BAB I
PENDAHULUAN

A. Latar Belakang

Pemrograman dalam lingkup ilmu komputer (informatika) merupakan sebuah proses menciptakan suatu program komputer untuk menjalankan sebuah tugas tertentu berdasarkan sejumlah urutan instruksi sekuensial. Urutan instruksi sebuah program pada umumnya ditulis dalam bentuk kode sumber (*source code*) oleh seorang pengembang perangkat lunak (*software developer*). Pada umumnya, semakin kompleks sebuah kode sumber, semakin penting dilakukan pemisahan berkas (*file*) agar program tersebut masih dapat dirawat (*maintain*) oleh pengembang lain dan meminimalisir refaktorisasi (*refactoring*) [1].

Dalam menulis kode sumber, seorang pengembang dapat menggunakan cuplikan kode sumber yang sudah dikembangkan pengembang lain untuk menghemat waktu pengembangan. Di era saat ini, seorang pengembang dapat mencari cuplikan kode sumber pada berbagai media, misal pada forum daring *StackOverflow*, bertanya pada *AI Chatbot* seperti *ChatGPT* atau *Google Gemini*, atau bahkan meminta pada sesama pengembang. Menggunakan kembali cuplikan kode sumber pengembang lain (*code reuse*) sebenarnya merupakan hal yang wajar dalam proses pengembangan perangkat lunak untuk menghemat waktu pengembangan [2].

Hal ini memiliki pandangan berbeda dalam dunia akademik. Seorang pelajar atau mahasiswa yang menggunakan cuplikan kode yang diberikan temannya, baik sebagian maupun seluruh, dapat dinyatakan telah melakukan plagiarisme [3][4]. Plagiarisme dapat dimulai dengan hal-hal yang kecil, misal dengan menyalin sebuah *function* atau *method* dari pengembang lain tanpa memberi sumber secara cukup; hingga hal-hal yang besar, misal dengan menyalin semua kode sumber pada suatu berkas (*file*). Dalam dunia akademik, ada batasan toleransi yang diberikan berkaitan dengan plagiarisme, walau batasan ini sering tidak dapat didefinisikan dengan jelas [5][6].

Mendeteksi plagiarisme pada kode sumber bukanlah sesuatu yang mudah dilakukan. Pemeriksaan plagiarisme sering memakan banyak waktu serta membutuhkan ketelitian [7][8]. Hal ini semakin dipersulit apabila sebuah kode sumber terdiri atas lebih dari 1 (satu) berkas (*file*) serta menggunakan banyak bahasa pemrograman, seperti dalam pembangunan sebuah situs web (*website*).

Melihat skenario yang telah dipaparkan di atas, maka penting untuk memiliki sebuah algoritma terpadu yang dapat mendeteksi plagiarisme pada sebuah proyek perangkat lunak multi-*file* dengan berbagai bahasa pemrograman. Selama ini, sudah terdapat beberapa algoritma hasil penelitian yang dikembangkan untuk mendeteksi plagiarisme kode sumber. Sebagian besar algoritma ini hanya dapat digunakan untuk mengukur tingkat kemiripan satu berkas (*file*) saja, dan bukan untuk sebuah proyek perangkat lunak multi-*file*.

Penelitian ini dimaksud untuk mengadaptasi alat Dolos serta algoritma jarak *levenshtein* untuk mengukur tingkat kemiripan kode sumber proyek perangkat lunak multi-*file*, serta membuat sebuah sistem informasi agar pengguna dapat mengakses algoritma ini. Algoritma dan sistem ini diharapkan dapat bermanfaat bagi para akademisi yang mengampu pembelajaran terkait pengembangan proyek perangkat lunak multi-*file*, serta pihak-pihak yang mengutamakan integritas dalam pengembangan perangkat lunak.

B. Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan di atas, dapat ditarik beberapa rumusan masalah yang akan diteliti, antara lain:

1. Bagaimana cara mengadaptasi alat Dolos dan algoritma jarak *levenshtein* untuk memeriksa tingkat kemiripan kode sumber pada proyek perangkat lunak multi-*file*?
2. Bagaimana cara membuat sebuah sistem informasi dengan menggunakan prinsip desain modern agar pengguna dapat mengakses algoritma yang telah diimplementasi?

3. Seberapa tinggi akurasi dan/atau presisi dari algoritma yang telah diimplementasi jika diuji terhadap *dataset* yang ada?

C. Batasan Masalah

Agar tujuan utama penelitian ini dapat tercapai dalam waktu yang telah ditentukan, perlu ditarik batasan masalah sebagai berikut:

1. Algoritma yang diimplementasikan hanya akan berfokus pada proyek perangkat lunak multi-*file*.
2. Algoritma yang diimplementasikan hanya dapat memeriksa tingkat kesamaan pada beberapa jenis proyek perangkat lunak, antara lain:
   a. Proyek perangkat lunak dengan *framework*, yakni *Laravel*, *Vue.js* CLI, *Vite React.js*, serta Java *NetBeans*.
   b. Proyek perangkat lunak tanpa *framework*, yakni C/C++, PHP, dan HTML.
3. Proses perbandingan hanya akan dibatasi pada berbagai proyek perangkat lunak yang diunggah pengguna dan tidak melakukan perbandingan dengan *repository* publik yang tersedia di internet.
4. Sistem informasi yang dibangun hanya sebatas pada sebuah situs web (*web site*).

D. Tujuan Penelitian

Penelitian ini bertujuan untuk mengadaptasi alat *Dolos* karya Maertens, R. [9] dan melalui pengukuran jarak *levenshtein* untuk mengukur tingkat kesamaan (mendeteksi plagiarisme) kode sumber antara dua atau lebih proyek perangkat lunak multi-*file*, serta membuat sebuah sistem informasi yang memungkinkan pengguna menjalankan algoritma serta melihat hasil pemeriksaan dan analisis.

E. Metode Penelitian

Penelitian ini akan menggunakan metode pengembangan *agile*. Metode *agile* merupakan pendekatan yang fleksibel dan iteratif dalam proses pengembangan perangkat lunak, dan memungkinkan perubahan yang cepat terhadap perangkat lunak yang telah dikembangkan berdasarkan kebutuhan terbaru. Metode ini dipilih karena proses pengembangan algoritma sering melibatkan iterasi

dan evaluasi untuk mencapai hasil yang optimal. Metode *agile* juga memungkinkan sistem informasi dibangun secara adaptif terhadap perkembangan implementasi algoritma, sehingga algoritma yang diimplementasi dapat diubah tanpa harus mengulang pembangunan sistem informasi dari awal [10]. Adapun berbagai tahapan yang akan ditempuh antara lain:

1. Analisis Masalah dan Kebutuhan

   Tahap pertama dari penelitian ini adalah analisis masalah dan kebutuhan. Analisis masalah diperlukan agar algoritma yang akan diimplementasikan dan sistem yang akan dibangun memiliki batasan dan ruang lingkup penelitian yang memadai. Selain analisis masalah, terdapat pula analisis kebutuhan untuk mendapat gambaran terkait kebutuhan penelitian, misalnya perangkat dan instrumen yang akan digunakan.

2. Studi Literatur

   Tahap kedua dari penelitian ini adalah studi literatur. Tahap ini penting dilaksanakan guna mencari calon solusi yang tepat digunakan dengan cara membaca referensi dan literatur terdahulu. Dalam tahap ini, akan diperoleh berbagai literatur yang berkaitan dengan konsep plagiarisme, metode pengukuran tingkat kemiripan *string*, serta berbagai teknologi untuk membangun sistem informasi berbasis web. Dari ketiga literatur yang disebutkan, tahap ini akan lebih fokus pada pencarian literatur yang berkaitan dengan algoritma pemeriksaan tingkat kesamaan *string*.

3. Desain Algoritma dan Sistem

   Tahap ketiga dari penelitian ini adalah desain algoritma dan sistem. Dalam tahap ini, akan berbagai literatur yang disebutkan akan dievaluasi berkaitan dengan efektivitas, performa, dan adaptabilitasnya. Dalam proses ini, akan dikaji desain dari algoritma, metode klasterisasi, serta desain dari sistem informasi. Integrasi antara algoritma dengan sistem informasi akan turut dikaji dalam tahapan ini.

4. Implementasi Algoritma

   Pada tahap ini, desain algoritma yang telah dipaparkan pada tahap ketiga akan diimplementasikan. Algoritma tersebut akan menggunakan Node.js sebagai lingkungan pengembangan dan mengadaptasi alat *Dolos* oleh Maertens, R. [9] untuk memeriksa tingkat kesamaan kode sumber. Apabila hasil luaran dari alat tersebut kurang memuaskan, maka pengukuran jarak *levenshtein* akan digunakan sebagai gantinya. Tahapan desain dan implementasi algoritma dapat diulang lagi apabila hasil yang diperoleh dirasa kurang memuaskan.

5. Pembangunan Sistem Informasi

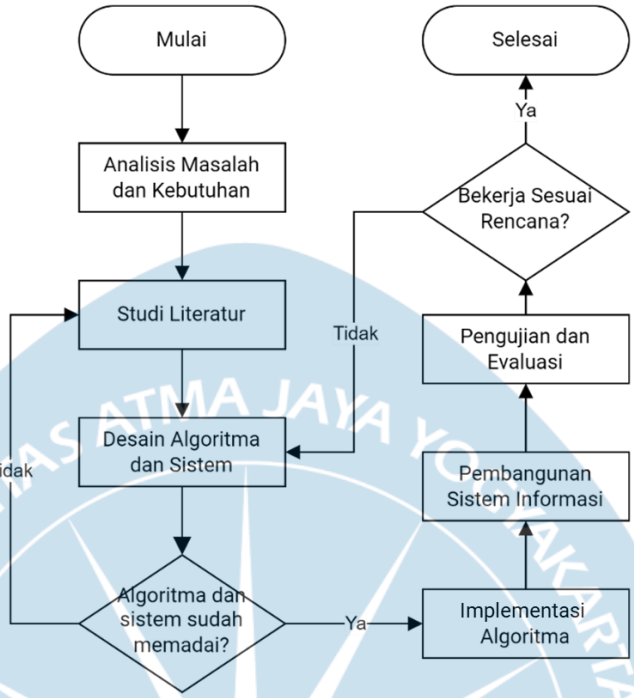
   Setelah proses pengembangan algoritma, integrasi algoritma tersebut ke dalam sebuah sistem informasi akan diterapkan, dengan menggunakan *Express*.js sebagai *backend* dan *React*.js sebagai *frontend*. Di sisi *backend*, akan dibangun sebuah *application programming interface* (*API*) dengan berbagai *endpoint* yang dapat diakses oleh pengguna melalui sebuah situs *frontend*. Di sisi *frontend*, akan dibangun antarmuka pengguna yang sederhana namun responsif.

6. Pengujian dan Evaluasi

   Di akhir penelitian, akan ada pengujian dan evaluasi untuk memastikan akurasi dan presisi sistem dalam mengukur tingkat kemiripan serta integrasinya dengan sistem informasi yang telah dibuat. Beberapa metode pengujian yang akan digunakan antara lain *performance testing* dan *functional testing* (pengujian fungsional).

   a. *Performance testing* akan digunakan untuk menguji performa dan akurasi dari algoritma yang telah diimplementasi [11][12].
   b. *Functional testing* (pengujian fungsional) akan digunakan untuk memastikan sistem informasi yang dibangun memenuhi kebutuhan fungsional yang telah ditetapkan di awal [13].

Apabila digambarkan dalam bentuk diagram alir, maka tahapan yang akan ditempuh dalam penelitian ini dapat diamati pada Gambar 1.1.



Gambar 1.1 Diagram Metode Penelitian

F. Sistematika Penulisan

BAB I PENDAHULUAN

Bab ini berisi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, metode penelitian, serta sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini berisi penelitian-penelitian terdahulu yang berkaitan dengan penelitian yang dilakukan. Pada akhir bab ini, terdapat sebuah tabel perbandingan penelitian yang dilakukan terhadap berbagai penelitian terdahulu.

BAB III LANDASAN TEORI

Bab ini berisi penjelasan mengenai berbagai konsep teoritis yang dilakukan dalam penelitian.

BAB IV ANALISIS DAN PERANCANGAN SISTEM

Bab ini berisi analisis sistem dan algoritma, ruang lingkup masalah, perspektif produk, serta berbagai perancangan sistem.

BAB V IMPLEMENTASI DAN PENGUJIAN SISTEM

Bab ini berisi hasil implementasi dan pengujian sistem berdasarkan berbagai konsep yang telah dipaparkan pada bab III dan perancangan yang dipaparkan pada bab IV.

BAB VI PENUTUP

Bab ini berisi kesimpulan dari penelitian yang dilakukan serta saran pengembangan sistem kedepannya.