DETEKSI PLAGIARISME PADA MATA KULIAH PEMROGRAMAN BERORIENTASI OBJEK

PLAGIARISM DETECTION ON OBJECT ORIENTED PROGRAMMING COURSE

Jefry Kurniawan[1], Sari Dewi Budiwati, S.T., M.T.[2], Dahliar Ananda, S.T,M.T.[3]

[1,2,3]Prodi D3 Manajemen Informatika, Fakultas Ilmu Terapan, Universitas Telkom
Jefrykurniaone@gmail.com

Abstrak

Plagiarisme merupakan tindakan penipuan hasil karya orang lain tanpa sepengetahuan dari penulis aslinya. Pada perkuliah Pemrograman Berorientasi Objek (PBO) beberapa kali ditemukan mahasiswa yang melakukan plagiarisme, terutama saat pelaksanaan assessment dan tugas. Akibatnya kemampuan mahasiswa dalam memahami PBO tidak tercapai.

Oleh karena itu, diusulkan solusi dengan Proyek Akhir ini untuk mempermudah pemeriksaan plagiat terhadap Assessment dan tugas mahasiswa dengan menggunakan Aplikasi Deteksi Plagiarisme yang memiliki 2 fitur utama yaitu pengecekkan plagiarisme dan penilaian kode program. Selain dapat memeriksa kesamaan kode program dan menilai kode program aplikasi ini juga dapat menyimpan hasil dari pemeriksaan kode program lalu menampilkannya kembali. Aplikasi ini dibuat dengan menggunakan metode prototipe dan bahasa pemrograman Java.

Dari hasil pengujian white box aplikasi ini sudah dapat melakukan deteksi kemiripan kode program. Namun, masih memiliki kekuranganan yaitu keterbatasan file yang dimasukkan yang masih berjumlah 2 file saja. Lalu untuk fitur penilaian kode program juga sudah berhasil mengidentifikasi bagian-bagian kode program seperti nama kelas, variabel, prosedur dan fungsi. Namun, fitur ini juga masih memiliki kekurangan yaitu tidak bisa mendeteksi jika ada 2 variabel dalam 1 baris dan hanya bisa mendeteksi variabel-variabel primitif.

Kata Kunci: ***Plagiarisme, PBO, Assessment, Tugas, Mahasiswa, White Box***

Abstract

Plagiarism is an act of fraud by others without the knowledge of the original author. In Object Oriented Programming (OOP) several times found students who do plagiarism, especially during the assessment and task implementation. As a result, the students' ability to understand PBO is not achieved.

Therefore, the authors offer a solution with this Final Project to facilitate plagiarism examination of Assessment and student assignment by using Application Plagiarism Detection which has 2 main features that is checking plagiarism and assessment of source code. In addition to checking the similarity of the source code and rate the source code of this application can also save the results of the source code checks and display them again. This application is created using prototype method and Java programming language.

From the results of white box testing this application is able to detect the resemblance of source code. However, it still has a deficiency of the limitations of the files entered which still amounted to 2 files only. Then for the assessment feature source code has also been successfully identified the parts of the source code such as class names, variables, procedures and functions. However, this feature also still has the disadvantage of not being able to detect if there are 2 variables in 1 line and can only detect primitive variables.

Keywords : Plagiarism, OOP, Asessment, Assignment, Student, White Box

1. Pendahuluan

Dalam ranah ilmu informatika mendeteksi kemiripan kode program sangat dibutuhkan. Salah satu fungsi aplikasi ini adalah untuk mendeteksi terjadinya praktik plagiarisme dalam lingkungan akademis pemrograman. Selain itu, dengan sedikit modifikasi, teknik ini dapat pula diterapkan untuk mendeteksi kemiripan antar bagian program untuk memudahkan proses modularisasi program yang sudah ada.

Pada perkuliahan Pemrograman Berorientasi Objek (PBO) beberapa kali ditemukan mahasiswa yang melakukan plagiarisme, terutama saat pelaksanaan assessment dan tugas. Namun pada saat pemeriksaan tidak

bisa diketahui dengan pasti siapa yang menjadi sumber plagiat dan siapa yang memplagiat source codenya. Oleh karena itu diperlukan sistem untuk melakukan pendeteksian plagiarisme pada mata kuliah PBO.

Plagiarisme merupakan tindakan penipuan hasil karya orang lain tanpa sepengetahuan dari penulis aslinya, yang melanggar suatu Hak Cipta dan Hak Moral. Ketertarikan mahasiswa terhadap tindakan plagiarisme, dibangun oleh rasionalitas instrumental. Mahasiswa lebih memperhitungkan tentang efisiensi, efektifitas dan nilai yang dimiliki oleh sumber dayanya (tugas akademik) untuk mencapai tujuan yang diharapkan. Ketika aktor (mahasiswa) menentukan tujuan, aktor akan dihadapkan pada sebuah pilihan cara alternatif yaitu cara SKS (Sistem Kebut Semalam) dan SKJ (Sekali Kerja Jadi). Pilihan tersebut akan memunculkan suatu bentuk tindakan plagiarisme dan konsekuensi dari tindakan plagiarisme.[6]

Pada proyek akhir ini, akan dibuatkan aplikasi untuk mendeteksi kemiripan kode pada mata kuliah PBO. Kode yang akan dilakukan pengecekkan kemiripannya adalah kode hasil assessment dan tugas mata kuliah PBO. Pendeteksian kemiripan kode pada mata kuliah PBO ini dilakukan agar memudahkan para dosen untuk mengoreksi jawaban dari mahasiswa dan agar memudahkan proses penilaian dari assessment dan tugas.

Untuk mendeteksi kemiripan kode pada mata kuliah Pemrograman Berbasis Objek, digunakan metode pengecekkan kode menggunakan Abstract Syntax Tree (AST). Abstract Syntax Tree (AST) merupakan representasi pohon struktur sintaksis abstrak kode yang ditulis dalam bahasa pemrograman. Setiap node dari pohon menunjukkan suatu konstruksi yang terjadi dalam kode.

2. Dasar Teori

Plagiarisme

Plagiarisme adalah bentuk penyalahgunaan hak kekayaan intelektual milik orang lain, yang mana karya tersebut dipresentasikan dan diakui secara tidak sah sebagai hasil karya pribadi [1]. Pagiat dapat dianggap sebagai tindak pidana karena mencuri hak cipta orang lain. Di dunia pendidikan, pelaku plagiarisme dapat mendapat hukuman berat seperti dikeluarkan dari sekolah/universitas.

Pelaku plagiat disebut sebagai plagiator. Singkat kata, plagiat adalah pencurian karangan milik orang lain. Dapat juga diartikan sebagai pengambilan karangan (pendapat dan sebagainya) orang lain yang kemudian dijadikan seolah-olah miliknya sendiri. Setiap karangan yang asli dianggap sebagai hak milik si pengarang dan tidak boleh dicetak ulang tanpa izin yang mempunyai hak atau penerbit karangan tersebut.

Jenis – jenis plagiarisme menurut [2] :

a. Plagiarisme ide : Mengambil ide yang sudah ada tanpa menyebut sumber dengan jelas.

b. Plagiarisme isi (data penelitian): Mengambil data penelitian orang lain.

c. Plagiarisme kata, kalimat, paragraph

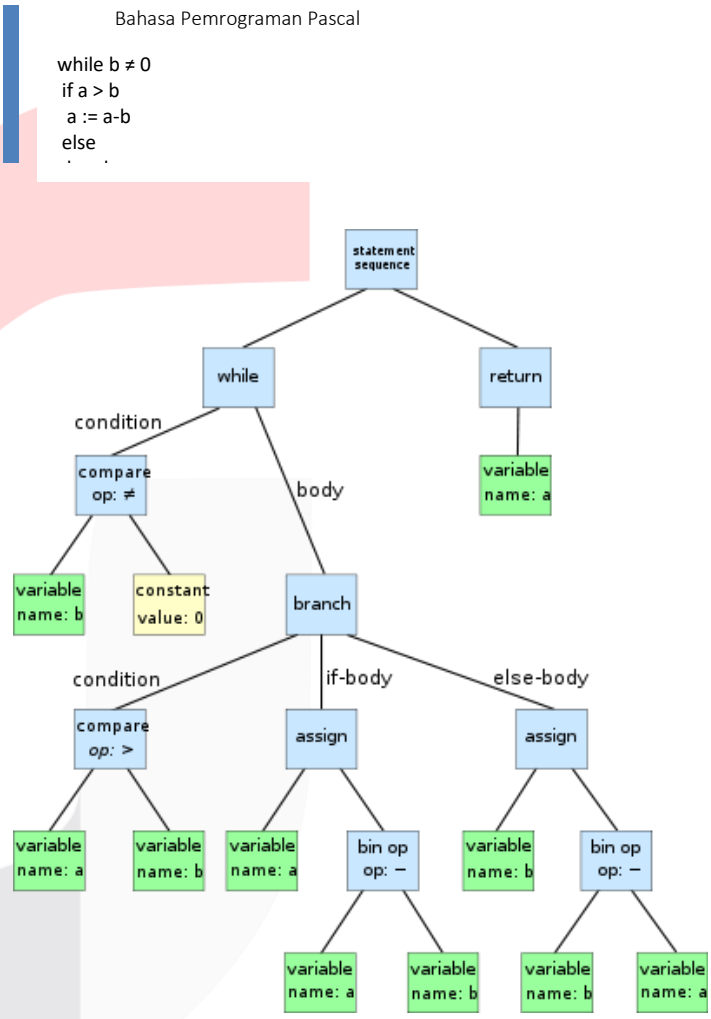
d. Plagiarisme total

Aplikasi Desktop

Aplikasi desktop merupakan aplikasi yang dapat berjalan sendiri tanpa memerlukan adanya koneksi internet. Ditinjau dari asal katanya yaitu desk yang berarti meja dan top yang berarti di atas, dapat diartikan secara harfiah menjadi suatu komputer yang ditempatkan di atas meja dan digunakan untuk berbagai keperluan, baik itu keperluan pribadi maupun pekerjaan. Desktop merupakan komputer yang tidak diperuntukkan untuk dibawa kemana-mana ataupun dijinjing. Sebuah komputer desktop digunakan untuk mengakses program tertentu. Program yang diakses pada komputer desktop inilah yang seringkali disebut dengan aplikasi desktop [3].

Abstract Sysntax Tree

Abstract syntax tree(AST) merupakan representasi sebuah kode program kedalam pohon node. Setiap node menunjukan algoritma dari kode program. Satu simpul (node)/vertex disebut akar (root) yang memiliki lintasan ke setiap simpul. Pohon sintaks/pohon penurunan (syntax tree/derivation tree/ parse tree) berguna untuk menggambarkan bagaimana memperoleh suatu string dengan cara mengelompokkan simbol-simbol variabel atau sebuah kondisi if - then - else.[8] Contoh dapat dilihat pada gambar 2.2, sebuah abstract syntax tree untuk kode program :



Gambar 2-1 Abstract Syntax Tree

Framework

Framework adalah sebuah kerangka kerja dalam aplikasi yang didalamnya memiliki suatu potongan-potongan program yang disusun (modul), sehingga programer tidak perlu membuangun program dari nol, karena framework telah menyediakannya. Dengan adanya framework, pekerjaan kita akan lebih terorganisir. Sehingga dalam pencarian kesalahan dalam pembuatan program akan lebih mudah dideteksi [6].

Framework juga bisa diartikan sebagai komponen pemrograman yang siap di re-use kapan saja, sehingga programer tidak harus membuat skrip yang sama untuk tugas yang sama. Berikut adalah beberapa manfaat framework:

1. Dapat membantu kerja developer dalam membangun aplikasi sehingga aplikasi bisa selesai dalam waktu yang singkat
2. Penerapan design patterns memudahkan dalam rancangan, pengembangan dan pemeliharaan sistem
3. Stability dan reliability aplikasi yang kita bangun lebih stabil dan handal karena berbasis pada framework yang sudah teruji stabilitas dan kehandalannya.
4. Coding style konsisten, memudahkan dalam membaca kode dan dalam menemukan bugs.
5. Dokumentasi, framework dapat mendisiplinkan kita untuk menulis dokumentasi atas apa yang kita kerjakan.

UML

UML adalah singkatan dari Unified Modeling Languange yang merupakan standarisasi bahasa pemodelan untuk membangun perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek. UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan mendokumentasikan dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung. UML sendiri teridiri atas pengelompokan diagram-diagram sistem menurut aspek atau sudut pandang tertentu. UML mempunya 9 diagram, yaitu; use-case, class, objek, state, sequence, collaboration, activity, component, dan deployment diagram [7].

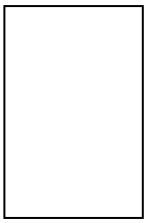
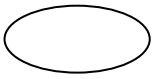
UML diaplikasikan untuk maksud tertentu, antara lain untuk:

1. Merancang perangkat lunak
2. Sarana komunikasi antara perangkat lunak dengan proses bisnis
3. Menjabarkan sistem secara rinci untuk analisa dan mencari apa yang diperlukan sistem
4. Mendokumentasi sistem yang ada, proses-proses dan organisasinya

Use Case Diagram

Use Case Diagram merupakan pemodelan untuk menggambarkan kelakuan (behaviour) sistem yang akan dibuat. Diagram use case mendeklarasikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Diagram use case digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa-siapa saja yang berhak menggunakan fungsi-fungsi tersebut [7]. Berikut adalah simbol-simbol yang ada pada use case diagram :

Tabel 2-1 Use Case Diagram

NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case.*
2	- - - - - - - ->	*Include*	Menspesifikasikan bahwa *use case* sumber secara *eksplisit.*
3	————	*Assosiation*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
4		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
5		*Use case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.

Java Code Clone Detection (JCCD)

Java Code Clone Detection API (JCCD) memungkinkan untuk mengimplementasikan detektor kloning kode individual. Detektor kloning kode ini dapat dengan mudah diintegrasikan ke aplikasi Java lainnya. JCCD didasarkan pada pipeline architecture yang memungkinkan untuk mengganti seluruh bagian proses pendeteksian tanpa mengubah keseluruhan pipeline. Hal ini memungkinkan untuk menerapkan kembali teknik mutakhir serta untuk mewujudkan gagasan dan konsep baru.

JCCD API memiliki beberapa operator preprosesing atau komparator dalam fase pipeline yang sesuai. Karena kode mungkin bisa memberikan hasil yang sama. Satu-satunya yang membedakan adalah cara kode berjalan. Sebagai contoh, dalam pipeline berbasis AST anda bisa menggunakan preprosesing untuk menggeneralisasi kode dengan menghapus node. Tapi anda juga bisa menggunakan komparator yang menganggap semua deklarasi metod adalah identik dan mengabaikan nama.

Operator disini dapat dibagi menjadi dua kelompok yaitu, independen dan dependen. Operator independen adalah operator yang bekerja sendiri. Operator yang dependen adalah operator yang membutuhkan operator lain untuk bekerja.[10]

Java

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di Sun Microsystems saat ini merupakan bagian dari Oracle dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin aras bawah yang minimal. Aplikasi-aplikasi berbasis java umumnya dikompilasi ke dalam p-code (bytecode) dan dapat dijalankan pada berbagai Mesin Virtual Java (JVM). Java merupakan bahasa pemrograman yang bersifat umum/non-spesifik (general purpose), dan secara khusus didisain untuk memanfaatkan dependensi implementasi seminimal mungkin. Karena fungsionalitasnya yang memungkinkan aplikasi java mampu berjalan di beberapa platform sistem operasi yang berbeda, java dikenal pula dengan slogannya, "Tulis sekali, jalankan di mana pun". Saat ini java merupakan bahasa pemrograman yang paling populer digunakan, dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi.[9]

Berikut adalah contoh program HalloWorld yang ditulis dengan menggunakan bahasa pemrograman java :

```
// Outputs "Hello, world!" and then exits
public class HelloWorld {
   public static void main(String args[]) {
       System.out.println("Hello, world!");
   }
}
```

Gambar 2-2 Program HalloWorld

Pemrograman Berorientasi Objek

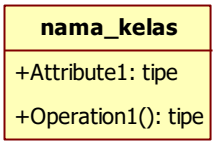
Pemrograman Berorientasi Objek (PBO) merupakan paradigma pemrograman yang berorientasikan pada objek. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek. Bandingkan dengan logika pemrograman terstruktur. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya.

Model data berorientasi objek dikatakan dapat meberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik peranti lunak skala besar. Lebih jauh lagi, pendukung PBO mengklaim bahwa PBO lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan PBO lebih mudah dikembangkan dan dirawat.

Class Diagram

Diagram kelas atau class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

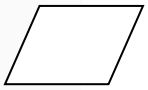
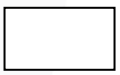
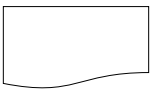
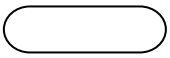
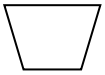
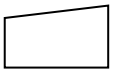
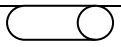
Tabel 2-2 Class Diagram

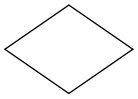
No	Simbol	Deskripsi
1	Kelas 	Kelas pada struktur sistem
2	Asosiasi	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
3	Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)
4	Agregasi 	Relasi antar kelas dengan makna semua bagian (*whole part*)
5	Asosiasi berarah 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*

Flowmap

Flowmap merupakan diagram yang menggambarkan aliran dokumen pada suatu prosedur kerja di organisasi dan memperlihatkan diagram alir yang menunjukan arus dari dokumen, aliran data fisik, entitas-entitas *system* informasi. Penggambaran biasanya diawali dengan mengamati dokumen apa yang menjadi media data atau informasi. Selanjutnya ditelusuri bagaimana dokumen tersebut terbentuk, kebagian atau entitas mana dokumen tersebut mengalir, perubahan apa yang terjadi pada dokumen tersebut, proses apa yang terjadi terhadap dokumen tersebut dan seterusnya [16]. Berikut simbol-simbol yang ada pada *flowmap* pada Tabel 2-1 :

Tabel Error! No text of specified style in document.**-3 Simbol-Simbol Flowmap**

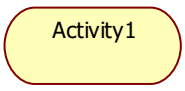
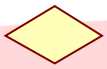
Nama	Simbol	Arti
Input/Output		Merepresentasikan *Input* data atau *Output* data yang diproses atau Informasi.
Proses		Mempresentasikan operasi.
Anak Panah		Mempresentasikan alur kerja.
Dokumen		Menggambarkan *input* dan *output* berupa dokumen
Terminator		Awal/Akhir *flowchart.*
Manual Operation		Operasi manual.
Manual Input		Input yang dimasukan secara manual pada *keyboard.*
Akses penyimpanan		Tempat penyimpanan

Nama	Simbol	Arti
langsung		data.
Display		*Output* yang ditampilkan pada terminal.
Decision		Menunjukan pilihan yang akan dikerjakan atau keputusan yang harus dibuat dalam proses pengolahan data.

Activity Diagram

Activity Diagram adalah diagram yang menggambarkan worlflow (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan oleh aktor. Komponen yang ada pada activity diagram antara lain:

Tabel 2-4 Activity Diagram

No	Simbol	Deskripsi
1	*Swimlines* 	Memecah *activity diagram* menjadi baris dan kolom untuk membagi tanggung jawab obyek-obyek yang melakukan aktivitas
2	Aktivitas 	Menunjukkan aktivitas yang diakukan
3	Inisialisasi	Menunjukkan awal aktivitas dimulai
		
4	Transisi	Menunjukkan aktivitas selanjutnya setelah aktivitas ebelumnya
5	*Decission* 	Digunakan untuk menggambarkan tes kondisi untuk memastikan bahwa kontrol *flow* atau objek *flow* mengalir lebih ke satu jalur
6	Final 	Menunjukkan bagian akhir dari aktivitas

White Box

White box testing adalah pengujian yang didasarkan pada pengecekan terhadap detail perancangan, menggunakan struktur kontrol dari desain program secara prosedural untuk membagi pengujian ke dalam beberapa kasus pengujian. Secara sekilas dapat diambil kesimpulan white box testing merupakan petunjuk untuk mendapatkan program yang benar secara 100%.

Pengujian dilakukan berdasarkan bagaimana suatu software menghasilkan output dari input. Pengujian ini dilakukan berdasarkan kode program. White box testing memiliki kebelihan dan kekurangan sebagai berikut:

Kelebihan white box testing:

1. White box testing akan mendeteksi kondisi-kondisi yang tidak sesuai dan mendeteksi kapan proses pengulangan akan berhenti

2. Menampilkan asumsi yang tidak sesuai kenyataan, untuk dianalisa dan diperbaiki

3. Mendeteksi bahasa pemrograman yang bersifat case sensitive

Kelemahan white box testing:

1. Untuk perangkat lunak yang tergolong besar, white box testing dianggap sebagai strategi yang terlalu boros. Karena, akan melibatkan sumberdaya yang besar untuk melakukannya

3. Analisis dan Perancangan

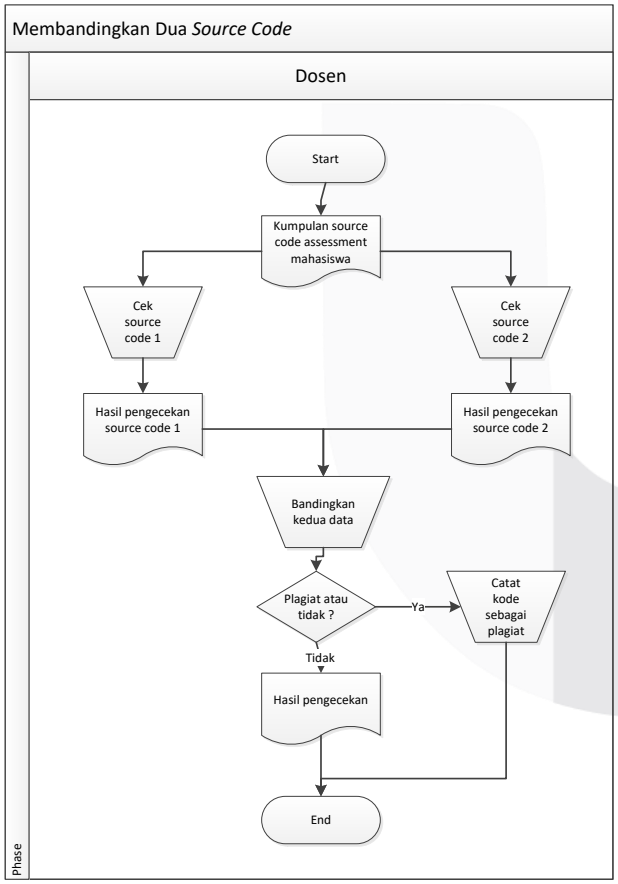
3.Gambaran Sistem Saat ini

Saat ini semua proses pengecekkan kode program assessment dan tugas semuanya dilakukan dengan cara manual, yaitu dibandingkan satu per satu antara kode program yang pertama dengan kode program yang lainnya. Namun pada praktiknya proses ini cukup memakan waktu karena dosen harus membuka semua file dan membandingkannya. Beberapa mengalami kesulitan dalam pemeriksaan kode program dikarenakan banyaknya file yang harus dibandingkan dan memberikan penilaian pada kode program. Oleh karena itu proses pemeriksaan menjadi tidak efektif. Berikut adalah flowmap dari sistem membandingkan dan penilaian dari kode program.

Flowmap

Gambaran Alur Sistem Berjalan Membandingkan Dua Source Code

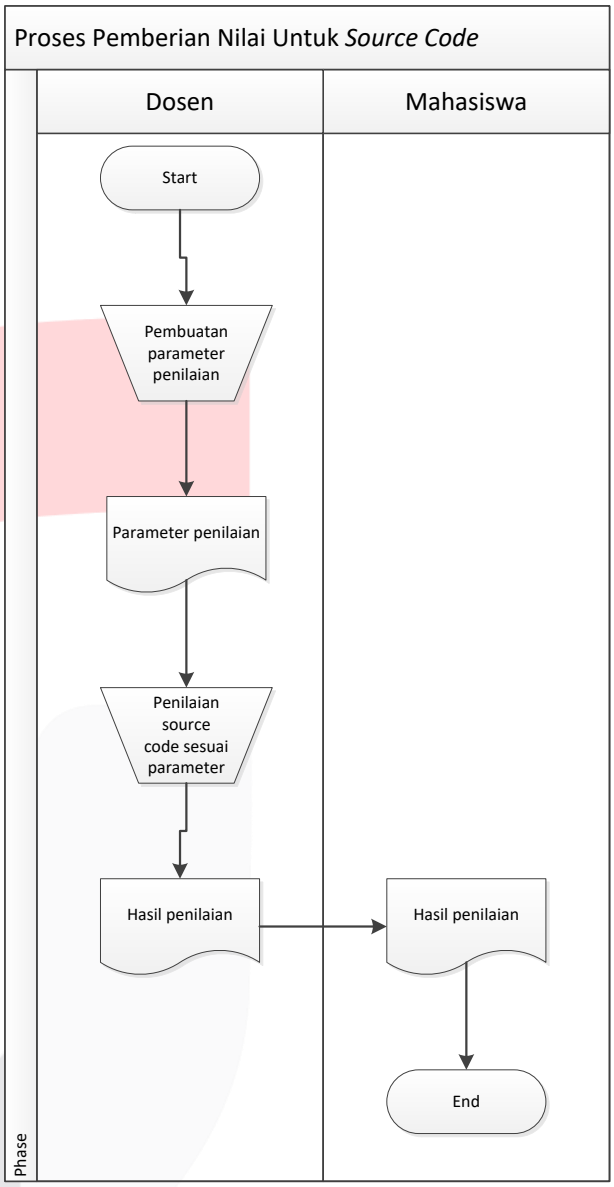
Pada flowmap dibawah menjelaskan proses perbandingan kode program masih manual. Dosen harus membuka kode program pertama lalu dibandingkan dengan kode program kedua. Kemudian dosen harus melihat bagian bagian dari kode program untuk memerika apakah kedua kode program tersebut plagiat atau tidak. Jika plagiat, maka kode program tersebut akan dicatat sebagai plagiat dan akan ditindak lanjuti lebih lanjut. Jika tidak maka hasil perbandingan akan keluat dan akan dilanjutkan ke proses penilaian.



Gambar 3-1 Flowmap Membandingkan Dua Source Code

Gambaran Alur Sistem Berjalan Penilaian Source Code

Pada flowmap diatas menjelaskan proses penilaian untuk kode program juga masih berjalan secara manual. Proses diatas dimulai dari dosen membuat parameter untuk penilaian assessment dan tugas. Setelah parameter selesai dibuat, selanjutnya dosen membuka satu per satu file kode program dan memberikan nilai dari kode program tersebut berdasarkan parameter yang telah dibuat. Jika sudah diberikan nilai maka hasil penilaian akan dibagikan ke mahasiswa dan dosen juga menyimpan hasil penilaian tersebut.



Gambar 3-2 Flowmap Penilaian Source Code

Sistem Usulan

Setelah menganalisa sistem yang ada maka dibutuhkan sebuah aplikasi deteksi plagiarisme yang bisa mempermudah proses pemeriksaan dan penilaian kode program assessment dan tugas. Berikut adalah rancangan analisis sistem yang diusulkan.

Kebutuhan Perangkat Keras dan Perangkat Lunak

Pengembangan Sistem

Untuk membangun pada aplikasi Deteksi Plagiarisme dibutuhkan perangkat keras dan perangkat lunak yang mendukung dengan spesifikasi sebagai berikut.

a. Kebutuhan Perangkat Keras
   Spesifikasi perangkat keras yang menjadi syarat minimal untuk dapat membangun aplikasi secara normal adalah sebagai berikut

Tabel 3-1 Perangkat Keras

No	Jenis Hardware	Keterangan
1	Prosesor	Intel Core i3
2	RAM	4 GB
3	Hard Disk	500 GB
4	VGA	Intel HD Grapich 3000

b. Kebutuhan Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan untuk menjalankan aplikasi ini adalah sebagai berikut.

Tabel 3-2 Perangkat Lunak

No	Jenis Software	Keterangan
1	Sistem Operasi	Windows 10
2	Editor	Netbean IDE 7.4
3	Bahasa Pemrograman	Java

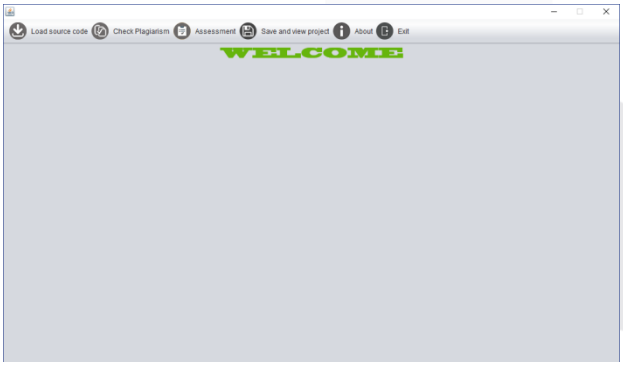
4. Implementasi Dan Pengujian

Implementasi

Tampilan aplikasi menggambarkan antarmuka atau interface Aplikasi Deteksi Plagiaarisme. Tampilan aplikasi dibuat agar pengguna dapat berkomunikasi dengan aplikasi yang dibangun. Berikut adalah antarmuka yang terdapat pada Aplikasi Deteksi Plagiarisme.

Tampilan Aplikasi

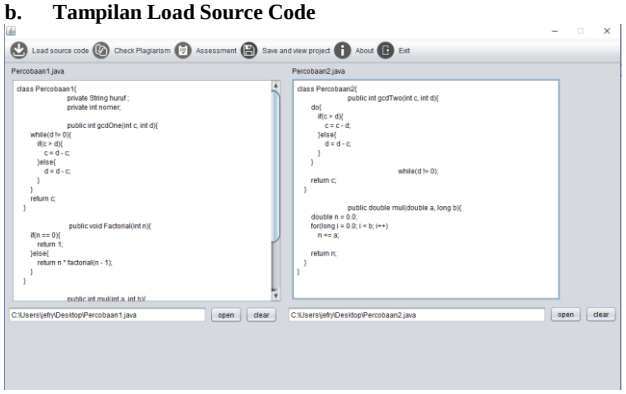
a. Welcome



Gambar 4-1 Welcome

Gambar 4-1 merupakan tampilan pertama yang akan dijumpai user pada saat membuka aplikasi.

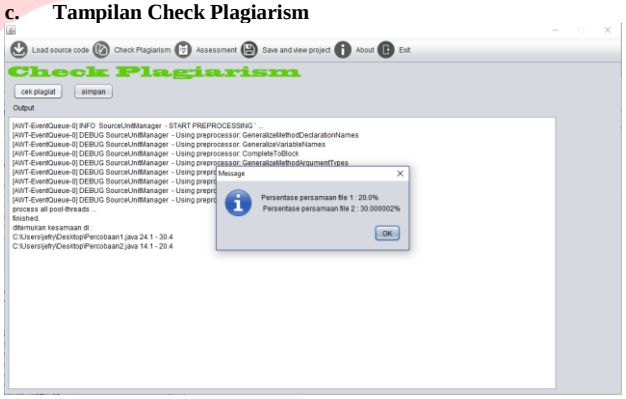
b. Tampilan Load Source Code



Gambar 4-2 Tampilan Load Source Code

Gambar 4-2 merupakan tampilan halaman load source code yang digunakan untuk memasukkan source code yang ingin diperiksa kedalam aplikasi. Maksimal source code yang bisa diperiksa adalah dua file. Pada halaman load source code ini terdapat tombol open untuk membuka file explorer untuk mencari letak source code yang ingin diperikasa, lalu terdapat tombol clear untuk membersihkan text area.

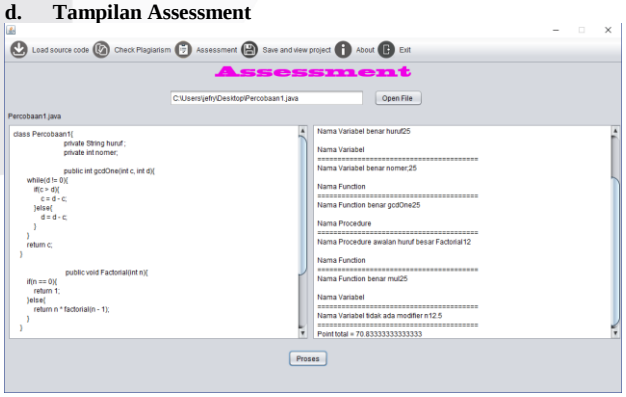
c. Tampilan Check Plagiarism



Gambar 4-3 Tampilan Check Plagiarism

Gambar 4-3 merupakan tampilan halaman check plagiarism yang digunakan untuk mendeteksi kemiripan antar source code. Pada halaman ini terdapat dua tombol yaitu. Tombol cek plagiat yang digunakan untuk memulai proses pendeteksian dan tombol simpan untuk menyimpan hasil dari proses pendeteksian.

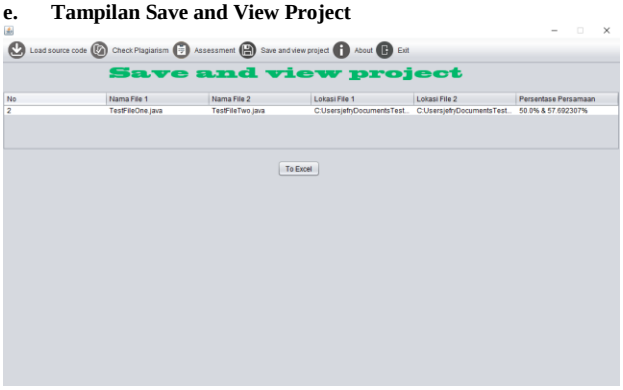
d. Tampilan Assessment



Gambar 4-4 Tampilan Assessment

Gambar 4-4 merupakan tampilan halaman assessment yang digunakan untuk melakukan penilaian terhadap kode program. Pada halaman ini terdapat tombol open file untuk membuka file exlporer dan memilih kode program yang ingin dilakukan penilaian. Lalu, setelah kode

e. Tampilan Save and View Project



Gambar 4-5 Tampilan Save and View Project

Gambar 4-5 merupakan tampilan halaman save and view project yang digunakan untuk menyimpan hasil dari pendeteksian dan penilaian source code. Pada halaman ini terdapat sebuah table dan tombol import to excel untuk mengexport data menjadi file microsoft excel.

Pengujian

Pengujian aplikasi bertujuan untuk menentukan kesalahan yang terdapat dalam aplikasi serta mengetahui apakah program telah sesuai dengan hasil yang diharapkan mau tujuan. Pengujian yang dilakukan dalam aplikasi ini menggunakan metode whitebox yaitu strategi pengujian (testing) yang diterapkan pada mekanisme internal suatu sistem untuk komponen (IEEE, 1990). Strategi ini digunakan untuk mengamati struktur dan logika kode-kode program yang ditulis.

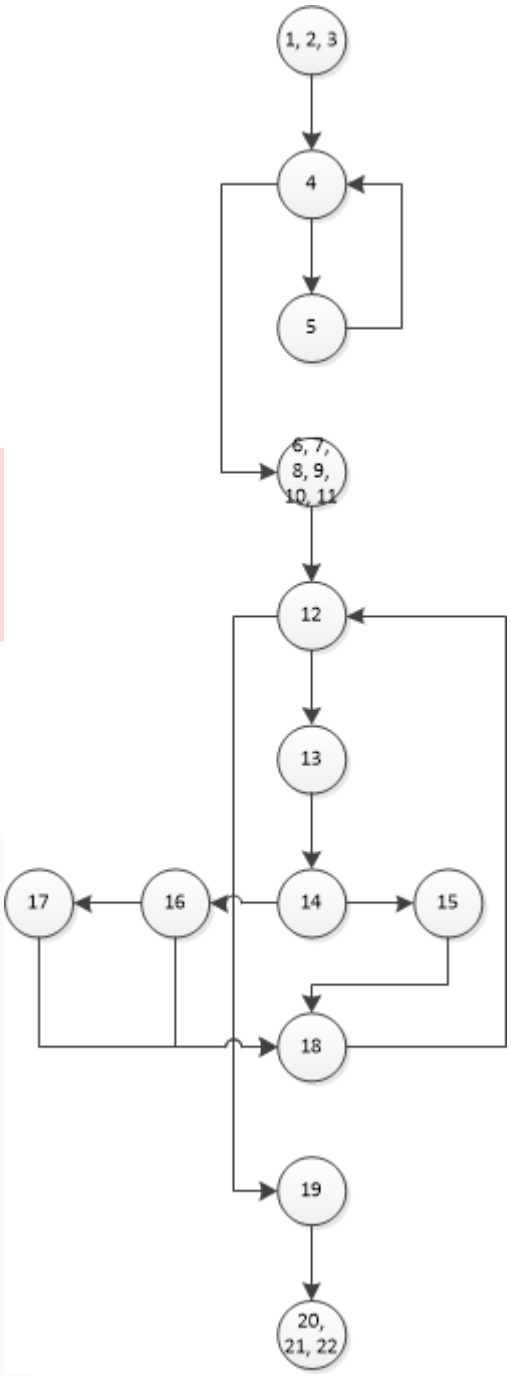
a. Pengujian Potongan Kode Menu Cek Plagiat
Pengujian dilakukan pada potongan kode menu Cek Plagiat. Pengujian ini dilakukan untuk mengetahui alur logika dari potongan kode ketika program dijalankan.

```
HalamanUtama t = new HalamanUtama();
ArrayList<String> arrOut = t.printSimilarityGroups1(detector.process());
String output = "ditemukan kesamaan di :\n";
for (int i = 0; i < arrOut.size(); i++) {
    output += arrOut.get(i) + "\n";
}
taHasil.setText(prepocessing + "\n" + output);
int jumlah_baris1 = lineFileOne;
int jumlah_baris2 = lineFileTwo;
int jumlahSama1 = 0;
int jumlahSama2 = 0;
for (int i = 0; i < t.barisList.size(); i++) {
    Baris baris = t.barisList.get(i);
    if (baris.getJenisFile() == 0) {
        jumlahSama1 += baris.getJumlahBarisSama();
    } else if (baris.getJenisFile() == 1) {
        jumlahSama2 += baris.getJumlahBarisSama();
    }
}
persentase1 = ((float) jumlahSama1 / (float) jumlah_baris1) * 100;
persentase2 = ((float) jumlahSama2 / (float) jumlah_baris2) * 100;
JOptionPane.showMessageDialog(null, "Persentase persamaan file 1 : " + persentase1 + "% \n Persentase persamaan file 2 : " +
persentase2 + "%");
```

Gambar 4-6 Potongan Kode Menu Cek Plagiat

Tahapan dari white box testing adalah sebagai berikut:

1. Menggambarkan kode program yang akan diuji kedalam bentuk graph yaitu node dan edge. Berikut adalah gambaran graph dari potongan kode program (Gambar 4-6):



Gambar 4-7 Flowgraph Potongan Kode Menu Cek Plagiat

2. Basic Path
Independent path yang terbentuk dari flowgraph (Gambar 4-7) adalah sebagai berikut:
1. Path – 1 : 1-2-3-4-5-4-6-7-8-9-10-11-12-13-14-15-18-12-19-20-21-22
2. Path – 2 : 1-2-3-4-5-4-6-7-8-9-10-11-12-13-14-16-18-12-19-20-21-22
3. Path – 3 : 1-2-3-4-5-4-6-7-8-9-10-11-12-13-14-16-17-18-12-19-20-21-22
4. Path – 4 : 1-2-3-4-6-7-8-9-10-11-12-13-14-15-18-12-19-20-21-22
5. Path – 5 : 1-2-3-4-6-7-8-9-10-11-12-13-14-16-18-12-19-20-21-22
6. Path – 6 : 1-2-3-4-6-7-8-9-10-11-12-13-14-16-17-18-12-19-20-21-22

7. Path – 7 : 1-2-3-4-5-4-6-7-8-9-10-11-12-19-20-21-22
8. Path – 8 : 1-2-3-4-6-7-8-9-10-11-12-19-20-21-22

3. Cyclometic Complexity
Dari flowgraph (Gambar 4-7) diketahui bahwa:
Jumlah edge (panah) / E = 16
Jumlah node (lingkaran) / N = 13
Jumlah predicate node / P = 4
Maka, cyclometic complexitynya / V(G) adalah sebagai berikut:
V(G) = E – N +2
V(G) = 16 – 13 + 2
V(G) = 5
Atau,
V(G) = P + 1
V(G) = 4 + 1
V(G) = 5

4. Basis Path Worksheet

Tabel 4-1 Basis Path Worksheet

No Path	arrOut. size	i	barisList. size	i	Baris. getJenisFile	Jumlah Sama1	Jumlah Sama2	Jumlah Baris1	Jumlah Baris2	Presentase1	Presentase2
Path-1	30	29	30	29	0	1	0	30	30	3,33%	0%
Path-2	30	29	30	29	3	0	0	30	30	0%	0%
Path-3	30	29	30	29	1	0	1	30	30	0	3,33%
Path-4	30	30	30	29	0	1	0	30	30	3,33%	0%
Path-5	30	30	30	29	3	0	0	30	30	0%	0%
Path-6	30	30	30	29	1	0	1	30	30	0%	3,33%
Path-7	30	29	30	30	-	0	0	30	30	0%	0%
Path-	30	30	30	30	-	0	0	30	30	0%	0%

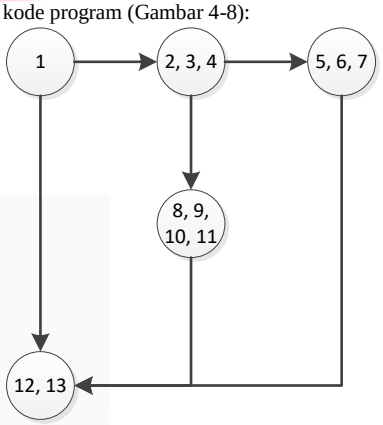
b. Pengujian Potongan Kode Menu Assessment
Pengujian dilakukan pada potongan kode menu Assassment. Pengujian ini dilakukan untuk mengetahui alur logika dari potongan kode ketika program dijalankan.

```
if (perKata[i].equals("class")) {
    namaClass = perKata[i + 1];
    char[] namaClassChar = namaClass.toCharArray();
    if (Character.isUpperCase(namaClassChar[0])) {
        System.out.println("Nama Class = " + namaClass);
        point = 25;
        output = "Nama Class benar " + namaClass +"25";
    } else {
        point = 12;
        output = "Nama Class awalan huruf kecil " + namaClass + "12";
    }
    break;
}
```

Gambar 4-8 Potongan Kode Menu Assessment

Tahapan dari white box testing adalah sebagai berikut:

1. Menggambarkan kode program yang akan diuji kedalam bentuk graph yaitu node dan edge. Berikut adalah gambaran graph dari potongan kode program (Gambar 4-8):



Gambar 4-9 Flowgrap Potongan Kode Menu Assessment

2. Basic Path
Independent path yang terbentuk dari flowgraph (Gambar 4-9) adalah sebagai berikut:

1. Path – 1 : 1-12-13
2. Path – 2 : 1-2-3-4-8-9-10-11-12-13
3. Path – 3 : 1-2-3-4-5-6-7-12-13

3. Cyclometic Complexity
Dari flowgraph (Gambar 4-7) diketahui bahwa:
Jumlah edge (panah) / E = 6
Jumlah node (lingkaran) / N = 5
Jumlah predicate node / P = 2
Maka, cylometic complexitynya / V(G) adalah :
V(G) = E – N + 2
V(G) = 6 – 5 + 2
V(G) = 3
Atau,
V(G) = P + 1
V(G) = 2 + 1
V(G) = 3

4. Basis Path Worksheet

Tabel 4-2 Basis Path Worksheet

No_Path	perKata[i].equals	isUpperCase	Point
Path-1	void	-	-
Path-2	class	True	25
Path-3	class	Flase	12

5. Kesimpilan dan Saran

Kesimpulan

Setelah melakukan analisis, perancangan dan pengujian, maka kesimpulan yang dapat diambil dari Proyek Akhir ini adalah :

1. Aplikasi ini dapat melakukan pendeteksian plagiarisme pada program hasil assessment dan tugas mata kuliah Pemrograman Berorientasi Objek dengan menggunakan fitur Check Plagiat sehingga dapat menemukan dibaris-baris mana saja kesamaan antara kode program yang dibandingkan.

2. Aplikasi ini dapat membantu proses penilaian pada hasil assessment dan tugas mata kuliah Pemrograman Berorientasi Objek menggunakan fitur assessment sehingga dapat mengidentifikasi bagian-bagian dari kode program seperti nama kelas, variabel, prosedur dan fungsi.

Saran

Dalam pembangunan sebuah aplikasi sangat diperlukan sebuah pembaharuan agar aplikasi dapat dikembangkan menjadi lebih baik sehingga dapat memudahkan pihak yang menggunakannya. Maka dari itu untuk pengembangan selanjutnya diharapkan dapat :

1. Menambahkan fitur multiple selection sehingga dapat memasukkan banyak file sekaligus.

2. Menambahkan algoritma untuk dapat mendeteksi dua variabel dalam satu baris.

3. Menambahkan algoritma untuk dapat mendeteksi tipe data bentukan.

DAFTAR PUSTAKA

[1] Abu Hamzah Yusuf Al Arsary, *Pengantar Mudah Belajar Bahasa Arab*. Bandung, Pustaka Adhawa, 2007.

[2] Eka Widhi, *Modul MI 1042 Rekayasa Perangkat Lunak*. Bandung, 2011.

[3] N. Sudjana, *Dasar-dasar proses belajar mengajar*. Bandung , Sinar Baru Algensindo, 2004.

[4] G. Core, *Theory and Practice of Counseling and Psychotheraphy*. Mounterey Brooks/Cole Publishing Company, 1986.

[5] B. Madjidji, *Metodelogi Pengajaran Bahasa Arab*. Yogyakarta, Sumbangsih

[6] A. Hamid, *Pembelajaran Bahasa Arab*. Malang: Miskyat, 2008.

[7] *Peraturan Menteri Pendidikan dan Kebudayaan tentang Kurikulum 2013 Sek Madrasayah Ibtidaiyah*.

[8] S. T. Hakim, *Program Pemula Membaca Kitab Kuning*. Jepara: Falah Offset,

[9] S. Mustofa, *Pembelajaran Mufradat*. Malang, 2010.

[10] D. Salma and E. S, *Pengertian E-Learning*., 2008.

[11] K. Sudarma, *Multimedia Pembelajaran : Teknik Produksi dan Pengembanga*

[12] A. Syarief, *Bedah Action Script : Menguasai Penulisan Script Macromedia F* Elex Media Komputindo Kelompok Gramedia, 2003.

[13] M., *Adobe Flash CS4*. Yogyakarta: CV. Multi Karya, 2010.

[14] R. Pressman, *Software Engineering : A Practitioner's Approach, Seventh Edi* Graw-Hill, 2010.

[15] A. A. R. D. Mutiara, *Testing Implementasi Website Rekam Medis Elektronik Metode Acceptance Testing, vol(8), hlm.2302-3740*., 2014.

[16] R. A. M. Salahudin, *Modul Pembelajaran Rekayasa Perangkat Lunak*. Bandu

[17] A. H. Suyanto, *Step by Step Web Design Theory and Practice Edisi II*. Yogya Publisher, 2008.

[18] Iwan Binanto, *Multimedia Digital Dasar Teory dan Pengembangannya*. Yog Offset, 2010.

[19] R. T. Isnainy R., *Proyek Akhir "Aplikasi Pembelajaran Aksara Sunda untuk S Berbasis Multimedia"*. Bandung : Telkom University, 2013.