DAFTAR PUSTAKA

Armin, F. (2019). Pengembangan Perangkat Lunak Deteksi Potensi Plagiasi Menggunakan Metode Pengelompokan TF-IDF dan Perhitungan Kemiripan Cosine Similarity.

Barakbah, A. R., Karlita, T., & Ahsan, A. S. (2013). *LOGIKA DAN ALGORITMA.* Surabaya: pens.

Biantong, W. S. (2019). Deteksi Clickbait berbahasa Indonesia pada Artikel Berita.

Christen, P. (2006). A Comparison of Personal Name Matching: Techniques and Practical Issues. *THE AUSTRALIAN NATIONAL UNIVERSITY.*

Drabler, K., & Ngomo, A. C. (2015). On the Efficient Execution of Bounded Jaro-Winkler Distances. *AKSW Research Group*, 1-12.

Friendly, F. (2019). Jaro–Winkler Distance Improvement For Approximate String Using Indexing Data For Multiuser Application. *Journal of Physics: Conference Series*, 1-7.

Hidayat, A. (2018, December 15). *Esprima Release Master*. Diambil kembali dari esprima.org: https://esprima.org/

Krisnawati, L. (2016). *Plagiarism Detection for Indonesian Texts.* Munchen: Centrum f¨ur Informations- und Sprachverarbeitung (CIS) Universitat Munchen.

Kunwar, S. (2019, Maret 2). *String Similarity Comparision in JS with Examples*. Diambil kembali dari medium.com: https://medium.com/@sumn2u/string-similarity-comparision-in-js-with-examples-4bae35f13968

Kurniawati, A., Puspitodjati, S., & Rahman, S. (t.thn.). Implementasi Algoritma Jaro-Winkler Distance untuk Membandingkan Kesamaan Dokumen Berbahasa Indonesia.

Leonardo, B., & Hansun, S. (2017). Text Documents Plagiarism Detection using Rabin-Karp and Jaro-Winkler Distance Algorithms. *Indonesian Journal of Electrical Engineering and Computer Science*, 462-471.

Pawar, A. (2013, November 13). *Esprima - What is that*. Diambil kembali dari slideshare.net: https://www.slideshare.net/abhijeetkpawar/esprima-what-is-that

Pr., E. (t.thn.). *Aplikasi Web Node.js.* Creative Commons.

Prasetya, E. B., & Jalal, A. F. (t.thn.). Deteksi Similarity Source Code Menggunakan Metode Deteksi Abstract Syntax Tree. `1-7.

Prasetyo, A., Baihaqi, W. M., & Had, I. S. (2018). ALGORITMA JARO-WINKLER DISTANCE: FITUR AUTOCORRECT DAN SPELLING SUGGESTION PADA PENULISAN NASKAH BAHASA INDONESIA DI BMS TV. *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK)*, 435-444.

Purba, L. J., & Sitorus, L. (2018). Perancangan Aplikasi Untuk Menghitung Persentase Kemiripan Proposal Dan Isi Skripsi Dengan Algoritma Rabin-Karp. *Jurnal Teknik Informatika Unika St. Thomas (JTIUST)*, 17-25.

Putra, E. K., & Rahmayeni, F. (2016). IMPLEMENTASI DATABASE MONGODB UNTUK SISTEM INFORMASI BIMBINGAN KONSELING BERBASIS WEB. *TEKNOIF*, 67-73.

Surahman, A. M. (t.thn.). PERANCANGAN SISTEM PENENTUAN SIMILARITY KODE PROGRAM PADA BAHASA C DAN PASCAL DENGAN MENGGUNAKAN ALGORITMA RABIN-KARP.

Surahman, A. M. (t.thn.). PERANCANGAN SISTEM PENENTUAN SIMILARITY KODE PROGRAM PADA BAHASA C DAN PASCAL DENGAN MENGGUNAKAN ALGORITMA RABIN-KARP.

Susanto, D., Basuki, A., & Duanda, P. (2016). Deteksi Plagiat Dokumen Tugas Daring Laporan Praktikum Mata Kuliah Desain Web Menggunakan Metode Naive Bayes. *Nusantara Journal of Computers and its Applications*, 1-9.

Tannga, M. J., Rahman, S., & Hasniati. (2017). ANALISIS PERBANDINGAN ALGORITMA LEVENSHTEIN. *JTRISTE*, 44-54.

Tinaliah, & Elizabeth, T. (2018). Perbandingan Hasil Deteksi Plagiarisme Dokumen dengan Metode JaroWinkler Distance dan Metode Latent Semantic Analysis. *Jurnal Teknologi dan Sistem Komputer*, 7-12.

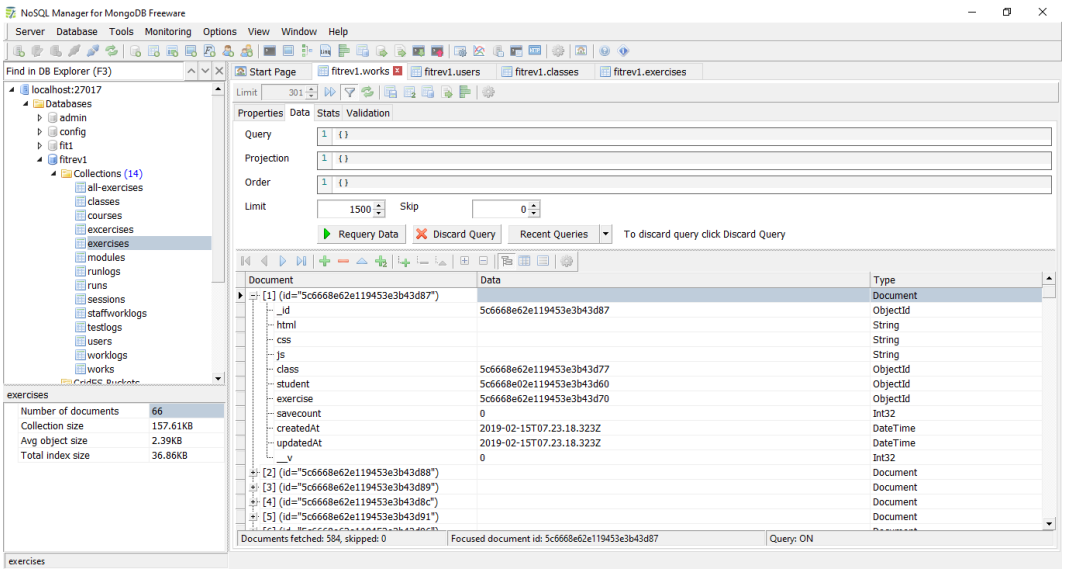
Wali, M., & Safrizal. (2018). Similar text sebagai Pengkodean Aplikasi Plagiarisme. *Jurnal JTIK (Jurnal Teknologi Informasi dan Komunikasi)*, 11-19.

Yancy, W. E. (2005). *Evaluating String Comparator Performance for Record Linkage.* Washington: Statistical Research Division.

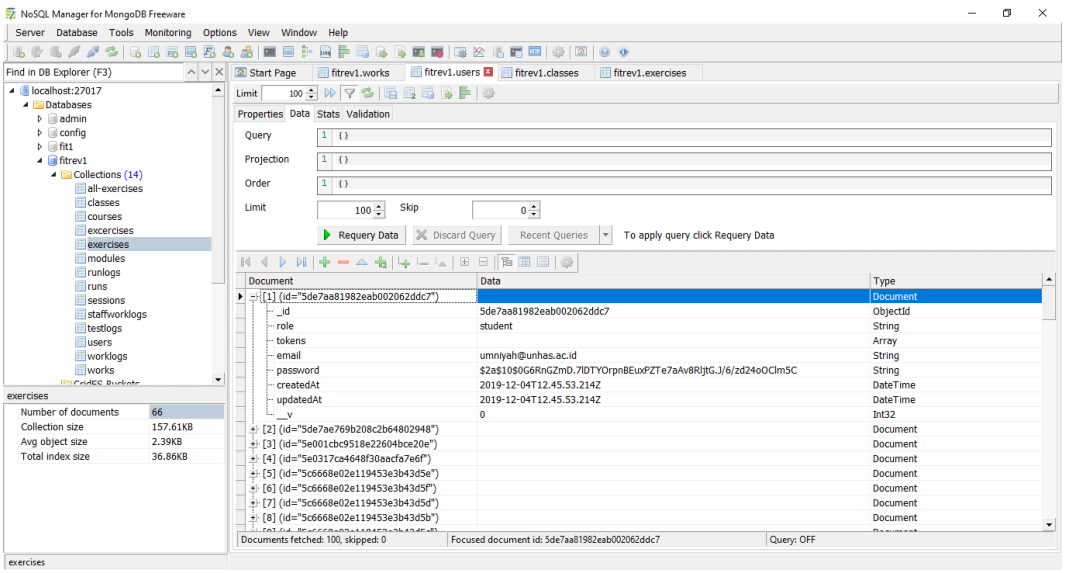
LAMPIRAN

Lampiran 1 (Database Sistem)

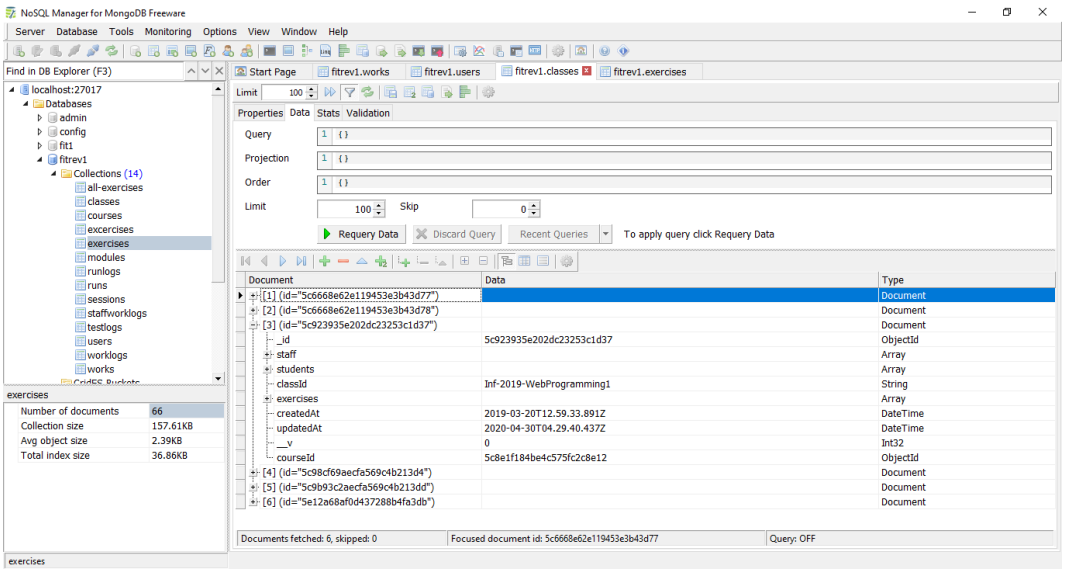
- Tabel Works



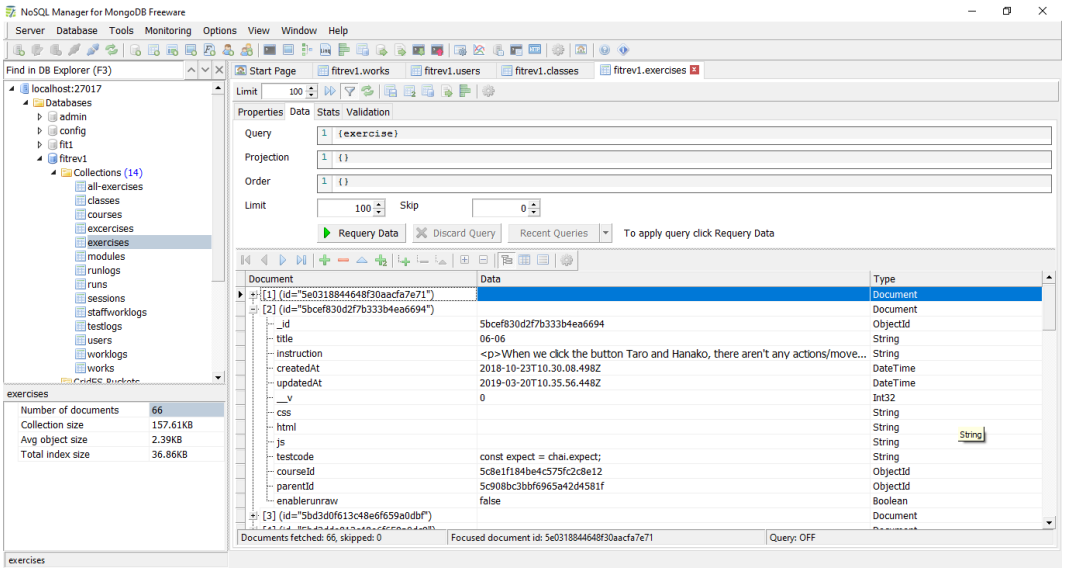
- Tabel Users



- Tabel Classes



- Tabel Exercises



Lampiran 2 (Codingan Sistem)

- Esprima

```
var esprima = require('esprima');

function traverse(node, func) {
    func(node);//1
    for (var key in node) { //2
//      console.log(key);
        if (node.hasOwnProperty(key)) { //3
            var child = node[key];
            if (typeof child === 'object' && child !== null) { //4

                if (Array.isArray(child)) {
                    child.forEach(function(node) { //5
                        traverse(node, func);
                    });
                } else {
//                  console.log('tidak punya anak');
                    traverse(child, func); //6
                }
            }
        }
    }
}
function analyzeCode(code) {
  var ast = esprima.parse(code);
//console.log(ast);
//return JSON.stringify(ast);
  // console.log(ast.body[0]);

    let temp = '';
    traverse(ast, function(node) {
    // console.log(node.type);
        if(node.type==='Program') node.type = 'A';
        if(node.type==='AssignmentExpression') node.type = 'B';
        if(node.type==='BinaryExpression') node.type = 'C';
        if(node.type==='BlockStatement') node.type = 'D';
        if(node.type==='CallExpression') node.type = 'E';
        if(node.type==='ExpressionStatement') node.type = 'F';
        if(node.type==='FunctionDeclaration') node.type = 'G';
        if(node.type==='Identifier') node.type = 'H';
        if(node.type==='Literal') node.type = 'I';
        if(node.type==='MemberExpression') node.type = 'J';
        if(node.type==='VariableDeclaration') node.type = 'K';
        if(node.type==='VariableDeclarator') node.type = 'L';
        if(node.type==='FunctionExpression') node.type = 'M';
        if(node.type==='WhileStatement') node.type = 'N';
        if(node.type==='IfStatement') node.type = 'O';
        if(node.type==='LogicalExpression') node.type = 'P';
        if(node.type==='ReturnStatement') node.type = 'Q';
        if(node.type==='ArrayExpression') node.type = 'R';
        if(node.type==='EmptyStatement') node.type = 'S';

        temp += node.type;

    });
    return temp;
}
//analyzeCode(program);

module.exports = analyzeCode;
```

- Algoritma Jaro-Winkler

```
(function () {
        JaroWrinker2  = function (s1, s2) {
        var m = 0;
        var temp_1 = [];
        var temp_2 = [];
        // Exit early if either are empty.
        if ( s1.length === 0 || s2.length === 0 ) {
            return 0;
        }
        // Exit early if they're an exact match.
        if ( s1 === s2 ) {
            return 1;
        }
        var range     = (Math.floor(Math.max(s1.length, s2.length) / 2)) - 1,
            s1Matches = new Array(s1.length),
            s2Matches = new Array(s2.length);
        for ( i = 0; i < s1.length; i++ ) {
            var low  = (i >= range) ? i - range : 0,
                high = (i + range <= s2.length) ? (i + range) : (s2.length - 1);
            for ( j = low; j <= high; j++ ) {
            if ( s1Matches[i] !== true && s2Matches[j] !== true && s1[i] === s2[j] ) {
                ++m;
                s1Matches[i] = s2Matches[j] = true;
                temp_1.push(i);
                temp_2.push(j);
                break;
            }
            }
        }
        // Exit early if no matches were found.
        if ( m === 0 ) {
            return 0;
        }

        // Count the transpositions.
        var k = n_trans = 0;
        for ( i = 0; i < s1.length; i++ ) {
            if ( s1Matches[i] === true ) {
            for ( j = k; j < s2.length; j++ ) {
                if ( s2Matches[j] === true ) {
                k = j + 1;
                break;
                }
            }
            if ( s1[i] !== s2[j] ) {
                ++n_trans;
            }
            }
        }
        var weight = (m / s1.length + m / s2.length + (m - (n_trans / 2)) / m) / 3,
            l      = 0,
            p      = 0.1;
        if ( weight > 0.7 ) {
            while ( s1[l] === s2[l] && l < 4 ) {
            ++l;
            }
            weight = weight + l * p * (1 - weight);
        }
       var result = [];
       result['weight'] = weight;
       result['temp1'] = temp_1;
       result['temp2'] = temp_2;
        return result;
    }
})();
module.exports = JaroWrinker2;
```

- App.js

```
/**
 * Controllers (route handlers).
 */
const homeController = require('./controllers/home');
const userController = require('./controllers/user');
const similarityController = require('./controllers/similarity');
//const apiController = require('./controllers/api');


/**
 * Primary app routes.
 */
app.get('/', homeController.index);
app.get('/login', userController.getLogin);
app.get('/similarity', similarityController.similarity);
app.post('/login', userController.postLogin);
app.get('/logout', userController.logout);
app.get('/forgot', userController.getForgot);
app.post('/forgot', userController.postForgot);
app.get('/reset/:token', userController.getReset);
app.post('/reset/:token', userController.postReset);
app.get('/signup', userController.getSignup);
app.get('/signup-roles', userController.getSignupRoles);
app.post('/signup', userController.postSignup);
app.get('/contact', contactController.getContact);
app.post('/contact', contactController.postContact);
app.get('/account', passportConfig.isAuthenticated, userController.getAccount);
app.post('/account/profile', passportConfig.isAuthenticated, userController.postUpdateProfile);
app.post('/account/password', passportConfig.isAuthenticated, userController.postUpdatePassword);
app.post('/account/password-by-staff', passportConfig.isAuthenticated, userController.postUpdateP
app.post('/account/delete', passportConfig.isAuthenticated, userController.postDeleteAccount);
app.get('/account/unlink/:provider', passportConfig.isAuthenticated, userController.getOauthUnlin


/**
 * SPELL routes
 */

app.get('/class', passportConfig.isAuthenticated, classController.index);
app.post('/class',  passportConfig.isAuthenticated, classController.create);

//ambil satu kelas pakai id, isinya ada daftar user yang ada di kelas tersebut
app.get('/class/:classId', passportConfig.isAuthenticated, classController.getById);
app.put('/class/:classId', passportConfig.isAuthenticated, classController.updateById);
app.get('/class/delete/:classId', passportConfig.isAuthenticated, classController.deleteById);
app.get('/classedit/:classId', passportConfig.isAuthenticated, classController.editById);

app.get('/similarity/:simId', passportConfig.isAuthenticated, similarityController.similarity);
app.get('/similarity/check/tugas/:simId/:classId', passportConfig.isAuthenticated,
        similarityController.listMahasiswa);
app.get('/similarity/check/:simId', passportConfig.isAuthenticated, similarityController.fungsiSendiri);
app.get('/similarity/:classId', passportConfig.isAuthenticated, similarityController.similarity);
app.get('/similarity/check/tugas/:exId/similarity/:simId', passportConfig.isAuthenticated,
        similarityController.tugasSama);
app.get('/similarity/check/tugas/:exId/similarity/:simId1/:simId2', passportConfig.isAuthenticated,
        similarityController.perbandinganTugas);
```

- Controllers(similarity.js)

```
const User = require('../models/User');
const Class = require('../models/Class');
const Exercise = require('../models/Exercise');
const Work = require('../models/Work');
const JaroWrinker = require('../library/jaro');
const JaroWrinker2 = require('../library/jaro2');
const analyzeCode = require('../library/esprima');
const crypto = require('crypto');

exports.similarity= async(req, res) => {
  if (req.user.role == 'student')
      return res.redirect('/student');
    const [classData, exerciseData] = await Promise.all([
      Class.findById(req.params.simId).exec(),
      Exercise.find({}).sort('title').exec(),
      ]);
    const tampilData = await Class.findById(req.params.simId)
      .populate('exercises.exId', 'title')
      .exec();
    var exercise = exerciseData
      .map((s) => {
        s=Object.assign({},s._doc);
        s.inClass = classData.exercises.indexOf(s._id);
        return s;
      })
    return res.render('similarity/similarity', {title: 'SPELL IDE', classData, exerciseData, tampilData});
};
exports.listMahasiswa = async(req,res) => {
  if (req.user.role == 'student')
    return res.redirect('/student');
  const [workData, exerciseData, judul] = await Promise.all([
    Work.find({exercise: req.params.simId}).exec(),
    Exercise.find({}).sort('title').exec(),
    Exercise.findById(req.params.simId)
  ]);

    const kelasData = await Class.findById(req.params.classId)
      .populate('students')
      .exec();
    const cobaData = await Work.find({exercise: req.params.simId})
      .populate('class')
      .populate('student')
      .populate('exercise')
      .exec();
    var kelas = cobaData;
    let mahasiswa2 = [];
    let color = [];
    let data = [];
    let tugasKembar = [];
    let cheatList = [];
    let index = 0;
    let mahasiswa = [];
    for (f=0; f<kelasData._doc.students.length; f++){
      for(r=0; r<kelas.length; r++)
      {
        if(kelasData.students[f].email == kelas[r]._doc.student.email)
        {
          mahasiswa.push(kelas[r]);
        }
      }
    }
    for (u = 0; u < mahasiswa.length - 1; u++){
      if(mahasiswa[u]._doc.js !=''){
        for (n = u+1; n < mahasiswa.length; n++){
          if(mahasiswa[n]._doc.js !=''){
            let analyzeCode1 = analyzeCode(mahasiswa[u]._doc.js);
            let analyzeCode2 = analyzeCode(mahasiswa[n]._doc.js);
            let md5 = crypto.createHash('md5').update(analyzeCode1).digest('hex');
            hasil = JaroWrinker(analyzeCode1, analyzeCode2);
            persen = hasil.toFixed(3)*100 + '%';
            data[index] = [];
```

```
            if (hasil <= 0.9) {
              data[index]['nilai'] = hasil;
              data[index]['persen'] = persen;
              data[index]['mahasiswa1'] = mahasiswa[u];
              data[index]['mahasiswa2'] = mahasiswa[n];
              warna = '#5acc65';
              data[index]['color'] = warna;
            }
            if (hasil > 0.9) {
              if(hasil === 1) {
                if(typeof tugasKembar[md5] === 'undefined') tugasKembar[md5] = [];
                  if(tugasKembar[md5].indexOf(mahasiswa[u])<0)
                    {tugasKembar[md5].push(mahasiswa[u]); }
                  if(tugasKembar[md5].indexOf(mahasiswa[n])<0)
                    {tugasKembar[md5].push(mahasiswa[n]); }
                  if(cheatList.indexOf(mahasiswa[u]) < 0 || cheatList.indexOf(mahasiswa[n]) < 0)
                    cheatList.push(mahasiswa[n]);
              }
              data[index]['nilai'] = hasil;
              data[index]['persen'] = persen;
              data[index]['mahasiswa1'] = mahasiswa[u];
              data[index]['mahasiswa2'] = mahasiswa[n];
              warna = '#ff0000';
              data[index]['color'] = warna;
            }
            index++;
          }
        }
      }
    }
    var data2 = data.filter(function(value, index, arr){
      return ((cheatList.indexOf(value.mahasiswa1) < 0) && (cheatList.indexOf(value.mahasiswa2) < 0));
    });
    data2.sort(function(a,b){return b['nilai']-a['nilai']});
    let newTugas = [];
    for(x in tugasKembar){
      newTugas.push(tugasKembar[x]);
    }
    return res.render('similarity/listMahasiswa', {title: 'Persentasi Similarity', data2, newTugas,
      judul, kelasData});
}
exports.fungsiSendiri = async(req,res) => {
  if (req.user.role == 'student')
    return res.redirect('/student');
  const [workData, exerciseData, judul] = await Promise.all([
    Work.find({exercise: req.params.simId}).exec(),
    Exercise.find({}).sort('title').exec(),
    Exercise.findById(req.params.simId)
  ]);
  const test = await Work.find({_id:'5c6668e62e119453e3b43d87'})
    .populate('class')
    .exec();
    console.log(test)
}
exports.tugasSama= async(req, res) => {
    if (req.user.role == 'student')
      return res.redirect('/student');
    const tugas = await
      Work.findById(req.params.simId)
      .populate('student');
    var esprima = analyzeCode(tugas._doc.js);
    return res.render('similarity/tugasSama', {title: 'Hasil Similarity', tugas, esprima})
};
exports.perbandinganTugas= async(req, res) => {
    if (req.user.role == 'student')
      return res.redirect('/student');
    const[tugas1,tugas2] = await Promise.all([
      Work.findById(req.params.simId1)
      .populate('student'),
      Work.findById(req.params.simId2)
```

```
      .populate('student'),
    ]);
  var esprima1 = analyzeCode(tugas1.js);
  var esprima2 = analyzeCode(tugas2.js);
  var akhir = JaroWrinker2(esprima1,esprima2)['weight'];
  var hasil = (JaroWrinker2(tugas1.js, tugas2.js)['weight']).toFixed(3)*100 + '%'
  esprima = JaroWrinker2(analyzeCode(tugas1.js), analyzeCode(tugas2.js))['weight'].toFixed(3)*100 + '%'
  var value = (JaroWrinker2(analyzeCode(tugas1.js), analyzeCode(tugas2.js)));
  var nilai = value['weight'];
  var uun1 = value['temp1'];
  var uun2 = value['temp2'];
  var cari = []
  var simsimi = []
  var flag = false;
  for ( p = 0; p < esprima1.length; p++ ){
    for ( l = 0; l < uun1.length; l++ ){
      if ( p !== uun1[l]){
        flag = false;
        for ( y = 0; y < uun1.length; y++){
          if ( p === uun1[y]){
            console.log(p + " dan " + y)
            console.log("sekarang " + p + " dan " + uun1[l]);
            console.log("oke huruf yang sama " + esprima1[p]);
            cari.push(esprima1[p]);
            simsimi.push('#ff0000');
            flag = true;
            break;
          }
        }
        if ( p !== uun1[l] ){
          console.log("sekarang " + p + " dan " + uun1[l]);
          console.log('huruf disini adalah ' + esprima1[p])
          if(flag==false){
          cari.push(esprima1[p]);
          simsimi.push('#000000');
        }
          break;
        }
      }
    }
  }
  var cari2 = []
  var simsimi2 = []
  flag = false;
  for ( p = 0; p < esprima2.length; p++ ){
    for ( l = 0; l < uun2.length; l++ ){
      if ( p !== uun2[l]){
        flag = false;
        for ( y = 0; y < uun2.length; y++){
          if ( p === uun2[y]){
            console.log(p + " dan " + y)
            console.log("sekarang " + p + " dan " + uun2[l]);
            console.log("oke huruf yang sama " + esprima2[p]);
            cari2.push(esprima2[p]);
            simsimi2.push('#ff0000');
            flag = true;
            break;
          }
        }
        if ( p !== uun2[l] ){
          console.log("sekarang " + p + " dan " + uun2[l]);
          console.log('huruf disini adalah ' + esprima2[p])
          if(flag==false){
          cari2.push(esprima2[p]);
          simsimi2.push('#000000');
        }
          break;
        }
      }
    }
  }
    if (akhir <= '0.9') {
     warna = '#5acc65';
    }
    if (akhir > '0.9') {
     warna = '#ff0000';
    }
  return res.render('similarity/perbandinganTugas', {title: 'Hasil Similarity', simsimi, cari, simsimi2,
    cari2, tugas1, tugas2, esprima, esprima1, esprima2})
};
```

- View

(similarity.pug)

```
extends ../layout

block content
  .container
    .page-header
      h3 #{ title }
      h4 Web Programming
      h4 #{tampilData.classId}
      h4 List Nomor Tugas

      ul.list-group
      each tugas in tampilData.exercises
        li.list-group-item
          a.nounderline(href='/course/exercise/' + tampilData.courseId + '/' + tugas.exId.id ) #{tugas.exId.title}&
          nbsp
          span.pull-right
           a.badge(href='/similarity/check/tugas/' + tugas.exId.id + '/' + tampilData.id, style="margin-right:5px;"
           )  Similarity Check
```

(checksimilarity.pug)

```
extends ../layout
block content
  .container
    .page-header
      h3 #{ title }
      h4 Web Programming

      table.table.table-striped.table-bordered.table-hover
       tr
        td Nama/Nim
        td Nama/Nim
        td Hasil Cek
       each val in exercise
        tr
         td #{val.title}
         td #{val.title}
         td.p-3.mb-2.bg-success.text-white #{val.title}
```

(tugasSama.pug)

```
extends ../layout
block content
  .container
    .page-header
      h3 #{ title }
      h4 Web Programming
      p
       font(color='red' ) Tingkat Kesamaan Tugas Mahasiswa 100%
      table.table.table-striped.table-bordered.table-hover
        tr
         td Email Mahasiswa
         td #{tugas.student.email}
        tr
         td Tugas Mahasiswa
         td #{tugas.js}
        tr
         td Hasil Esprima
         td #{esprima}
      h4 Keterangan:
      h5 A = Program
      h5 B = AssignmentExpression
      h5 C = BinaryExpression
      h5 D = BlockStatement
      h5 E = CallExpression
      h5 F = ExpressionStatement
      h5 G = FunctionDeclaration
      h5 H = Identifier
      h5 I = Literal
      h5 J = MemberExpression
      h5 K = VariableDeclaration
      h5 L = VariableDeclarator
      h5 M = FunctionExpression
      h5 N = WhileStatement
      h5 O = IfStatement
      h5 P = LogicalExpression
      h5 Q = ReturnStatement
      h5 R = ArrayExpression
      h5 S = EmptyStatement
```

(perbandinganTugas.pug)

```
extends ../layout
block content
  .container
    .page-header
      h3 #{ title }
      h4 Web Programming
      p
       font(color=warna ) Tingkat Kesamaan Tugas Mahasiswa : #{esprima}
      table.table.table-striped.table-bordered.table-hover
        tr
         td Email Mahasiswa
         td #{tugas1.student.email}
         td #{tugas2.student.email}
        tr
         td Tugas Mahasiswa
         td #{tugas1.js}
         td #{tugas2.js}
        tr
         td Hasil Esprima
         td
          each data, index in simsimi
           font(color=data) #{cari[index]}
          td
           each data2, index in simsimi2
            font(color=data2) #{cari2[index]}
```

```
      h4 Keterangan:
      h5 A = Program
      h5 B = AssignmentExpression
      h5 C = BinaryExpression
      h5 D = BlockStatement
      h5 E = CallExpression
      h5 F = ExpressionStatement
      h5 G = FunctionDeclaration
      h5 H = Identifier
      h5 I = Literal
      h5 J = MemberExpression
      h5 K = VariableDeclaration
      h5 L = VariableDeclarator
      h5 M = FunctionExpression
      h5 N = WhileStatement
      h5 O = IfStatement
      h5 P = LogicalExpression
      h5 Q = ReturnStatement
      h5 R = ArrayExpression
      h5 S = EmptyStatement
```