang sebenarnya.

Beberapa contoh praktik plagiat pada sebuah program, yaitu:

a. Leksikal: perubahan pada kode (*source code*) program, misalnya:
   - Komentar diubah (ditambah, dikurangi, atau diganti).
   - Format penulisan diubah.
   - Nama variabel diubah.
b. Struktural: perubahan struktur program
   - Perubahan urutan algoritma yang tidak mengubah jalannya program.
   - Prosedur diubah menjadi fungsi atau sebaliknya.
   - Pemanggilan prosedur diganti dengan isi prosedur itu sendiri.

1.3.2 Praktikum Algoritma dan Pemrograman

Praktikum algoritma dan pemrograman merupakan salah satu mata kuliah wajib dalam kurikulum pendidikan di Prodi Sistem Informasi Universitas Hasanuddin. Mata kuliah ini diambil pada semester pertama dan diwajibkan untuk semua mahasiswa. Praktikum ini bertujuan untuk memperkenalkan mahasiswa dengan dasar-dasar algoritma dan pemrograman, dengan bahasa pemrograman Python sebagai bahasa utama.

Mata kuliah algoritma dan pemrograman ini terdiri dari 3 SKS (Satuan Kredit Semester), yang terbagi menjadi 2 SKS untuk kuliah teori dengan dosen pengajar dan 1 SKS untuk praktikum yang dilakukan bersama asisten laboratorium. Selama kuliah praktikum, mahasiswa diberikan tugas praktikum setiap minggu nya yang sesuai dengan topik yang diajarkan pada minggu tersebut.

Praktikum algoritma dan pemrograman penting karena dasar-dasar algoritma dan pemrograman merupakan pondasi bagi pemahaman lebih mendalam dalam bidang ilmu komputer. Python, sebagai bahasa pemrograman yang digunakan, memiliki sintaksis yang mudah dipahami, sehingga cocok untuk pemula. Dalam konteks penelitian ini, fokus pada mendeteksi plagiarisme kode Python yang mungkin terjadi dalam tugas praktikum.

1.3.3 Levenshtein Distance

Levenshtein Distance lebih dikenal dengan *edit distance*. Konsep dari Levenshtein distance yaitu mencari jumlah minimum *point mutation* yang diperlukan untuk merubah suatu string ke string yang lain. *Point mutation* tersebut adalah *insertion*, *subtitution* dan *deletion*. Levenshtein distance adalah sebuah matriks string yang berfungsi untuk mengukur perbedaan atau *distance* antara dua string. Nilai *distance* ini ditentukan oleh jumlah dari operasi-operasi perubahan yang digunakan untuk melakukan transformasi dari suatu string menjadi string lainnya (Widjaja dkk, 2022).

Terdapat 3 macam operasi utama yang dapat dilakukan oleh algoritma ini yaitu (Febiawan dkk, 2020):

a. Operasi Pengubahan Karakter Operasi pengubahan karakter merupakan operasi menukar sebuah karakter dengan karakter lain contohnya penulis menuliskan string “yamg” menjadi “yang”. Dalam kasus ini karakter “m” diganti dengan huruf “n”.
b. Operasi Penambahan Karakter Operasi penambahan karakter berarti menambahkan karakter ke dalam suatu string. Contohnya string “kepad” menjadi “kepada”, dilakukan penambahan karakter “a” diakhir string. Penambahan karakter tidak hanya dilakukan diakhir kata, namun bisa ditambahkan diawal maupun disisipkan di tengah string.
c. Operasi Penghapusan Karakter Operasi penghapusan karakter dilakukan untuk menghilangkan karakter dari suatu string. Contohnya string “barur” diubah menjadi “baru”. Pada operasi ini dilakukan penghapusan huruf “r”.

Algoritma Levenshtein Distance dapat digunakan dalam mendeteksi kemiripan antara dua string yang berpotensi melakukan tindak plagiarisme. Algoritma Levenshtein Distance bekerja dari sisi kiri atas sebuah array, pada dua matriks yang telah berisi string A dan string B. Berikut adalah algoritma Levenshtein Distance dalam mendapatkan nilai distance:

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 1: & \text{if } X(i) \neq Y(j) \\ 0: & \text{if } X(i) = Y(j) \end{cases} \end{cases}$$

Levenshtein Distance melakukan perhitungan bobot *similarity* setelah mendapatkan nilai *distance* dari dua dokumen yang dibandingkan. Kemudian menggunakan suatu persamaan dalam menentukan bobot *similarity*, yaitu:

$$\text{Bobot } Similarity = \left(1 - \frac{d[m,n]}{Max(S,T)}\right) \times 100\%$$

Dengan d[m,n] adalah nilai *distance*, terletak pada baris ke m dan kolom ke n, S adalah panjang string awal, T adalah panjang string target, dan Max(S,T) adalah panjang string terbesar antara string awal dan string target.

1.3.4 Rabin-Karp

Algoritma Rabin-Karp adalah salah satu algoritma pencarian string dikembangkan oleh Michael O. Rabin dan Richard M. Karp pada tahun 1987 yang menggunakan fungsi hashing untuk menemukan pattern di dalam string teks. Sebuah algoritma pencarian string sederhana yaitu Brute-Force *Algorithm* yang kemudian berkembang menjadi beberapa algoritma diantaranya adalah Algoritma Knuth Morris Pratt (KMP), Algoritma Boyer-Moore (BM) dan Algoritma Rabin dan Karpp. Menurut (Abdeen, 2011) pada prinsipnya algoritma Rabin-Karp menghitung sebuah fungsi *hash* untuk mencari pola di dalam sebuah teks yang diberikan. Setiap karakter M *subsequence* dari pada teks akan dikomparasi, jika nilai *hash* tidak sama algoritma akan

menghitung nilai *hash* untuk karakter M *subsequence* berikutnya. Dan jika nilai *hash* sama maka algoritma akan melakukan perbandingan secara brute-force antara pola dan karakter M subsequence, dengan cara ini hanya akan ada satu perbandingan per teks subsequence dan brute-force hanya dibutuhkan jika nilai *hash* cocok atau sama. (Jain dkk, 2012). Menurut Arliandinda dalam (Mutiara dkk, 2008) terdapat empat kategori proses perbandingan yaitu:

a) Dari kanan ke kiri.
b) Dari kiri ke kanan.
c) Dalam order spesifik.
d) Dalam order apapun.

Berdasarkan keempat kategori tersebut algoritma Rabin-Karp termasuk dalam kategori dari kiri ke kanan. Algoritma Rabin-Karp menerapkan fungsi *hash* yang menyediakan metode sederhana untuk mencegah kompleksitas waktu $O(m^2)$. Algoritma Rabin-Karp menggunakan fungsi hash yang disebut dengan rolling hash untuk menentukan apakah kata-kata yang dicocokkan sama. Rolling hash adalah sebuah fungsi hash yang input-nya dikelompokkan ke dalam suatu blok yang digerakkan secara keseluruhan melewati input. Beberapa fungsi hash memungkinkan rolling hash untuk di komputasi dengan cepat. Nilai hash yang baru dapat dengan cepat dihitung dari nilai hash yang lama dengan cara menghilangkan nilai lama dari kelompok hash dan menambahkan nilai baru ke dalam kelompok tersebut (Putra et al., 2019).

Algoritma Rabin-Karp memiliki beberapa karakteristik yaitu menggunakan K-Gram dan hashing. Penerapan algoritma Rabin-Karp dilakukan setelah melewati tahapan preprocessing (Siahaan et al., 2018). K-gram adalah sebuah metode yang digunakan untuk menghasilkan kata atau karakter. Dalam metode k-gram ini dilakukan pengambilan potongan-potongan karakter huruf sejumlah k dari sebuah kata yang secara kontinuitas dibaca dari teks sumber hingga akhir dari dokumen Dalam sistem ini, proses hashing menggunakan tabel hash ASCII. Misalkan k-grams yang terbentuk dari c1 sampai ck akan di hash dengan nilai base b. Persamaan hash value dari H(c1…ck) terdapat di persamaan berikut.

$$H(c_1 \dots c_k) = c_1 * b^{k-1} + c2 * b^{k-2} + \cdots + c_{k-1} * b + c_k$$

Hash value dari *sequence* berikutnya akan dihitung dengan menggunakan *rolling hash* seperti di persamaan berikut.

$$H(c_2 \dots c_{k+1}) = (H(c_1 \dots c_k) - \ c_1 * b^{k-1}) * b + c_{k+1}$$

Setelah proses hash selesai, dilakukan perhitungan *similarity* dengan menggunakan Jaccard similarity. Metrik ini melakukan kalkulasi dengan menghitung seberapa banyak data yang sama antara dua buah set yang dibagi dengan jumlah elemen unik di kedua set.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

1.3.5 Ensemble Learning

Ensemble Learning bertujuan untuk mengombinasikan keputusan dari beberapa algoritma pembelajaran dalam upaya meningkatkan akurasi hasil (terutama untuk algoritma pembelajaran yang kurang optimal) dan menghasilkan model yang lebih baik untuk melakukan prediksi. Menggunakan metode *ensemble* terbukti lebih efektif dibandingkan hanya mengandalkan satu algoritma pembelajaran (Onan dkk, 2016). Secara garis besar, *ensemble learning* dapat dibagi menjadi dua kategori, yaitu dependen dan independen. Dalam metode *dependen*, output dari proses pembelajaran digunakan untuk membangun klasifikasi selanjutnya, dengan kata lain, hasil pembelajaran iterasi sebelumnya dapat ditransfer ke iterasi berikutnya. Sementara itu, dalam metode independen, proses pembelajaran dibangun secara terpisah, dan output dari algoritma pembelajaran dikombinasikan melalui metode seperti ma atau metode *meta-learning*.

Majority voting merupakan salah satu bentuk pendekatan *voting based method*, yang hanya berfokus pada label. Dalam penerapan *majority voting*, untuk klasifikasi *instance* yang belum berlabel, dilakukan pemungutan suara untuk menentukan label yang paling banyak dipilih (Roakch, 2019). Salah satu algoritma yang mengadopsi konsep majority voting adalah *Voting Classifier* algorithm (Mahabub dkk, 2019).

1.3.6 Voting Classifier

Voting Classifier merupakan sebuah algoritma yang menggabungkan beberapa model pembelajaran dalam metode *ensemble learning*. Algoritma ini bertindak sebagai *meta-classifier* yang membuat prediksi dengan mengombinasikan hasil prediksi dari beberapa model dasar berdasarkan strategi pemungutan suara yang telah ditetapkan sebelumnya. Dengan *Voting Classifier*, keuntungan dari tiap-tiap base *classifier* dimanfaatkan, dimana setiap *classifier* berusaha memberikan yang terbaik sambil mengurangi dampak kelemahan masing-masing base *classifier* pada bagian lain dari *dataset* (Kumari dkk, 2021). Dalam *Voting Classifier*, setiap base *classifier* dilatih dan disesuaikan pada *dataset* yang sama secara paralel. Dalam membuat prediksi akhir, *Voting Classifier* menggabungkan hasil prediksi *base classifier* menggunakan *soft voting* atau *hard voting* (Misra dkk, 2020). *Soft voting* memprediksi target berdasarkan jumlah prediksi probabilitas kelas. *Hard voting* memprediksi target berdasarkan mayoritas suara (Mahabub dkk, 2019).

1.3.7 Streamlit

Streamlit adalah kerangka kerja aplikasi yang membantu membuat aplikasi *web machine learning* dengan cara yang mudah. Ini sangat cocok dengan perpustakaan utama Python yang digunakan dalam *machine learning* untuk visualisasi dan analisis data. Streamlit menjadi salah satu alat populer untuk mengembangkan dan mendeploy aplikasi web interaktif dengan cepat, terutama dalam konteks *Machine Learning* (Jain dkk, 2022).Penggunaan Streamlit dalam penelitian ini merupakan elemen kunci dalam pengembangan alat untuk mendeteksi plagiarisme kode Python

dalam praktikum algoritma dan pemrograman. Berikut adalah beberapa fungsi dari streamlit:

- Kemudahan penggunaan: Streamlit memiliki API yang sederhana dan mudah dipahami, sehingga developer tidak perlu memiliki pengetahuan yang mendalam tentang teknologi web untuk menggunakannya.
- Kemampuan interaktif: Streamlit memungkinkan developer untuk membuat aplikasi web yang interaktif dengan menggunakan widget-widget seperti slider, text input, checkbox, dan selectbox. Widget-widget ini dapat digunakan untuk mengubah parameter model machine learning atau memfilter data yang ditampilkan.
- Kemampuan visualisasi data: Streamlit menyediakan berbagai macam fungsi untuk memvisualisasikan data, seperti plot, chart, dan tabel.
- Integrasi dengan pustaka Python lainnya: Streamlit dapat diintegrasikan dengan pustaka Python lainnya, seperti NumPy, Pandas, dan Matplotlib.

BAB II
METODE PENELITIAN

2.1 Sumber Data

Data penelitian kali ini menggunakan *dataset* dari 5 *repository* github tugas praktikum bahasa pemrograman python pada praktikum algoritma dan pemrograman 2023. *Repository-repository* tersebut antara lain:

Tabel 1. *Dataset* praktikum algoritma dan pemrograman 2023

No.	Link Repo Github	Jumlah	
		Praktikan	*File* tugas
1	https://github.com/IhlasulMufti/LAB-AP-01-2023	7	164
2	https://github.com/Riofuad/LAB-AP-02-2023	7	203
3	https://github.com/faizahmp/LAB-AP-04-2023	7	174
4	https://github.com/adnanamiruddin/LAB-AP-10-2023	6	164
5	https://github.com/RasyadBima15/LAB-AP-13-2023	6	146
Total		33	851

Pada Tabel 1 menyajikan sampel data *repository* GitHub yang digunakan sebagai sumber penelitian untuk mendeteksi plagiarisme kode python pada praktikum Algoritma dan Pemrograman 2023. Terdapat 5 *repository* yang berisi *file* tugas dari total 33 mahasiswa dan jumlah *file* tugas yang terkumpul dalam setiap *repository* bervariasi, dengan total keseluruhan mencapai 851 *file* tugas python.

2.2 Waktu dan Tempat penelitian

2.2.1 Waktu Penelitian

Tabel 2. Waktu penelitian

No.	Tahapan	April				Mei			
		1	2	3	4	1	2	3	4
1	Pengumpulan Data								
2	Pemrosesan Data								
3	Implementasi Algoritma								

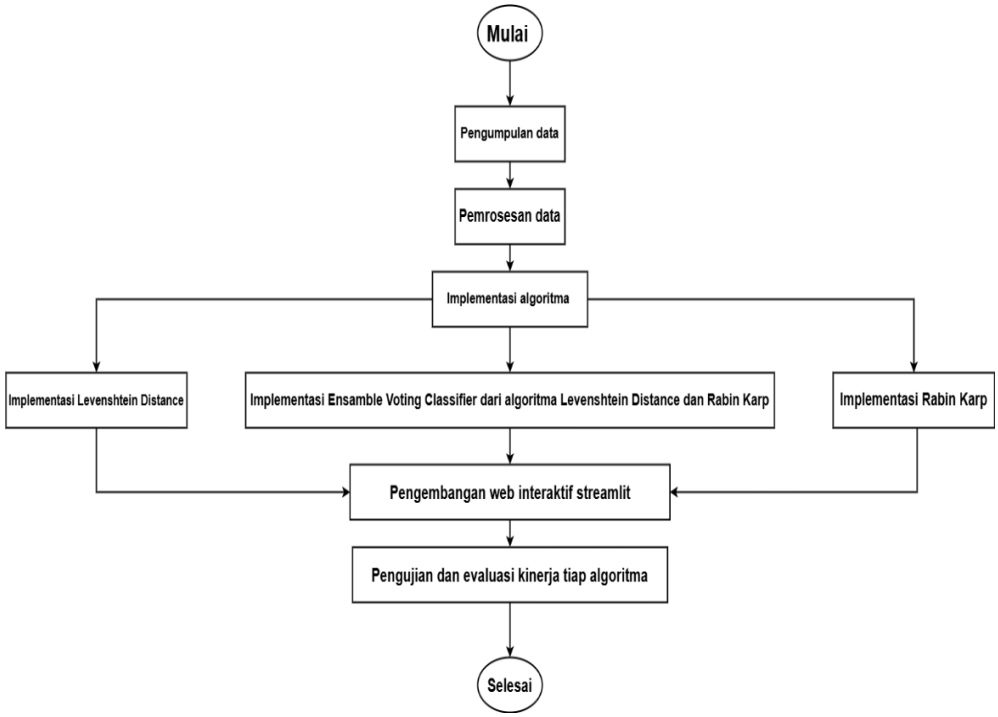
No.	Tahapan	April				Mei			
		1	2	3	4	1	2	3	4
4	Pengembangan *web* interaktif Streamlit								
5	Pengujian dan Evaluasi Algoritma								

2.2.2 **Tempat penelitian**

Tempat penelitian dilakukan di Laboratorium sistem informasi yaitu Laboratorium Rekayasa dan Perangkat Lunak (Lab RPL) Universitas Hasanuddin Kota Makassar, Indonesia.

2.3 Tahapan Penelitian

Pada penelitian ini menggunakan 3 tahapan yang berbeda yaitu menggunakan algoritma Levenshtein distance, Rabin Karp dan Ensemble *Voting Classifier*. Dengan menggunakan tiga pendekatan yang berbeda, penelitian ini berusaha untuk meningkatkan akurasi deteksi plagiarisme pada kode sumber Python. Algoritma Levenshtein Distance dan Rabin-Karp masing-masing memiliki kelebihan dan kekurangan dalam mengidentifikasi kemiripan antara *file*, sehingga dengan menggabungkan keduanya menggunakan *Ensemble Voting Classifier*, diharapkan dapat memberikan hasil yang lebih akurat dan komprehensif dalam mendeteksi plagiarisme.



Gambar 1. Tahapan Penelitian

Gambar 1 merupakan alur tahapan penelitian yang dimulai dari tahap pengumpulan data hingga uji dan validasi web interaktif pada streamlit. Berikut penjelasan dari tiap tahap tersebut:

1. Pengumpulan data
   Mengumpulkan sample data berupa *file* kode python dari *repository* github kelompok praktikum mahasiswa yang akan dianalisis untuk mendeteksi plagiarisme.
2. Pemrosesan data
   Tahap ini melibatkan pembersihan dan persiapan data. Seperti merapikan nama berkas-berkas python dari berbagai repository kemudian dikelompokkan dalam suatu folder sesuai dengan nomor soal praktikum.
3. Implementasi Algoritma
   Pada tahap ini, algoritma yang akan digunakan untuk deteksi plagiarisme diimplementasikan. Terdapat tiga metode implementasi:
   - Implementasi Levenshtein Distance:
     Mengimplementasikan algoritma Levenshtein Distance untuk mengukur kesamaan antara dua string atau teks.
   - Implementasi Rabin Karp:
     Mengimplementasikan algoritma Rabin-Karp yang menggunakan *hashing* untuk menemukan pola dalam string.

- Implementasi Ensemble *Voting Classifier* dari algoritma Levenshtein Distance dan Rabin Karp:
  Menggabungkan hasil dari kedua algoritma menggunakan teknik ensemble *Voting Classifier* untuk meningkatkan keakuratan deteksi plagiarisme.

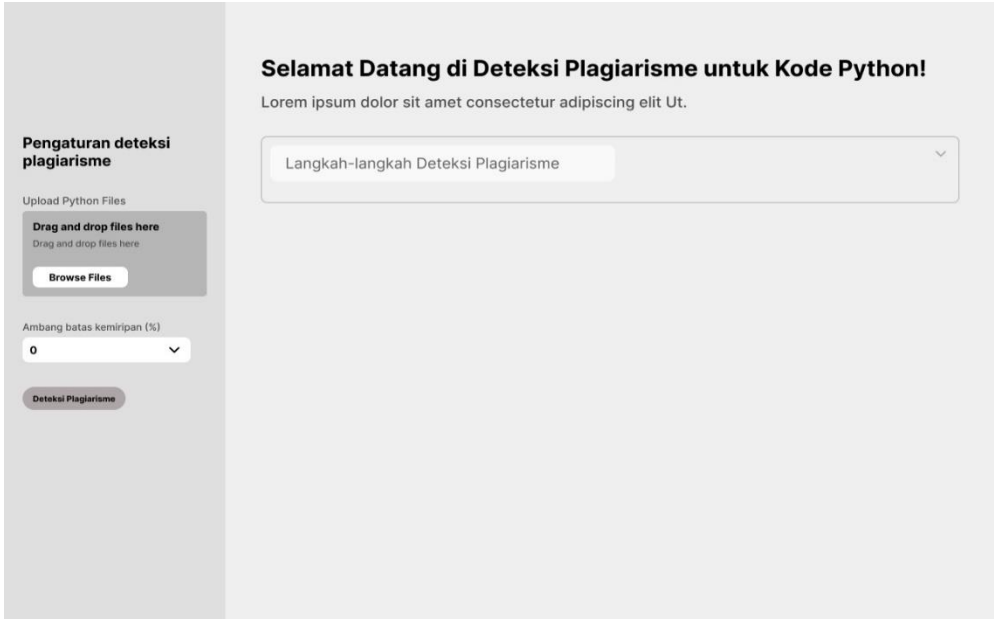
4. Pengembangan *Web* Interaktif Streamlit
   Mengembangkan aplikasi web interaktif menggunakan Streamlit untuk memudahkan pengguna dalam mendeteksi plagiarisme kode Python.
5. Pengujian dan Evaluasi Kinerja Tiap Algoritma
   Menguji dan mengevaluasi kinerja masing-masing algoritma serta ensemble *Voting Classifier* dalam mendeteksi plagiarisme. Analisis dilakukan untuk melihat kelebihan dan kekurangan masing-masing pendekatan. Pengujian dan evaluasi yang dilakukan terbagi menjadi dua yaitu:
   1) Dengan Berbagai Kondisi Plagiarisme:
      - Kondisi 1: Sampel dua file yang sama, namun hanya merubah penamaan variabel dan menambahkan komentar.
      - Kondisi 2: Sampel dua file yang sama dengan satu file yang memiliki urutan fungsi yang berbeda.
      - Kondisi 3: Sampel dua file yang sama dengan satu file menggunakan baris tambahan.
      - Kondisi 4: Sampel dua file yang memiliki struktur logika yang sama dengan penamaan variabel dan fungsi yang berbeda.
      - Kondisi 5: Sampel dua file yang dengan fungsi yang sama tetapi implementasi yang berbeda.
   2) Dengan menggunakan dataset Praktikum
      Setiap algoritma dievaluasi berdasarkan dua parameter yaitu Durasi pelatihan dan jumlah file plagiat yang terdeteksi.

2.4 Rancangan Dashboard pada Streamlit

Dalam penelitian ini, dibangun sebuah antarmuka pengguna atau *dashboard* menggunakan streamlit, yang merupakan framework open-source untuk membangun aplikasi web data science dengan Python. *Dashboard* ini dirancang untuk memudahkan pengguna dalam mendeteksi plagiarisme pada kode sumber Python. Dalam penelitian ini *dashboard* akan menampilkan beberapa bagian yaitu:

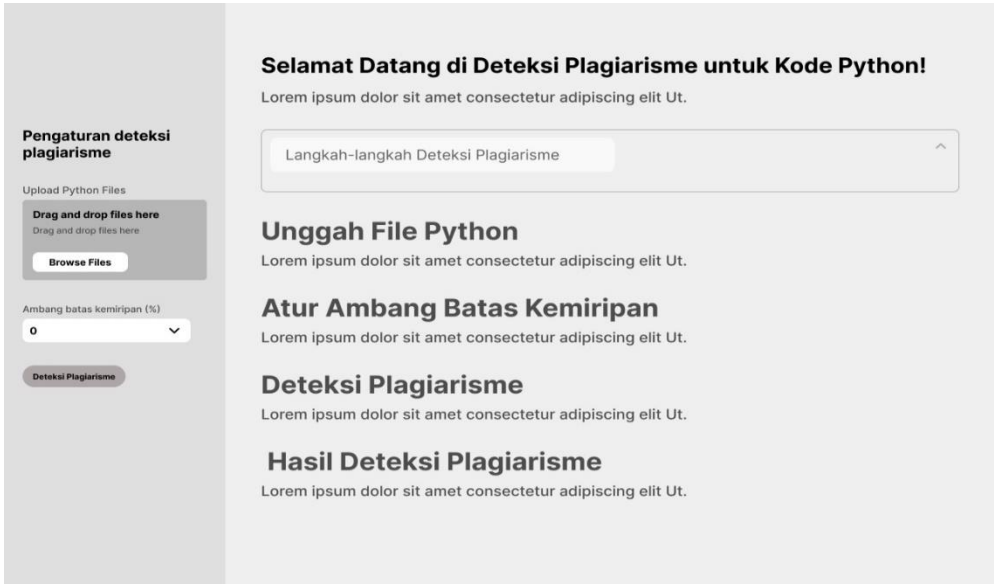
1. Rancangan tampilan utama *Dashboard*
   Pada Gambar 2 menunjukkan tampilan utama dari *dashboard* deteksi plagiarisme untuk kode Python, yang terdiri dari dua bagian utama. Bagian kiri menyediakan pengaturan deteksi, termasuk fitur unggah *file* dengan *drag and drop* atau *browse*, serta pengaturan ambang batas kesamaan (%). Bagian kanan menampilkan judul "Welcome to Plagiarism Detection for Python Code!", *dropdown* menu untuk langkah-langkah deteksi, dan tombol "Detect Plagiarism" untuk memulai proses. Desain ini bertujuan memudahkan pengguna mengunggah *file*, mengatur

preferensi, dan menjalankan deteksi plagiarisme dengan tampilan yang bersih dan intuitif.



Gambar 2. Rancangan Tampilan Utama *Dashboard*

2. Rancangan Tampilan Langkah Deteksi Plagiarisme
   Pada Gambar 3 menampilkan panduan langkah demi langkah bagi pengguna untuk mendeteksi plagiarisme pada kode Python.

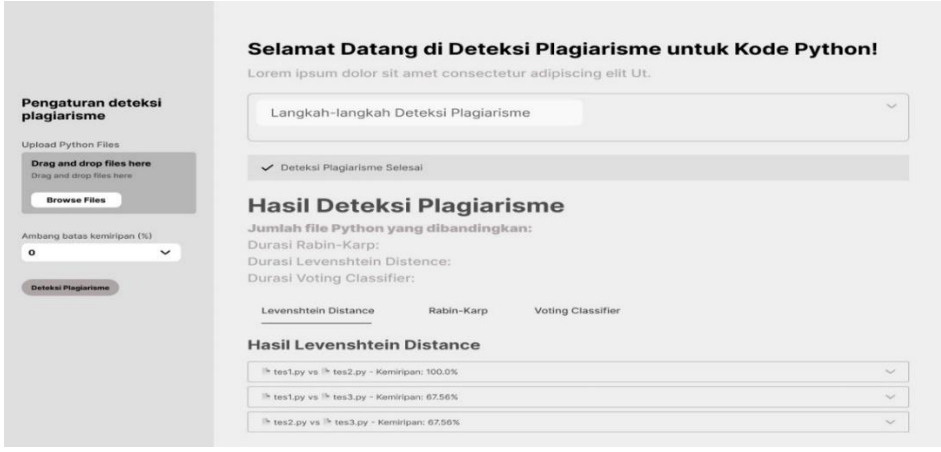


Gambar 3. Rancangan Tampilan Langkah Deteksi Plagiarisme

2. Rancangan Tampilan Proses Deteksi Plagiarisme
   Pada Gambar 4 menunjukkan tampilan saat proses deteksi plagiarisme sedang berjalan yaitu menampilkan *progress bar* yang menunjukkan presentasi *loading* proses deteksi.

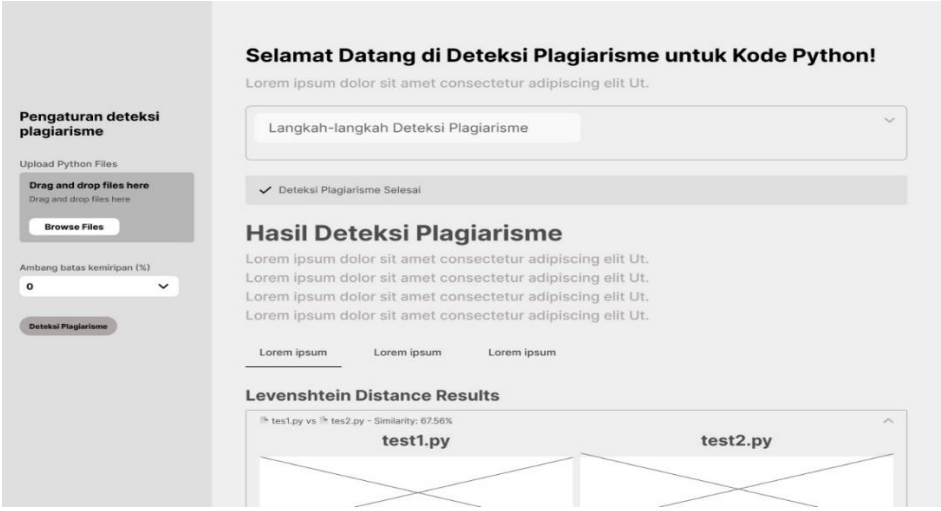
Gambar 4. Rancangan Tampilan Proses Deteksi Plagiarisme

3. Rancangan Tampilan Hasil Deteksi Plagiarisme
   Pada Gambar 5. menampilkan informasi hasil deteksi dari file-file yang telah dipilih, informasi ini berupa nama-nama pasangan file beserta presentasi kemiripannya dengan tab untuk melihat hasil dari masing-masing metode deteksi yaitu algoritma Levenshtein Distance, Rabin-Karp dan *Voting Classifier*.



Gambar 5. Rancangan Tampilan Hasil Deteksi Plagiarisme

4. Rancangan tampilan detail hasil perbandingan kode
   Pada Gambar 6 tampilan detail hasil perbandingan kode, pengguna dapat melihat perbandingan antara dua *file* python secara berdampingan. Setiap *file* ditampilkan dalam dua kolom yang memperlihatkan baris-baris kode yang dibandingkan.



Gambar 6. Rancangan Tampilan Detail Hasil Perbandingan Kode

2.5 Instrumen Penelitian

Adapun instrumen penelitian yang digunakan berupa perangkat keras (*hardware*) dan perangkat lunak (*software*).

2.5.1 Perangkat Keras (*Hardware*)

Perangkat keras yang digunakan berupa Laptop dengan spesifikasi, dapat dilihat pada Tabel 3.

Tabel 3. Spesifikasi Perangkat Keras

No.	Perangkat Keras	Spesifikasi
1.	Operating System (OS)	Windows 11 Home
2.	Processor	Intel(R) Core (TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz
3.	RAM	16,0 GB (15,7 GB usable)
4.	GPU	Radeon 620 Series

2.5.2 Perangkat Lunak (Software)

Pada Tabel 4 merupakan tabel informasi mengenai *software* dan *library* yang digunakan pada penelitian ini:

Tabel 4. Spesifikasi Perangkat Lunak

No	Perangkat Lunak	Kegunaan
1	Python	Bahasa pemrograman yang digunakan pada penelitian ini.
2	Visual Studio Code	Editor kode yang digunakan untuk bahasa pemrograman python.
3	Jupiter Notebook	Extensi Visual studio code untuk menulis dan menjalankan kode python, digunakan untuk analisis data dan eksperimen kode.
4	Streamlit	*Framework* yang digunakan dalam membangun aplikasi web interaktif untuk menampilkan hasil deteksi plagiarisme.
5	os	Modul standar Python untuk berinteraksi dengan sistem operasi, seperti manipulasi *file* dan direktori.
6	re	Modul standar Python untuk operasi regex, digunakan untuk memproses dan memanipulasi teks.
7	pandas	*Library* Python yang digunakan untuk analisis data, terutama untuk menangani dan menganalisis struktur data tabel.
8	numpy	*Library* Python yang digunakan untuk operasi numerik dan aljabar linear, berguna untuk perhitungan efisien.
9	tempfile	Modul standar Python untuk membuat *file* dan direktori sementara, berguna untuk menangani *file* sementara selama pemrosesan data.
10	io	Modul standar Python untuk operasi *input/output*, digunakan untuk manipulasi aliran data (*streams*).
11	tokenize	Modul standar Python untuk membagi string menjadi token, berguna untuk analisis kode sumber.
12	BaseEstimator	Kelas dasar dari scikit-learn untuk membuat estimator, digunakan dalam membangun model deteksi plagiarisme.
13	ClassifierMixin	Mixin *class* dari scikit-learn untuk membuat *classifier*, digunakan dalam mengembangkan model klasifikasi.

No	Perangkat Lunak	Kegunaan
14	time	Modul standar Python untuk mengukur waktu dan durasi eksekusi, digunakan untuk mengukur kinerja algoritma.
15	GitHub	Platform untuk versi kontrol dan kolaborasi kode, digunakan sebagai sumber kode penelitian dari *repository* kelompok praktikum