BAB I
PENDAHULUAN

A. Latar Belakang

Perkembangan teknologi memberikan kemudahan akses bagi mahasiswa untuk mendapatkan berbagai informasi yang dapat digunakan sebagai media belajar. Salah satu contohnya adalah mahasiswa dapat dengan mudah mendapatkan contoh *source code* pemrograman secara daring. Meskipun mendukung proses pembelajaran, kondisi ini dapat menimbulkan dampak negatif pada mahasiswa dalam menyelesaikan tugas kuliah. Mahasiswa dapat dengan mudah menggunakan *source code* yang diperoleh tanpa memahami isi dan logika program. Hal ini dapat meningkatkan potensi terjadinya plagiarisme *source code* pada tugas pemrograman. Plagiarisme *source code* adalah penggunaan kembali kode milik orang lain, sebagian atau seluruhnya, tanpa atribusi yang semestinya dalam konteks tugas di perguruan tinggi [1].

Pada saat ini sudah ada berbagai macam alat untuk deteksi plagiarisme *source code*, contohnya adalah MOSS, JPlag, Plaggie, Sim, dan Sherlock. Meskipun sudah terdapat alat deteksi plagiarisme *source code*, namun alat tersebut masih memiliki keterbatasan. Alat deteksi yang sudah ada saat ini masih memiliki keterbatasan untuk mendeteksi plagiarisme *source code* yang disamarkan dengan cara mengganti nama variabel, melakukan perubahan pada *control flow*, memecah atau menggabungkan *function*, serta melakukan penambahan atau penghapusan komentar pada program [2]. Bahkan saat ini ada alat khusus yang bisa melakukan transformasi otomatis pada *source code* untuk menghasilkan variasi unik pada kode program sehingga dapat digunakan untuk mengakali aplikasi deteksi plagiarisme seperti MOSS dan JPlag [3].

Berdasarkan temuan tersebut penelitian ini membuat aplikasi deteksi plagiarisme *source code* berbasis *machine learning* yang menggunakan enam parameter sebagai fitur. Parameter utama pada

penelitian ini diartikan sebagai ukuran tingkat kemiripan *source code* pada kesamaan baris logika, urutan penyelesaian program, kemiripan komentar, serta penamaan variabel.

Dengan rancangan ini, diharapkan sistem dapat membantu dosen dan asisten dosen dalam menilai tugas pemrograman mahasiswa bahwa tugas tersebut terindikasi plagiarisme atau tidak. Dosen dan asisten dosen dapat melihat skor kemiripan program, fitur atau parameter yang paling berpengaruh dalam keputusan program, dan *highlight* bagian program yang ditulis sama persis. Dengan adanya informasi tersebut, dosen dan asisten dosen dapat melakukan pengecekan ulang jika terdapat kesalahan pada sistem.

B. Rumusan Masalah

Rumusan masalah yang menjadi fokus utama dalam penelitian ini adalah bagaimana mengembangkan aplikasi web yang mampu membantu dosen dan asisten dosen dalam mendeteksi plagiarisme *source code* Java pada tugas kuliah mahasiswa secara efektif dan mudah digunakan.

C. Batasan Masalah

Untuk menjaga fokus dan konsistensi, maka dibuat batasan masalah sebagai berikut:

1. Penelitian ini hanya membahas plagiarisme *source code* Java pada tugas pemrograman mahasiswa.
2. Analisis dilakukan pada dua *source code* dalam satu tugas yang sama. Sistem tidak dirancang untuk melakukan unggah *file* dan analisis secara massal terhadap seluruh berkas tugas mahasiswa.
3. Sistem hanya berfungsi sebagai alat bantu untuk mengidentifikasi indikasi plagiarisme. Keputusan akhir tetap berada pada dosen atau asisten dosen melalui pemeriksaan manual berdasarkan informasi yang berada di web.
4. Sistem diimplementasikan sebagai aplikasi web mandiri. Integrasi dengan sistem lain, seperti LMS atau platform eksternal tidak dibahas dalam penelitian ini.

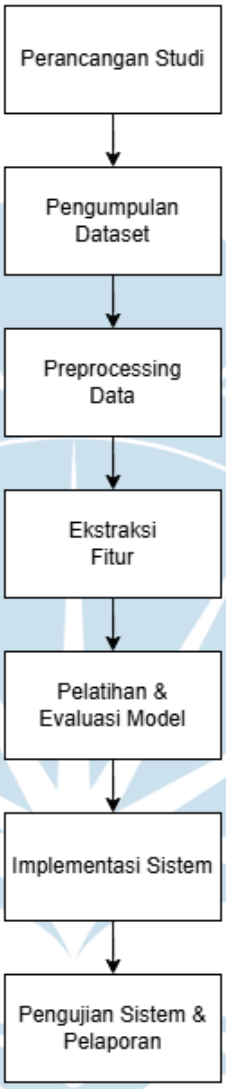
5. Sistem ini hanya mendeteksi plagiarisme berdasarkan kemiripan *token*, pola penamaan variabel, kemiripan komentar, urutan baris logika, kemiripan teks, kemiripan baris logika.
6. Sistem ini hanya mendukung pemeriksaan tugas Java yang terdiri dari satu *file* dan tidak mencakup tugas dengan lebih dari satu *file*.

D. Tujuan Penelitian

Tujuan dari penelitian ini adalah mengembangkan aplikasi web yang mampu membantu dosen dan asisten dosen dalam mendeteksi plagiarisme *source code* Java pada tugas kuliah mahasiswa secara efektif dan mudah digunakan.

E. Metode Penelitian

Pengembangan sistem deteksi plagiarisme *source code* Java pada tugas pemrograman mahasiswa dilakukan menggunakan model *Waterfall.* Model ini menerapkan tahapan pengembangan yang dilakukan secara berurutan dan terstruktur, dimulai dari perancangan studi, pengumpulan *dataset*, *preprocessing*, ekstraksi fitur, pelatihan dan evaluasi model, implementasi sistem, pengujian sistem, dan pelaporan. Model *Waterfall* dipilih karena kebutuhan sistem telah didefinisikan dengan jelas sejak awal, serta proses pengembangan bersifat bertahap dan tidak memerlukan perubahan kebutuhan yang bersifat *iteratif*. Alur pengembangan sistem menggunakan model *Waterfall* pada penelitian ini ditunjukkan pada Gambar 1.1



Gambar 1. 1 Model Waterfall dalam Pengembangan Sistem Deteksi Plagiarisme

1. **Perancangan studi**

   Tahap awal penelitian ini adalah menentukan arah dan ruang lingkup penelitian agar sesuai dengan konteks tugas pemrograman mahasiswa di lingkungan perkuliahan.

2. **Pengumpulan *Dataset***

   Dataset yang digunakan dalam penelitian ini adalah ConPlag sebagai *dataset* utama yang berisi *file source code* Java. *Dataset* ini dipilih

karena merepresentasikan kasus plagiarisme pada tugas pemrograman mahasiswa sehingga sesuai dengan konteks penelitian.

3. ***Preprocessing* data**

   Tahap *preprocessing* data dilakukan untuk menyiapkan *source code* agar dapat dianalisis secara konsisten. Proses ini mencakup penyeragaman format, validasi pasangan source code, pembersihan data, pembagian data, dan normalisasi representasi kode, sehingga perbandingan antar *source code* dapat dilakukan secara objektif dan adil.

4. **Ekstraksi fitur**

   Tahap ekstraksi fitur dilakukan untuk mengubah data hasil *preprocessing* menjadi nilai numerik yang merepresentasikan parameter kemiripan. Nilai numerik yang dihasilkan akan digunakan sebagai data untuk proses pelatihan model dan serta sebagai dasar penilaian sistem dalam mendeteksi indikasi plagiarisme.

5. **Pelatihan dan evaluasi model**

   Tahap pelatihan model dilakukan menggunakan hasil dari ekstraksi fitur. Model yang digunakan adalah *Logistic regression*, *Random forest*, SVM, dan *Gradient Boosting*.

6. **Implementasi sistem**

   Pada tahap ini, Model hasil pelatihan diintegrasikan ke dalam aplikasi web menggunakan *Streamlit*. Implementasi ini bertujuan agar sistem dapat langsung digunakan oleh dosen dan asisten dosen.

7. **Pengujian sistem dan pelaporan**

   Tahap pengujian sistem dilakukan untuk memastikan aplikasi dapat berjalan sesuai rancangan. Selanjutnya, hasil pengujian dan seluruh proses disusun dalam bentuk laporan.

F. Sistematika Penulisan

Skripsi ini disusun dalam beberapa bab agar pembahasan runtut dan mudah diikuti. Adapun sistematika penulisannya sebagai berikut.

1. BAB I: Pendahuluan

Bab ini menjelaskan latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, ringkasan metode, serta sistematika penulisan. Bab ini juga memberi gambaran umum alur penelitian.

2. BAB II: Tinjauan Pustaka

   Bab ini mengulas penelitian terkait, konsep dan temuan terdahulu yang relevan, serta perbandingan singkat dengan penelitian ini untuk menegaskan gap penelitian.

3. BAB III: Landasan Teori

   Bab ini menyajikan teori dan konsep yang mendasari penelitian, termasuk definisi dan penjelasan metode yang digunakan.

4. BAB IV: *Dataset* Pengembangan Model

   Bab ini menjelaskan sumber *dataset*, *preprocessing* data, perancangan fitur, proses pelatihan dan evaluasi model, serta kalibrasi ambang keputusan.

5. BAB V: Implementasi Model dan Pengujian Sistem

   Bab ini menjelaskan implementasi model ke dalam aplikasi web berbasis *Streamlit*, alur kerja sistem serta hasil pengujian fungsional.

6. BAB VI: Penutup

   Bab ini menyajikan kesimpulan hasil penelitian secara keseluruhan dan memberikan saran untuk penelitian selanjutnya.