**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI INFORMASI**

**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**

**USULAN TUGAS AKHIR**

# IDENTITAS PENGUSUL

**NAMA : Putu Linda Indita Sari**

**NRP : 5110100121**

**DOSEN WALI : Bilqis Amaliah, S.Kom, M.Kom.**

**DOSEN PEMBIMBING : 1. Dr. Ir. Siti Rochimah, MT.  
 2. Abdul Munif, S.Kom, M.Sc.**

# JUDUL TUGAS AKHIR

“Kakas Pendeteksi Plagiasi Kode Sumber pada Tugas Pemrograman dalam Kelas Virtual *Moodle*”

# LATAR BELAKANG

Plagiasi merupakan permasalahan yang sangat sering terjadi dalam berbagai bidang, di mana ide dan kreativitas seseorang ditiru oleh orang lain. Dalam dunia pendidikan, kasus plagiasi ini sangat sering ditemukan. Kasus-kasus ini bisa terjadi mulai dari pengumpulan tugas hingga publikasi ilmiah.

Pendidikan teknologi informasi dan komunikasi umumnya mencakup tugas-tugas pemrograman di dalamnya. Tugas pemrograman sering kali bersifat sama untuk seluruh peserta kelas. Tujuannya adalah untuk melatih dan menguji seberapa jauh pemahaman peserta kelas mengenai suatu algoritma atau untuk melihat seberapa kreatif peserta kelas dalam memecahkan suatu permasalahan. Akan tetapi, kasus plagiasi kode sumber pada tugas pemrograman tersebut masih banyak ditemui. Plagiasi ini umumnya dilakukan oleh peserta kelas yang sebenarnya kurang memahami algoritma, sehingga hanya meniru kode sumber milik orang lain dengan hanya mengubah sedikit bagian kode sumber tersebut.

Kasus plagiasi pada kode sumber memang sulit dideteksi jika penilaian tugas pemrograman tersebut dilakukan secara manual. Selain karena kode sumber yang bersifat kompleks, jumlah peserta kelas yang besar juga menjadi salah satu faktor yang menyulitkan pendeteksian kasus plagiasi ini. Dalam setiap pengumpulan tugas pemrograman yang berjumlah besar, seorang penguji akan disulitkan dalam pendeteksian plagiasi untuk semua kode sumber secara adil dengan standar yang sama. Pada umumnya yang terjadi adalah terdeteksinya satu atau dua kasus plagiasi, namun ada kemungkinan beberapa kasus plagiasi lainnya luput dari penilaian penguji kode sumber. Kenyataan ini mendorong kebutuhan akan suatu kakas yang dapat membantu pendeteksian kasus-kasus plagiasi dalam kode sumber, sehingga penguji dapat memberikan penilaian secara adil bagi seluruh peserta kelas.

Untuk mempermudah proses pembelajaran, baik akses materi maupun pengumpulan dan penilaian tugas, penggunaan kelas virtual dalam dunia pendidikan mulai diminati. Begitu pula dalam pendidikan teknologi informasi dan komunikasi, pembelajaran telah banyak didukung oleh adanya kelas virtual. Salah satu *platform* pembuatan kelas virtual yang populer saat ini adalah *Moodle*. Dengan adanya kelas virtual tersebut, pengumpulan tugas, termasuk tugas pemrograman, menjadi lebih mudah. Pada kelas virtual tersebut juga dapat ditambahkan kakas untuk mendeteksi plagiasi pada setiap tugas yang dikumpulkan. Hal ini dapat membantu penguji untuk menilai apakah dalam tugas tersebut telah terjadi plagiasi atau tidak. Fitur-fitur yang cukup lengkap dan mudah ini telah menjadikan kelas virtual, seperti *Moodle*, semakin banyak digunakan dalam dunia pendidikan.

Sayangnya, beberapa kakas pendeteksi plagiasi kode sumber yang sudah disediakan *Moodle* memiliki sedikit kekurangan. Salah satunya berasal dari algoritma pendeteksian berbasis *token* yang mendasari kakas [3]. Beberapa kakas kurang bisa mendeteksi plagiasi yang terjadi apabila dalam kode sumber yang diuji deklarasi fungsi dilakukan secara terbalik, meskipun isi fungsi sama persis [5]. Hal ini dapat mengakibatkan luputnya beberapa kode sumber yang sebenarnya plagiat dari pengujian. Untuk itu dibutuhkan kakas dengan algoritma berbeda yang dapat mendeteksi plagiasi bahkan jika posisi deklarasi fungsi pada kode sumber diubah.

Kakas pendeteksi plagiasi kode sumber yang diusulkan akan dibuat menggunakan metode yang umum digunakan untuk mendeteksi klon dalam evolusi perangkat lunak. Metode yang dimaksud akan membandingkan AST (*Abstract Syntax Tree*) dari kode sumber yang akan diuji. Perbedaan antara deteksi klon dengan deteksi plagiasi terletak pada kode sumber yang dibandingkan. Dalam deteksi klon pembandingan AST dilakukan hanya dalam satu proyek, sedangkan untuk deteksi plagiasi pembandingan AST dilakukan pada dua proyek berbeda. Jika terdapat AST yang serupa dari dua kode sumber dalam proyek berbeda, maka ada kemungkinan telah terjadi plagiasi di antara kedua kode sumber tersebut.

Perbandingan AST dilakukan berdasarkan fungsi dalam kode sumber. Kemiripan AST fungsi satu dengan fungsi lain di kode sumber berbeda akan dianalisis dengan metode LCS (*Longest Common Subsequence*). Dengan menggunakan LCS dalam membandingkan AST setiap fungsi, maka plagiasi dengan cara menyisipkan atau menghilangkan baris kode masih dapat dideteksi. Deteksi plagiasi dengan membandingkan AST tiap fungsi juga membantu mendeteksi plagiasi yang dilakukan dengan menukar posisi deklarasi fungsi. Sebab, AST tiap fungsi dari satu kode sumber akan dibandingkan dengan semua AST fungsi dalam kode sumber lainnya tanpa memperhatikan posisi deklarasi fungsi tersebut.

Sering kali tugas pemrograman dalam suatu kelas diberikan untuk menilai pemahaman peserta mengenai algoritma yang telah diajarkan di dalam kelas. Dalam kasus ini kemungkinan besar akan terjadi kemiripan pada bagian fungsi tertentu pada tugas yang dikumpulkan. Khususnya jika saat penjelasan algoritma telah diberikan *pseudocode* yang menunjukkan jalannya algoritma tersebut. Saat diuji dengan kakas pendeteksi plagiasi, tentunya potongan program dengan algoritma yang telah diajarkan tersebut akan dideteksi sebagai plagiat. Untuk mengatasi hal ini, maka pada kakas yang diusulkan, penguji dapat menambahkan potongan program sesuai algoritma yang diajarkan ke dalam kakas sebagai pengecualian dalam deteksi plagiasi. Kakas lalu membandingkan apakah potongan fungsi yang sedang diuji sama dengan potongan program yang dijadikan pengecualian tersebut. Jika sama, maka deteksi plagiasi pada potongan program tersebut tidak akan dianggap.

Kakas pendeteksi plagiasi kode sumber yang diusulkan akan dibuat sebagai *web service*, yang nantinya dapat digunakan sebagai *plugin* pada kelas virtual dalam *Moodle*. Dengan menjadikan kakas sebagai *plugin* kelas virtual, maka diharapkan pendeteksian plagiasi kode sumber dapat dilakukan dengan mudah oleh para pengajar dalam dunia pendidikan.

Kakas akan menerima masukan berupa dokumen tugas yang telah dikumpulkan dalam satu kelas virtual termasuk potongan program yang dijadikan pengecualian (jika ada). Kakas lalu mendeteksi adanya plagiasi dengan membangun AST dan membandingkan AST tersebut menggunakan metode LCS. Laporan hasil deteksi plagiasi kemudian dikirim kembali ke pengguna berupa persentase plagiasi serta bagian program yang merupakan plagiat.

Meskipun ditujukan sebagai kakas pendeteksi plagiasi, kakas ini hanya menunjukkan kemiripan satu kode sumber dengan kode sumber lainnya. Penilaian akhir mengenai apakah kemiripan tersebut benar-benar merupakan plagiasi atau hanya mirip secara kebetulan dikembalikan lagi kepada pengajar/penguji masing-masing. Kakas ini akan membantu menunjukkan bagian kode yang mirip beserta persentase kemiripannya. Selain itu kakas ini juga akan mengesampingkan kemiripan yang terjadi apabila kemiripan tersebut sama dengan potongan program yang dijadikan pengecualian dalam deteksi plagiasi.

# RUMUSAN MASALAH

Permasalahan yang akan diangkat dalam Tugas Akhir ini adalah sebagai berikut.

1. Bagaimana arsitektur untuk pembuatan kakas pendeteksi plagiasi kode sumber pada tugas pemrograman dalam kelas virtual *Moodle*?
2. Bagaimana cara menganalisis kumpulan program untuk menemukan plagiasi kode sumber yang terjadi di antaranya?
3. Bagaimana membuat kakas pendeteksi plagiasi kode sumber pada tugas pemrograman yang mudah digunakan dan memberikan hasil yang informatif?

# BATASAN MASALAH

Bahasan dalam Tugas Akhir ini memiliki beberapa batasan masalah, yakni sebagai berikut.

1. Kakas pendeteksi plagiasi kode sumber dibangun sebagai *web service* yang kemudian dijadikan *plugin* untuk *Moodle*.
2. *Web service* pendeteksi plagiasi dibangun dengan bahasa Java. *Plugin* pada *Moodle* dibangun menggunakan PHP.
3. AST untuk tiap kode sumber dibuat dengan *parser* ANTLR.
4. Bahasa pemrograman yang dapat diuji yaitu: C++, C#, dan Java.
5. Bahasa pemrograman yang digunakan kode sumber yang diuji harus ditentukan sebelum memulai pengujian.
6. Kakas memberikan laporan kemiripan antara satu kode sumber dengan kode sumber lain dari pengumpulan tugas dalam satu kelas beserta bagian-bagian kode yang mirip.

# TUJUAN PEMBUATAN TUGAS AKHIR

Tugas Akhir ini dikerjakan untuk mencapai tujuan merancang dan membangun kakas yang dapat digunakan untuk mendeteksi kode sumber yang dicurigai sebagai plagiat pada tugas pemrograman dalam kelas virtual *Moodle*.

# MANFAAT TUGAS AKHIR

Tugas Akhir ini dikerjakan dengan harapan dapat memberikan kontribusi bagi pendidikan, khususnya pendidikan Teknologi Informasi dalam hal pemrograman. Pengajar akan dipermudah dalam menemukan bagian program yang dicurigai sebagai plagiat, sehingga pengajar lebih fokus dalam mendeteksi plagiasi dalam pengerjaan tugas. Dengan adanya perangkat lunak ini diharapkan dapat mempermudah penguji/pengajar dalam mencegah tindakan plagiasi dalam tugas pemrograman.

# TINJAUAN PUSTAKA

## Plagiasi

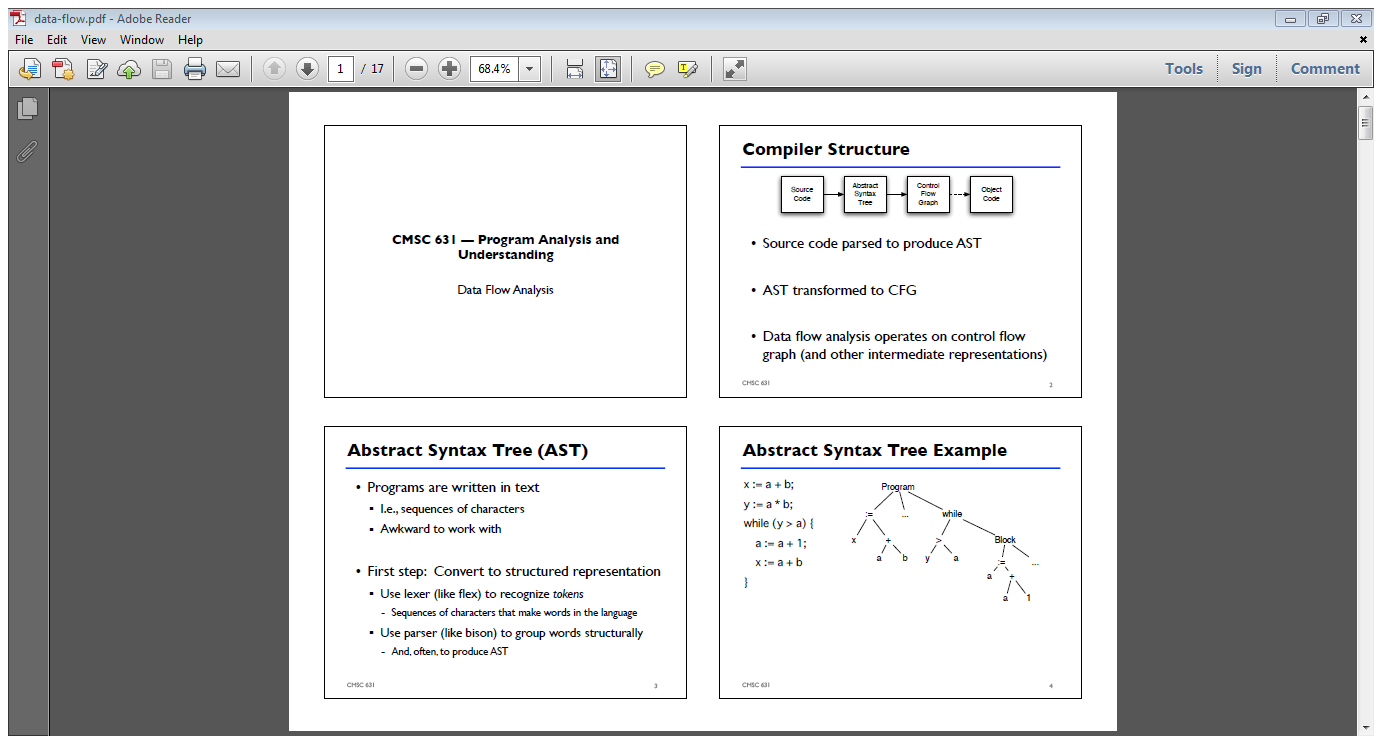
Plagiasi merupakan tindakan membuat kembali hasil karya orang lain tanpa menyebutkan sumber dari hasil karya tersebut dan menyajikannya seolah-olah merupakan hasil karya sendiri [1]. Kasus plagiasi paling sering ditemui dalam dunia pendidikan. Misalnya, saat siswa menyalin isi dari literatur, seperti buku, makalah, dan sebagainya tanpa menyebutkan sumbernya, atau ketika siswa menyalin tugas milik temannya dengan melakukan sedikit perubahan, kemudian mengumpulkan tugas tersebut sebagai miliknya sendiri.

Plagiasi pada kode sumber juga sering ditemukan dalam dunia pendidikan, khususnya bidang Teknologi dan Informasi. Kasus ini sering terjadi pada saat pengerjaan tugas-tugas pemrograman. Terkadang ada beberapa siswa yang menyalin pekerjaan siswa lainnya, dan dengan sedikit modifikasi kemudian mengumpulkan kode sumber tersebut seolah-olah merupakan miliknya sendiri.

Secara umum, plagiasi kode sumber dapat dibedakan menjadi dua tingkat yakni plagiasi tingkat rendah dan plagiasi tingkat tinggi [9]. Plagiasi tingkat rendah memiliki ciri berupa penyalinan sebagian besar kode sumber dengan sedikit modifikasi pada spasi dan komentar, perubahan nama variabel, fungsi, maupun kelas, membalik urutan kode (tanpa mempengaruhi status akhir), serta penambahan baris redundan pada kode sumber. Sementara plagiasi tingkat tinggi sudah meliputi perubahan kontrol program serta mengambil kode dari banyak sumber lain dan mencampurnya dengan kode buatan sendiri.

## AST (*Abstract Syntax Tree*)

Pohon sintaks abstrak atau AST (*Abstract Syntax Tree*) merupakan bentuk pohon dari sebuah struktur program. AST menampilkan struktur inti dari kode sumber dalam bentuk pohon, tanpa mempedulikan detil sintaks yang tidak begitu penting[4]. Perbedaan antara pohon sintaks abstrak dengan pohon sintaks konkrit terletak pada tidak tercantumnya detil sintaks seperti tanda koma, titik koma, dan sebagainya, sehingga AST tidak bersifat sensitif pada bahasa pemrograman.



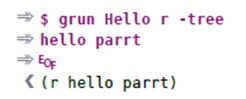
Gambar 1. Contoh AST dari potongan kode program

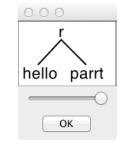
AST untuk kode program yang ditulis dalam bahasa pemrograman yang berbeda bisa jadi sama apabila kode program ditulis dengan algoritma dan langkah-langkah yang sama. AST dapat menunjukkan kesamaan logika antara satu kode program dengan kode program lainnya, sebab dalam AST detil yang kurang penting seperti komentar, tanda baca, dan lainnya yang tidak terkait dengan operasi logika pada kode tidak dicantumkan.

## ANTLR

ANTLR merupakan pembangkit *parser* yang dapat digunakan untuk membaca, memroses, menjalankan, atau menerjemahkan suatu teks terstruktur atau arsip biner [8]. Sebuah definisi tata bahasa diperlukan agar ANTLR dapat secara otomatis membuat *parser* yang dapat digunakan untuk mengenali bahasa tersebut. Dengan adanya *parser* itu pula ANTLR dapat menghasilkan *parse tree* yang menunjukkan struktur dari dokumen yang telah dibaca dengan *parser*.

ANTLR tidak hanya dapat digunakan untuk mengenali bahasa. Dengan tata bahasa yang tepat, ANTLR dapat mengasilkan *parser* yang dapat mengenali logika yang terkandung dalam kode program. Tanpa memperhatikan detil sintaks yang tidak perlu, ANTLR dapat digunakan untuk menghasilkan AST (*Abstract Syntax Tree*).





Gambar 2. Contoh perintah ANTLR pada layar baris perintah dan *parse tree* yang dihasilkan

Pada Gambar 2 ditunjukkan contoh perintah ANTLR pada layar baris perintah. Pada contoh ini dianggap ada sebuah tata bahasa bernama *Hello* yang digunakan untuk proses *parsing* kalimat “*hello part*”. Hasilnya diperlihatkan dalam bentuk pohon pada GUI baru. Bentuk pohon yang ditampilkan ini adalah *parse tree* dari kalimat tadi.

## LCS (*Longest Common Subsequence*)

Dalam berbagai persoalan di dunia nyata, terkadang dibutuhkan suatu ukuran untuk menyatakan kesamaan dari dua benda. Sebagai contoh, terdapat rangkaian DNA *S1* dan *S2* yang harus diukur kemiripannya. Cara mengukur kemiripan bermacam-macam. Bisa saja *S1* dikatakan mirip *S2* jika *S1* merupakan *substring* dari *S2*, maupun sebaliknya. Atau kedua rangkaian DNA dikatakan mirip jika jumlah perubahan yang dibutuhkan untuk mengubah *S1* menjadi *S2* sangat kecil. Cara lain untuk mengukur kemiripan kedua rangkaian ini adalah dengan mencari rangkaian DNA baru, *S3*, dimana basa DNA pada *S3* muncul di rangkaian *S1* dan *S2* dengan urutan yang sama meskipun tidak secara berurutan. Semakin panjang *S3* yang didapatkan, maka semakin mirip pula *S1* dan *S2*. Cara ini dikenal sebagai *Longest Common Subsequence* atau LCS [2].

LCS merupakan sub-rangkaian tertentu yang sama-sama terdapat pada rangkaian *X* dan *Y*. Sebuah rangkaian *Z* merupakan sub-rangkaian dari *X* apabila elemen pada *Z* terdapat pada *X* dengan urutan yang sama, meskipun tidak bersebelahan. Misalnya, *Z*={*B,C,D,B*} merupakan sub-rangkaian dari *X*={*A,B,C,B,D,A,B*} pada indeks {2,3,5,7}. Sebuah rangkaian *Z* merupakan LCS dari rangkaian *X* dan *Y* apabila *Z* adalah sub-rangkaian untuk *X* sekaligus *Y*.

LCS dapat digunakan untuk menentukan kemiripan dalam berbagai kasus di dunia nyata. Pengukuran kemiripan menggunakan LCS memungkinkan dideteksinya kemiripan meskipun terdapat sisipan dan/atau pengurangan di dalam rangkaian yang diuji kemiripannya.

## *Web Service*

*Web service* merupakan cara komunikasi suatu aplikasi dengan aplikasi lain melalui jaringan (*World Wide Web*). Aplikasi dapat mengakses *web service* melalui protokol web dan format data yang umum, seperti HTTP, XML, atau SOAP, sehingga aplikasi tidak harus mempedulikan bagaimana *web service* ini dibangun [6].

## *Moodle*

*Moodle* adalah sebuah sistem manajemen belajar-mengajar (*Learning Management System*) atau LMS yang dibangun secara *online*. *Moodle* merupakan akronim dari *Modular Object Oriented Dynamic Learning Environment*. *Moodle* diciptakan untuk membantu proses belajar mengajar di mana diharapkan terjadi interaksi yang lebih banyak antara pengajar dan siswa di mana saja dan kapan saja.

Dalam penggunaan *Moodle* sebagai sarana pembantu proses belajar-mengajar, *Moodle* dapat dianggap sebagai sebuah *platform* dengan banyak kakas yang dapat disusun di atasnya [7]. Kakas-kakas ini tersedia sebagai layanan di *Moodle* dan dapat dipadukan antara satu layanan dengan layanan lain untuk menciptakan lingkungan kelas virtual yang diharapkan. Layanan ini bisa berupa pengumpulan tugas, penilaian tugas, komunikasi, dan sebagainya. Dengan adanya *Moodle* proses belajar-mengajar diharapkan dapat berjalan dengan lancar di dunia maya dan dapat diakses tanpa terbatas tempat dan waktu.

# RINGKASAN ISI TUGAS AKHIR

Topik Tugas Akhir yang akan dikerjakan berkaitan dengan pendeteksian plagiasi kode sumber pada tugas pemrograman dengan membandingkan AST kode sumber dan menghitung kesamaannya menggunakan metode LCS. Kakas pendeteksi ini akan dibangun sebagai *web service* yang kemudian dapat diakses dan digunakan sebagai *plugin* pada kelas virtual *Moodle*.



Gambar 3. Diagram alir

Pada Gambar 3 ditunjukkan proses pengolahan kode sumber hingga pelaporan potongan program yang diduga plagiat beserta persentasenya. Proses pendeteksian ini terdiri dari dua tahap utama yakni *parsing* kode sumber menjadi bentuk AST kemudian pengukuran kemiripan potongan AST antar kode sumber dengan metode LCS. Metode pendeteksi plagiasi yang digunakan dalam pengerjaan tugas akhir ini sedikit mirip dengan metode pada pendeteksian klon yang biasa digunakan pada evolusi perangkat lunak. Perbedaannya terletak pada kode sumber yang diuji, jika pendeteksian klon dilakukan dalam satu kode sumber, pendeteksian plagiasi dilakukan pada kode sumber yang berbeda. Tujuannya sama-sama untuk menemukan potongan kode sumber yang memiliki kemiripan implementasi.

Tahap pertama, yaitu *parsing* kode sumber merupakan tahap mengubah kode sumber menjadi bentuk AST. *Parser* yang digunakan berbeda untuk masing-masing bahasa pemrograman, sehingga bahasa yang digunakan perlu didefinisikan terlebih dahulu. AST akan menggambarkan struktur program tanpa memperhatikan detil sintaks seperti tanda koma, titik koma, dan sebagainya. Sehingga, AST yang dihasilkan tidak sensitif terhadap bahasa pemrograman kode sumber. Hal ini memungkinkan pendeteksian plagiasi pada tugas pemrograman, meskipun bahasa pemrograman yang digunakan berbeda.

Tahap berikutnya adalah membandingkan AST dari setiap kode program dengan LCS untuk menemukan kemiripannya. AST akan dipecah berdasarkan fungsi dan dibandingkan antar satu kode sumber dengan kode sumber lainnya. Pengukuran kemiripan dengan metode LCS memungkinkan terdeteksinya kemiripan meskipun ada penyisipan dan/atau pengurangan kode pada fungsi tersebut. AST yang dianggap mirip kemudian dilaporkan sebagai kode yang dicurigai sebagai plagiat dan ditampilkan dalam sebuah laporan, sehingga pengajar dapat melakukan tindakan lebih lanjut. Isi dari laporan yang diberikan adalah potongan kode sumber yang diduga plagiat beserta persentase kemiripannya.

Metode pendeteksian plagiasi dengan AST memiliki keunggulan dibanding metode *token* pada umumnya, yakni kemampuan mendeteksi plagiasi meskipun urutan deklarasi fungsi dibalik. Karena AST akan dibagi berdasarkan fungsi, dan tiap fungsi tersebut akan dibandingkan untuk menemukan kemiripan. Dengan demikian, urutan deklarasi tidak akan mempengaruhi pendeteksian plagiasi.

Dalam pengerjaan tugas pemrograman, umumnya terdapat algoritma yang telah diajarkan di kelas sebelum penugasan program. Dalam kasus ini, kemungkinan besar siswa di kelas akan mengerjakan tugas program sesuai *pseudocode* yang telah diajarkan. Hal ini akan menyebabkan dideteksinya bagian program dari *pseudocode* tersebut sebagai bagian program yang dicurigai sebagai plagiat. Untuk menanggulangi hal ini, dapat ditambahkan fitur untuk membuat pengecualian dalam pendeteksian plagiasi. Pengajar dapat memasukkan potongan program sesuai *pseudocode* yang telah diajarkan. Kemudian, ketika pengukuran kesamaan dengan LCS, apabila potongan program ditemukan mirip dengan bagian program yang dijadikan pengecualian, maka kesamaannya dengan yang lain tidak akan diukur. Dengan demikian, bagian program yang sesuai dengan algoritma yang telah diajarkan tidak akan dicurigai sebagai plagiat.



Gambar 4. Diagram kasus penggunaan

Gambar 4 menunjukkan diagram kasus penggunaan pada kakas yang akan dibangun. Penjelasan fitur berdasarkan diagram di atas adalah sebagai berikut.

1. Memasang *plugin* deteksi plagiasi ke *Moodle*

Merupakan fitur untuk memasang *plugin* deteksi plagiasi kode sumber ini ke dalam kelas virtual *Moodle*.

1. Memperbaharui *plugin* deteksi plagiasi

Merupakan fitur untuk memperbaharui *plugin* ketika layanan yang disediakan dalam *plugin* mengalami perubahan.

1. Menguji plagiasi kode sumber

Merupakan fitur untuk menguji potongan kode yang dicurigai sebagai plagiasi dari sekumpulan kode sumber tugas pemrograman.

1. Menampilkan laporan hasil deteksi plagiasi

Merupakan fitur untuk menampilkan bagian kode sumber yang dicurigai sebagai plagiat. Hasil yang ditampilkan berupa kumpulan potongan program yang mirip beserta letak dan persentase kemiripannya.

# METODOLOGI

## Penyusunan proposal Tugas Akhir

Pada proposal Tugas Akhir ini dibahas mengenai rencana pengembangan kakas pendeteksi plagiasi pada tugas pemrograman khususnya pada kelas virtual *Moodle*. Perangkat lunak ini akan membantu para penguji, pengajar, maupun asisten pengajar untuk memudahkan menemukan bagian program yang diduga plagiat serta seberapa besar kemiripannya. Dari kumpulan kode sumber yang diuji akan dibuat bentuk AST yang kemudian dipecah berdasarkan fungsi. Setiap AST ini kemudian diukur kemiripannya dengan LCS untuk menemukan kode yang mirip dan dicurigai sebagai plagiat.

## Studi literatur

Studi literatur yang digunakan untuk menunjang pengerjaan Tugas Akhir ini adalah sebagai berikut.

1. Cara membangun AST dari kode program berbahasa C++, C#, dan Java dengan menggunakan ANTLR.
2. Cara membagi AST berdasarkan fungsi atau *method*.
3. Cara melakukan pengukuran kemiripan dengan LCS pada AST tiap fungsi yang ada pada kode sumber.
4. Cara membangun *plugin* pada kelas virtual *Moodle* dari *web service* Java.

## Analisis dan desain perangkat lunak

Aktor dari sistem ini adalah pengajar yang memberikan tugas pemrograman pada kelas virtual *Moodle* dan perlu mendeteksi terjadinya plagiasi pada pengerjaan tugas tersebut. Sistem ini memiliki kebutuhan fungsional sebagai berikut.

1. Memasukkan sejumlah kode sumber yang ingin diuji.
2. Menambahkan potongan program sebagai pengecualian.
3. Melihat hasil deteksi plagiasi kode sumber.

## Implementasi perangkat lunak

Pembuatan kakas pendeteksi plagiasi pada tugas pemrograman pada kelas virtual *Moodle* ini membutuhkan beberapa kakas bantu sebagai berikut.

1. IDE *Netbeans* 7.0.1
2. *Java Development Kit* 7
3. *Library* ANTLR 4.2
4. *StarUML* 5.0 untuk perancangan dan pemodelan

## Pengujian dan evaluasi

Pengujian akan dilakukan dengan metode *blackbox*. Sistem yang dikembangkan akan diuji dengan beberapa data uji buatan berupa kode sumber dari tugas pemrograman yang dimodifikasi sebagai kontrol deteksi kemiripan kode. Kemudian sistem akan diuji dengan kode sumber dari beberapa tugas pemrograman mahasiswa Teknik Informatika ITS. Hasil deteksi dari kakas kemudian dibandingkan dengan pemeriksaan manual terhadap kode sumber tugas pemrograman tersebut.

## Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat. Sistematika penulisan buku tugas akhir secara garis besar antara lain:

1. Pendahuluan
   1. Latar Belakang
   2. Rumusan Masalah
   3. Batasan Tugas Akhir
   4. Tujuan
   5. Metodologi
   6. Sistematika Penulisan
2. Tinjauan Pustaka
3. Desain dan Implementasi
4. Pengujian dan Evaluasi
5. Kesimpulan dan Saran
6. Daftar Pustaka

# JADWAL KEGIATAN

Pengerjaan Tugas Akhir ini akan dilakukan sesuai jadwal yang tertera pada Tabel 1.

Tabel 1. Jadwal pengerjaan Tugas Akhir

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tahapan | Tahun 2014 | | | | | | | | | | | | | | | | | | | | |
| Februari | | | | Maret | | | | April | | | | Mei | | | | | Juni | | | |
| Penyusunan Proposal |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Studi Literatur |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Perancangan system |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Implementasi |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Pengujian dan evaluasi |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Penyusunan buku |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

# DAFTAR PUSTAKA

1. Clough, Paul. "Plagiarism in natural and programming languages: an overview of current tools and technology." *Research Memoranda: CS-00-05, Department of Computer Science, University of Sheffield, UK*, 2000.
2. Cormen, Thomas H, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to Algorithm, Second Edition.* Massachusets: The MIT Press, 2001.
3. Hage, Jurriaan, Peter Rademaker, and Nike van Vugt. *A comparison of plagiarism detection tools.* Utrecht: Department of Information and Computing Technology, Uthrect University, 2010.
4. Jones, Joel. "Abstract Syntax Tree Implementation Idioms." *Proceedings of the 10th Conference on Pattern Languages of Programs*, 2003.
5. Li, Xiao, and Xiao Jing Zhong. "The Source Code Plagiarism Detection using AST." *International Symposium on Intelligence Information Processing and Trusted Computing*, 2010: pp.406-408.
6. Microsoft. *Web Services.* 2014. http://msdn.microsoft.com/en-us/library/ms950421.aspx (diakses Maret 5, 2014).
7. Moodle. *About Moodle - MoodleDocs.* Maret 4, 2014. http://docs.moodle.org/26/en/About\_Moodle (diakses Maret 5, 2014).
8. Parr, Terence. *The Definitive ANTLR 4 Reference.* Texas: The Pragmantic Bookshelf, 2012.
9. Poongodi, D, and G Tholkkappia Arasu. "An automatic method or statement level plagiarism detection in source code using abstract syntax tree." *International Journal of Advanced Research in Computer and Communication Engineering*, 2013: pp.1932-1938.