COMP5347 Web Application Development
Group Assignment Technical Report
Group: 6

Tutor: Johan Alibasa
Liang-Chun Yu (480317999)
Ruotao Lin (480509930)
Jinvara Vesvijak (470500387)

1. Abstract

In this assignment, our task is to build a small data analytic web application to analyze the individual and overall articles of Wikipedia and their authors.

2. Introduction

The main page of the application is where the user can see and implement the analytics functionalities. But the user has to Sign-up or Login before being able to access the main page. This is done by typing http://localhost:3000 in the browser, after successfully logged-in, the page is redirected to http://localhost:3000/main where the main page is accessed. The functions are divided into overall article analytics, individual article analytics, and author analytics.

In the overall article analytics part, we separately show titles of the two articles with the highest and lowest number of revisions. The user is allowed to change the number of articles shown. We also individually show the title of the articles edited by the largest and smallest group of registered users. Lastly, we show the top 2 articles with the longest history and an article with the shortest history, by calculating the duration between now and its creation time. A bar chart and pie chart is provided, showing the distribution of revisions by year and user type, and the revision number distribution by user type. The visual analytics gives our application users a clearer view of the analysis.

For the individual article analytics part, we provide a drop-down list to select the available article from the dataset and a search bar to allow end users to type the title of interest. Once the article is selected, we check if the history of the article is up to date with the boundary of 24 hours. We query MediaWiki API to pull all possible new revisions and show the number of newly pulled revisions on the page as well as the title, total number of revisions, top 5 regular users of the chosen article. For the visual analytics, we show the same two charts from the overall analytics part but specified to just the chosen article, and a year filter function is added. In addition, we show a bar chart of revision numbers by year made by one or more of the top 5 users for this article.

Finally, for the author analytics, we allow end users to display articles of a specific author by typing the author name in a free text format. The name of the author and the number of revisions is shown. End users are also allowed to see the timestamps of the article if by selecting the article title from the drop-down list.

3. Overall MVC structure

We applied the Model View Controller (MVC) structure for our architecture, setting 5 sub-folders in our app folder, routes, controllers, models, import, and views. Below we show the overview of our architecture:

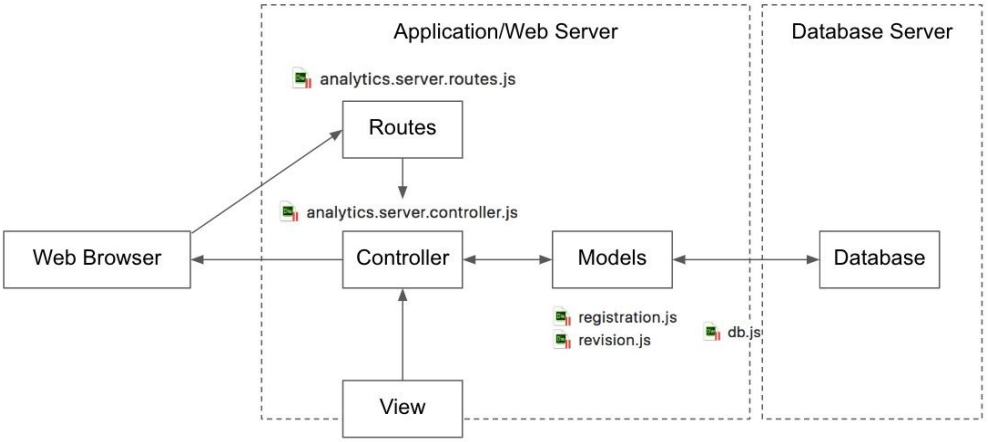

Figure 1: Architecture of MVC structure.

In the **routes** folder:

- *anlytics.server.routes.js* is used to map between the controller and HTTP requests.

In the **controllers** folder:

- *analytics.server.controller.js* controls the logic part of the application, calling functions from other models, returning results back to the browser. The controller is invoked by the routes.

In the **models** folder:

- *db.js* connects with the database.
- *revision.js* contains the functions to process data analytics tasks.
- *registration.js* deals with log-in and registration tasks.

In the **import** folder:

- *importJSON.js* imports the JSON dataset into the database.
- *update.js* updates the 4 administrator types as well as the bot users type.

In the **views** folder, the ejs files in the folder call a js file in the **public/js** folder as shown below:

- *index.ejs -> index.js* the page to log-in, register as a user.
- *main.ejs -> main.js* display all page components including the sidebar, the overall analytic results, and input forms for users to query the article or author of their interest for individual analytic and author analytic results.
- *highLowRevResult.ejs* display the articles with the highest and lowest number of revisions by a number of articles.
- *individualArticleResult.ejs -> individual.js* display the individual analytics results.
- *finalBarChartTop5.ejs -> articletop5.js* visualizes bar chart of revision number distributed by year made by one or a few of the top 5 users for the article.
- *authourAnalyticsResult.ejs -> author.js* displays the author analytics results.
- *timestampResult.ejs* shows the timestamp of the selected author and article.

The static files are in the public folder, containing the JS and CSS files. The main file for our application, *sever.js* is in the root folder.

4. Front end structure

Our application consists of two pages, the Log-in/Register page, and the Main page. First of all, the style of our application is in the *public/css/style.css* file, to make the font, color, structure reasonable and good-looking.

In the Log-in/Register page, users will be shown with a log-in form, or else users can choose to register a new account. Validation of new-users or checking if the logged-in information is identical to any of our users stored in the database. This is done by showing *index.ejs* file and extracting the script from *index.js*. After the user is successfully logged in, the page is redirected to the main page with url http://localhost:3000/main/.

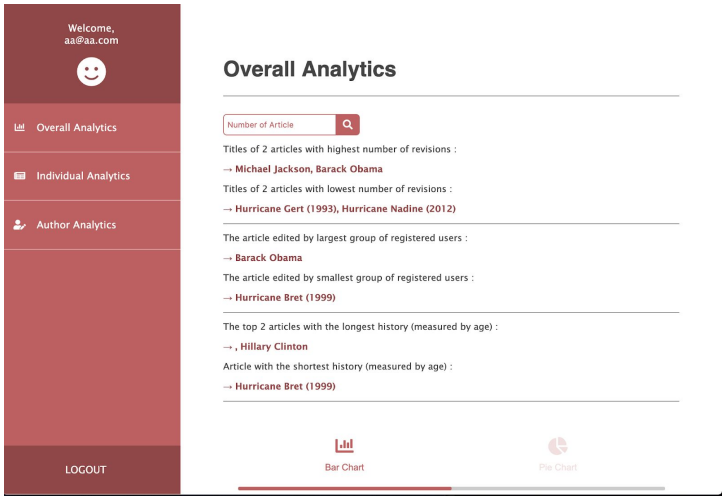

Figure 2: Display of the main page.

The main page is where all the analytics is shown, by showing the *main.ejs* file. The main page is split to the sidebar (id=”contentLeft”), and the analytics (id=”contentRight”), shown in figure 3. For the sidebar, we have Overall Analytics, Individual Analytics, Author Analytics buttons, as well as the Logout button. By default, we show the overall analytics, but when users click on other analytics buttons, the display of contentRight is changed to the selected area, defined in the *main.js* file.

```
//Display Overall section
$('#overallMenu').on('click', function(e){
  $('#overallSection').css("display", "block");
  $('#individualSection').css("display", "none");
  $('#authorSection').css("display", "none");
});

//Display Individual section
$('#individualMenu').on('click', function(e){
  $('#overallSection').css("display", "none");
  $('#individualSection').css("display", "block");
  $('#authorSection').css("display", "none");
});

//Display Author section
$('#authorMenu').on('click', function(e){
  $('#overallSection').css("display", "none");
  $('#individualSection').css("display", "none");
  $('#authorSection').css("display", "block");
});
```

Figure 3: Display of each analytic part from main.js.

When showing the default overall analytics, the top 2 highest and lowest revisions, the result is parsed to main.ejs from the results of functions written in the controller. User can choose the number of the top revisions they want to see, from the main.js, we use jquery AJAX to update our page asynchronously. A get request is sent to the routes, then calling functions in the controller. The result is then written to highLowRevResult.ejs, then parsed into main.ejs from the callback function of the initial get request.

```
//When submitted author's name, respond with that author's data analytics result
$('#selectAuthorSubmit').on('click', function(e){
  var author = $('#selectAuthor').val();
  var encodedAuthor = encodeURIComponent(author);
  console.log(author);
  $.get("/main/author?user=" + encodedAuthor, function(result) {
    //console.log(result);
    $('#authorerror').html("");
    $('#authorAnalyticsResult').html(result);
  });
});
```

Figure 4: Sample ajax request from main.js

The process is similar when getting the analytics results for the individual and author analytics, sending a get request to the routes, calling the function in controllers, writing results to an extended ejs file, then by the callback function of the get request, the result is parsed back to the main.ejs file. Figure 4 shows the code when the user clicks the button to select a specific article or searches a specific author in the main.js file.

The visualization of the analysis is from the Google Visualization API, to draw bar charts and pie charts of our results.

5. Server-side structure

Framework of MVC

This project is using MVC architectural. Model View Controller(MVC) is a useful pattern in Web application framework design. The MVC pattern is a proven, effective way for the generation of organized modular applications [1]. There are three different modules when using MVC pattern: model, view, controller, which associate with each other to implement all functions that underlying classes. By using MVC, the programmers can reduce the syntax error of SQL in command, and can easily control the users' events and render all data to the browser. MVC helps to reduce the complexity of program architectural and increase the flexibility of writing code.

Routing

the routing in MVC architectural is a mapper to associate work with front-end and backend system. it will make easier to handle the request and send the response to users(figure 5).

```
router.get('/', controller.showForm);
router.post('/main', controller.loginRegister);
router.get('/main', controller.showMain);
router.get('/main/getHighLowRev', controller.getHighLowRev);
router.get('/main/article', controller.showIndividualResult);
router.get('/main/article/getBar', controller.getIndividualBarChartTop5);
router.get('/main/author', controller.showAuthorResult);
router.get('/main/author/showTimestamp', controller.showTimestampResult);
router.get('/logout', controller.logout);
```

Figure 5: Routing code

Different requests will implement different functions that wrote in the Controller. For example, when the server is on, use request: localhost:3000/, the "showForm" function that set up in the controller will be activated and return the response to the client side.

Controller

In the controller, all functions that user may use are written down in this section. At first, before the user tries to login the main page, the controller will detect if there is a valid session in the user's browser. If the session is still valid, the user can access the main page directly without going through login/registration page. The registration.js file contains the models to query or insert data into the database collection "registration". In the controller, the function "loginRegister" will check if the user's login/registration information matches with any entity in the database. If a condition is not met such as registering email is already existed, the "getIndex" function will show the login/registration page with an error message. If all conditions match, then the session will be stored and "getMain" function will render the main page. For the overall analytics section, all related querying functions are in the "getMain" function. In this function, all results will be rendered to the main.ejs file to display the query results and the charts.

```
function parseAllResult(allResult, res, count, viewfile){
  console.log(count);
  if(count < 10){return;}
  else{res.render(viewfile, {allResult: allResult});}
}
```

Figure 6: parseAllResult function code

The "parseAllResult" function is used to check if all functions in "getMain" have been executed before rendering the results to main.ejs(figure 6). For the individual article analytics, when the user chooses one article title in the list, an ajax request will be sent and the "getIndividualResults" function will render all the results to "individualArticleResults.ejs" file and show on the main page. The similar flows are applied to other ajax requests contained in this project.

Models

In the models, all queries related to data analytics are written in revisions.js. And queries related to user login/registration are written in registration.js. Mongoose module is used to connect and communicate with the database(figure 7).

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

mongoose.connect('mongodb://localhost/testeddie', { useNewUrlParser: true }, function () {
  console.log('mongodb connected to testeddie!');
});

module.exports = mongoose;
```

Figure 7: Code for connecting with database

In order to make a query or insert new data, a schema and document need to be defined before reading/writing on MongoDB (figure 8).

```
var RevisionSchema = new mongoose.Schema({
  title: String,
  timestamp:String,
  user:String,
  anon:String,
  usertype:String,
  registered:Boolean,
  admintype:String
},{
  versionKey: false
});
```

```
var RegistrationSchema = new mongoose.Schema({
  email: String,
  password:String,
  firstname:String,
  lastname:String
},{
  versionKey: false
});
```

Figure 8: Code of the schemas

6. Pre-work: Import data into MongoDB and Update

To quickly import all the JSON file into the database, the file called “importJson.js” needs to be executed to run the mongoose command to import all JSON files into the “revisions” collection. Different types of users in the original JSON files need to be distinguished in order to get the correct result when doing the query. The “update.js” file needs to be executed after the import to update and insert usertype, registered, and admintype into the database. Below is one example that shows the updates to be made for all admin users (figure 9). Instruction on pre-work required before running the server is in “READ ME.txt” located in the root folder.

```
var Revision = mongoose.model('Revision', RevisionSchema, 'revisions');

//Set usertype for all admin users
Revision.update(
  {'user':{'$in':allAdmin, '$nin':bot}},
  {$set:{usertype:'admin',registered:true}},
  {multi: true, upsert:true},
  function(err,result){
  if (err){console.log("ERROR")}
  else{
    console.log('admin usertype has been updated/inserted');
    console.log(result);
  }}
);
```

Figure 9: Example of a model for updating Admin users

Reference

1. Hofmeister C, Nord R.L, Soni D (2000) Applied Software Architecture, Addison-Wesley.

Appendix: Source Code

Codes in app folder

File: app/routes/analytices.server.routes.js

```
var express = require('express')
var controller = require('../controllers/analytics.server.controller')
var router = express.Router();

router.get('/', controller.showForm);
router.post('/main', controller.loginRegister);
router.get('/main', controller.showMain);
router.get('/main/getHighLowRev', controller.getHighLowRev);
router.get('/main/article', controller.showIndividualResult);
router.get('/main/article/getBar', controller.getIndividualBarChartTop5);
router.get('/main/author', controller.showAuthorResult);
router.get('/main/author/showTimestamp', controller.showTimestampResult);
router.get('/logout', controller.logout);

module.exports = router;
```

File: app/controllers/analytics.server.controller.js

```
var express = require('express');
var Revision = require('../models/revision');
var Registration = require('../models/registration');
var https = require('https');
var fs = require('fs');
var pathModule = require('path');

var path = pathModule.join(process.cwd(),'app/import/dataset/');
var admin_active = fs.readFileSync(path + 'admin_active.txt').toString().split("\n");
var admin_former = fs.readFileSync(path + 'admin_former.txt').toString().split("\n");
var admin_inactive = fs.readFileSync(path + 'admin_inactive.txt').toString().split("\n");
var admin_semi_active = fs.readFileSync(path + 'admin_semi_active.txt').toString().split("\n");
var bot = fs.readFileSync(path + 'bot.txt').toString().split("\n");

var nodisplay = "display: none;";
var showdisplay = "display: block;";

//When logout
module.exports.logout = function(req, res) {
  console.log("logout function ran!");
  req.session.destroy();
  var loginerrormsg = "You have logged out from the system.";
  var errormsg = req.app.locals.errormsg;
  getIndex(req, res, loginerrormsg, errormsg, nodisplay, showdisplay);
}
```

```
//When login or register forms are submitted
module.exports.loginRegister = function(req, res) {
  console.log("loginRegister is ran!");
  var body = req.body;
  sess = req.session;
  console.log(sess);

  //If login form is posted
  if(body.formtype == "login"){
    Registration.loginCheck(body.loginemail, body.loginpassword, function(err,result){
      if(err){console.log(err);}
      else{
        if(result == 1){
          console.log("Logged in successfully!");
          sess.login = body.loginemail;
          getMain(req,res);
        }
        else{
          var loginerrormsg = "Email or Password is invalid. Please input again.";
          var errormsg = req.app.locals.errormsg;
          getIndex(req, res, loginerrormsg, errormsg, nodisplay, showdisplay);
        }
      }
    });
  }

  //If registration form is posted
  if(body.formtype == "registration"){
    Registration.userExist(body.email, function(err,result){
      if(err){console.log(err);}
      else{
        console.log(result);
        if(result > 0){
          var loginerrormsg = req.app.locals.loginerrormsg;
          var errormsg = "This email is already existed. Please input new email address or Login if you have
already registered";
          getIndex(req, res, loginerrormsg, errormsg, showdisplay, nodisplay);
        }
        else{
          Registration.addNewUser(body, function(err,result){
            if(err){console.log(err);}
            else{
              console.log("Registration Info Submitted Successfully!");
              sess.login = body.email;
              getMain(req,res);
            }
          });
        }
      }
    });
  }
}
```

```
//When GET '/', check if user is in session and render page
module.exports.showForm = function(req, res) {
  sess = req.session;
  console.log(sess);
  //If login session is still valid, enter main page
  if(sess && "login" in sess){
    console.log("user is still in session");
    getMain(req, res);
  }
  else{
    //If not, render login/registration page
    console.log("user not in session");
    var loginerrormsg = req.app.locals.loginerrormsg;
    var errormsg = req.app.locals.errormsg;
    getIndex(req, res, loginerrormsg, errormsg, nodisplay, showdisplay);
  }
};

//Render login/registration page
function getIndex(req, res, loginerrormsg, errormsg, registrationdisplay, logindisplay){
  console.log("get index ran!");
  var allResult = {};
  allResult.loginerrormsg = loginerrormsg;
  allResult.errormsg = errormsg;
  allResult.inputemail = req.app.locals.inputemail;
  allResult.inputfirstname = req.app.locals.inputfirstname;
  allResult.inputlastname = req.app.locals.inputlastname;
  res.render('index.ejs', {allResult:allResult, registrationdisplay:registrationdisplay,
logindisplay:logindisplay});
}

//When GET '/main', check if user is in session and render page
module.exports.showMain = function(req, res) {
  console.log("showMAIN ran");
  sess = req.session;
  console.log(sess);
  if(sess && "login" in sess){
    getMain(req, res);
  }
  else{
    console.log("Not in session");
    var loginerrormsg = req.app.locals.loginerrormsg;
    var errormsg = req.app.locals.errormsg;
    getIndex(req, res, loginerrormsg, errormsg, nodisplay, showdisplay);
  }
};

//Get query results and render main page
function getMain(req, res){
  console.log("getMain rannnn!!!!!!!");
  var allResult = {};
  var count = 0;

  sess = req.session;
```

```
allResult.email = sess.login;

//OVERALL Analytics
//Find 2 articles with most number of revisions
Revision.findMostNumRev(2, function(err, result){
  if(err){console.log(err);}
  else{
    allResult.mostNumRev1 = result[0]._id;
    allResult.mostNumRev2 = result[1]._id;
    count++;
    parseAllResult(allResult, res, count, 'main.ejs');
  }
});

//Find 2 articles with least number of revisions
Revision.findLeastNumRev(2, function(err, result){
  if(err){console.log(err);}
  else{
    allResult.leastNumRev1 = result[0]._id;
    allResult.leastNumRev2 = result[1]._id;
    count++;
    parseAllResult(allResult, res, count, 'main.ejs');
  }
});

//Find an article that are revised by largest group of registered users
Revision.findLargestRegisteredUsers(function(err, result){
  if(err){console.log(err);}
  else{
    allResult.largestRegUserRev = result[0]._id;
    count++;
    parseAllResult(allResult, res, count, 'main.ejs');
  }
});

//Find an article that are revised by smallest group of registered users
Revision.findSmallestRegisteredUsers(function(err, result){
  if(err){console.log(err);}
  else{
    allResult.smallestRegUserRev = result[0]._id;
    count++;
    parseAllResult(allResult, res, count, 'main.ejs');
  }
});

//Find an article with the oldest age
Revision.findLongestHistory(function(err, result){
  if(err){console.log(err);}
  else{
    allResult.longestHistory1 = result[0]._id;
    allResult.longestHistory2 = result[1]._id;
    count++;
    parseAllResult(allResult, res, count, 'main.ejs');
  }
```

```
  });

  //Find an article with the youngest age
  Revision.findYoungestHistory(function(err, result){
    if(err){console.log(err);}
    else{
      allResult.youngestHistory = result[0]._id;
      count++;
      parseAllResult(allResult, res, count, 'main.ejs');
    }
  });

  //Get the revision number distribution by year and by user type across the whole data set for Bar chart
  Revision.overallBarChart(function(err, result){
    if(err){console.log(err);}
    else{
      allResult.overallBarChart = result;
      count++;
      parseAllResult(allResult, res, count, 'main.ejs');
    }
  });

  //Get the revision number distribution by user type across the whole data set for Pie chart
  Revision.overallPieChart(function(err, result){
    if(err){console.log(err);}
    else{
      allResult.overallPieChart = result;
      count++;
      parseAllResult(allResult, res, count, 'main.ejs');
    }
  });

  //Get number of revisions by type of admin for Pie chart
  Revision.overallRevAdminType(function(err, result){
    if(err){console.log(err);}
    else{
      allResult.overallRevAdminType = result;
      count++;
      parseAllResult(allResult, res, count, 'main.ejs');
    }
  });

  //Get all the article names for drop down list
  Revision.articleDropDownList(function(err, result){
    if(err){console.log(err);}
    else{
      allResult.articleDropDownList = result;
      count++;
      parseAllResult(allResult, res, count, 'main.ejs');
    }
  });
}

//Sending Overall Data Analytics result back to browser: Before rendering, check that all the functions have
```

```
been run with count and then render to Main
function parseAllResult(allResult, res, count, viewfile){
  console.log(count);
  if(count < 10){return;}
  else{
    allResult.individualdisplay = nodisplay;
    allResult.authordisplay = nodisplay;
    res.render(viewfile, {allResult: allResult});
  }
}

//Get top # articles with highest/lowest revisions and render to highLowRevResult.ejs to be put in main.ejs
section
module.exports.getHighLowRev = function(req, res) {
  var allResult = {};
  allResult.mostNumRev = [];
  allResult.leastNumRev = [];
  var count = 0;
  var numberofarticle = parseInt(req.query.numberofarticle);
  //Get # articles with highest number of revision
  Revision.findMostNumRev(numberofarticle, function(err, result){
    if(err){console.log(err);}
    else{
      for(var i=0;i<numberofarticle;i++){
        allResult.mostNumRev[i] = result[i]._id;
      }
      allResult.numberOfArticle = numberofarticle;
      count++;
      parseHighLowRevResult(allResult, res, count, 'highLowRevResult.ejs');
    }
  });
  //Get # articles with lowest number of revision
  Revision.findLeastNumRev(numberofarticle, function(err, result){
    if(err){console.log(err);}
    else{
      for(var i=0;i<numberofarticle;i++){
        allResult.leastNumRev[i] = result[i]._id;
      }
      count++;
      parseHighLowRevResult(allResult, res, count, 'highLowRevResult.ejs');
    }
  });
};

//Sending results of # articles with highest/lowest revisions back as respond and render
function parseHighLowRevResult(allResult, res, count, viewfile){
  if(count < 2){return;}
  else{res.render(viewfile, {allResult: allResult});}
}

//Get Individual data analytics results and render to individualArticleResult.ejs
module.exports.showIndividualResult = function(req, res) {
  var wikiEndpointHost = "en.wikipedia.org",
  path = "/w/api.php"
```

```
parameters = [
  "action=query",
  "format=json",
  "prop=revisions",
  "rvdir=newer",
  "rvlimit=max",
  "rvprop=timestamp|userid|user|ids"],
headers = {
  Accept: 'application/json',
  'Accept-Charset': 'utf-8'
}
var options;

var allResult = {};
var title = req.query.title;
parameters.push("titles=" + encodeURIComponent(title));
allResult.individualTitle = title;
var latestRevTime;
var count = 0;

Revision.findTitleLatestRev(title, function(err, result){
  if(err){console.log(err);}
  else{
    latestRevTime = result[0].timestamp;
    parameters.push("rvstart=" + latestRevTime);
    var fullPath = path + "?" + parameters.join("&");
    options = {
        host: wikiEndpointHost,
        path: fullPath,
        headers: headers
    };

    //Compare timestamp if article is older than one day
    var currentDate = new Date();
    var articleLatestRev = new Date(latestRevTime);
    console.log("current: " + currentDate.toISOString() + " Articlate Rev: " + result[0].timestamp);
    var dateDiff = currentDate - articleLatestRev;
    var oneDay = 24 * 60 * 60 * 1000;
    //console.log(dateDiff + " compare " + oneDay);
    if(dateDiff < oneDay){
      //ALREADY UPDATED
      allResult.updateMsg = "The data is Up-to-Date!";
      count++;
      parseIndividualResult(allResult, res, count, 'individualArticleResult.ejs');
      getIndividualResult(title, res, allResult, count);

    }
    else{
      //NOT UPDATED
      console.log("not updated");

      //Request for revisions info from Wikipedia
      pullWikiData(options, function(revisions){
          allResult.updateMsg = "There are " + revisions.length + " new revisions. The articles data have been
```

```
updated!";
            count++;
            parseIndividualResult(allResult, res, count, 'individualArticleResult.ejs');

            //Set usertype, admintype, registered fields for updated users
            for(var i in revisions){
              revisions[i].title = title;
              //console.log(revisions[i]);
              if(revisions[i].hasOwnProperty("anon")){
                revisions[i].usertype = "anon";
                revisions[i].registered = false;
              }
              else if(admin_active.indexOf(revisions[i].user) > -1){
                revisions[i].usertype = "admin";
                revisions[i].admintype = "active";
                revisions[i].registered = true;
              }
              else if(admin_former.indexOf(revisions[i].user) > -1){
                revisions[i].usertype = "admin";
                revisions[i].admintype = "former";
                revisions[i].registered = true;
              }
              else if(admin_inactive.indexOf(revisions[i].user) > -1){
                revisions[i].usertype = "admin";
                revisions[i].admintype = "inactive";
                revisions[i].registered = true;
              }
              else if(admin_semi_active.indexOf(revisions[i].user) > -1){
                revisions[i].usertype = "admin";
                revisions[i].admintype = "semi-active";
                revisions[i].registered = true;
              }
              else if(bot.indexOf(revisions[i].user) > -1){
                revisions[i].usertype = "bot";
                revisions[i].registered = true;
              }
              else{
                revisions[i].usertype = "regular";
                revisions[i].registered = false;
              }
            }
            //Insert data to mongoDB
            Revision.addData(revisions, function(err, result){
              if(err){console.log(err);}
              else{
                getIndividualResult(title, res, allResult, count);
              }
            });
          });
        }
      }
    });
}
```

```
// Check all queries of individual analytics results are run before render
function parseIndividualResult(allResult, res, count, viewfile){
  console.log(count);
  if(count < 7){return;}
  else{
    res.render(viewfile, {allResult: allResult});
  }
}

//Get new revisions data from Wikipedia
pullWikiData = function(options, callback){
  console.log("pullwiki data function ran!!");
  https.get(options,function(res){
      var data ='';
      res.on('data',function(chunk){
          data += chunk;
      })
      res.on('end',function(){
          json = JSON.parse(data);
          pages = json.query.pages;
          revisions = pages[Object.keys(pages)[0]].revisions;
          revisions.splice(0, 1)
          //console.log(revisions);
          console.log("There are " + revisions.length + " new revisions.");
          var users=[]
          for (revid in revisions){
              users.push(revisions[revid].user);
          }
          uniqueUsers = new Set(users);
          console.log("The new revisions are made by " + uniqueUsers.size + " unique users");
          callback(revisions);
      })
  }).on('error',function(e){
      console.log(e);
  })
}

// Get all individual analytics query results
function getIndividualResult(title, res, allResult, count){
  //Get number of revisions of the article
  Revision.articleRevNumber(title, function(err, result){
    if(err){console.log(err);}
    else{
      allResult.articleRevNumber = result;
      count++;
      parseIndividualResult(allResult, res, count, 'individualArticleResult.ejs');
    }
  });

  //Get top 5 regular users that made most revisions on the article
  Revision.top5RegUsers(title, function(err, result){
    if(err){console.log(err);}
    else{
      allResult.top5users = [];
```

```
    //allResult.top5RegUsers1 = result[0];
    for(var i=0;i<result.length;i++){
      allResult.top5users[i] = result[i]._id;
    }
    count++;
    parseIndividualResult(allResult, res, count, 'individualArticleResult.ejs');
  }
});

//Get revision number distributed by year and by user type for this article for bar chart
Revision.articleBarChart(title, function(err, result){
  if(err){console.log(err);}
  else{
    allResult.articleBarChart = result;
    count++;
    parseIndividualResult(allResult, res, count, 'individualArticleResult.ejs');
  }
});

//Get revision number distribution based on user type for this article for pie chart
Revision.articlePieChart(title, function(err, result){
  if(err){console.log(err);}
  else{
    allResult.articlePieChart = result;
    count++;
    parseIndividualResult(allResult, res, count, 'individualArticleResult.ejs');
  }
});

//Get number of revisions by type of admin for Pie chart
Revision.articleAdminType(title, function(err, result){
  if(err){console.log(err);}
  else{
    allResult.articleAdminType = result;
    count++;
    parseIndividualResult(allResult, res, count, 'individualArticleResult.ejs');
  }
});

//Get year list for year range selection
Revision.articleYearList(title, function(err, result){
  if(err){console.log(err);}
  else{
    for(var i=0;i<result.length;i++){
      for(var j=0;j<result.length-i-1;j++){
        if(result[j]._id > result[j+1]._id){
          //console.log("swap" + result[j]._id + " " + result[j+1]._id);
          var temp = result[j]._id;
          result[j]._id = result[j+1]._id;
          result[j+1]._id = temp;
        }
      }
    }
    allResult.articleYearList = result;
```

```
      count++;
      parseIndividualResult(allResult, res, count, 'individualArticleResult.ejs');
    }
  });
}

//Get revision number distributed by year made by one or a few of the top 5 users for this article and render
module.exports.getIndividualBarChartTop5 = function(req, res) {
  var title = req.query.title;
  var topusers = req.query.topusers;
  var from = req.query.from;
  var yearfrom = (parseInt(from) - 1).toString();
  var to = req.query.to;
  var yearto = (parseInt(to) + 1).toString();
  //console.log("on server: " + topusers + " " + from + " " + to);
  var allResult = {};
  var usersArray = topusers.split(',');

  Revision.articleBarChartTop5(title, usersArray, yearfrom, yearto, function(err, result){
    if(err){console.log(err);}
    else{
      //console.log(result);
      allResult.usersArray = usersArray;
      allResult.articleBarChartTop5 = result;
      var arraytemp = [];
      for(var i=0;i<usersArray.length;i++){
        arraytemp.push(usersArray[i]);
      }

      for(var i=0;i<result.length;i++){
        for(var j=0;j<arraytemp.length;j++){
          if(result[i].user == arraytemp[j]){
            arraytemp.splice(j,1);
            j--;
          }
        }
      }
      if(arraytemp.length == 0){
        //console.log("user completed!");
        allResult.top5error = "";
      }
      else{
        //console.log("user not complete");
        allResult.top5error = "The graph could not be constructed because one or more authors do not have revision records in the year range selected. Please select a bigger range of years.";
      }

      res.render('finalBarChartTop5.ejs', {allResult: allResult});
    }
  });
}

//Get author analytics results and render
```

```
module.exports.showAuthorResult = function(req, res) {
  var user = req.query.user;
  allResult = {};
  //console.log("On server: " + user);

  Revision.authorAnalytics(user, function(err, result){
    if(err){console.log(err);}
    else{
      allResult.authorAnalytics = result;
      allResult.authorErrorMsg = "";
      if(result === undefined || result.length == 0){
        console.log("result is null");
        allResult.authorErrorMsg = "Author is not found. Please input the author's name again.";
      }
      res.render('authorAnalyticsResult.ejs', {allResult: allResult});
    }
  });
}

//Get list of timestamps of all revisions made by a user on an article and render
module.exports.showTimestampResult = function(req, res) {
  var user = req.query.user;
  var title = req.query.title;
  allResult = {};
  //console.log("On server: " + user + " " + title);

  Revision.getTimestamp(user, title, function(err, result){
    if(err){console.log(err);}
    else{
      allResult.timestampResult = result;
      res.render('timestampResult.ejs', {allResult: allResult});
    }
  });
}
```

File: app/import/importJSON.js

```
var exec = require("child_process").exec;
const fs = require("fs");
var pathModule = require('path');

var directoryPath = pathModule.join(process.cwd(),'dataset/revisions/');
var importcommand = 'mongoimport --jsonArray --db wikidata --collection revisions --file ' + directoryPath;

//check if directory exist
if(fs.existsSync(directoryPath)){
  console.log("Directory path existed!");
  //read files in directory
  fs.readdir(directoryPath, function(err, filenames){
    if(err){
      console.log(err);
```

```
    return;
   }
   filenames.forEach(function(filename){
    //var escapedfilename = filename.replace(/[^a-z0-9]/gi, '_').toLowerCase();
    //console.log(escapedfilename);
    fs.readFile(directoryPath + filename, 'utf-8', function(err, content){
     if(err){
      console.log(err);
      return;
     }
     console.log("Importing... " + filename);
     exec(importcommand + escapeFileName(filename));
    });
   });
  });
}
else{
  console.log("Directory not found");
}

function escapeFileName(filename){
   var name = "";
   for (var i=0;i<filename.length; i++) {
     if(filename[i].match(/[^a-z0-9.]/gi)){
      //console.log(filename[i]);
        name+="\\"+filename[i];
     }
     else{
        name+=filename[i];
     }
   }
   return name;
}
```

File: app/import/update.js

```
var mongoose = require('mongoose');
var fs = require('fs');
var express = require('express');
var pathModule = require('path');

var path = pathModule.join(process.cwd(),'dataset/');
var admin_active = fs.readFileSync(path + 'admin_active.txt').toString().split("\n");
var admin_former = fs.readFileSync(path + 'admin_former.txt').toString().split("\n");
var admin_inactive = fs.readFileSync(path + 'admin_inactive.txt').toString().split("\n");
var admin_semi_active = fs.readFileSync(path + 'admin_semi_active.txt').toString().split("\n");
var bot = fs.readFileSync(path + 'bot.txt').toString().split("\n");
var allAdmin = admin_active.concat(admin_former, admin_inactive, admin_semi_active);
var allRegisteredUser = allAdmin.concat(bot);
//console.log(allRegisteredUser);
```

```
mongoose.connect('mongodb://localhost/wikidata',function () {
            console.log('mongodb connected to wikidata!')
          });

  var RevisionSchema = new mongoose.Schema({
          title: String,
          timestamp:String,
          user:String,
          anon:String,
    usertype:String,
    registered:Boolean,
                    admintype:String
  },{
          versionKey: false
  });

var Revision = mongoose.model('Revision', RevisionSchema, 'revisions');

//Set usertype for all admin users
Revision.update(
  {'user':{'$in':allAdmin, '$nin':bot}},
  {$set:{usertype:'admin',registered:true}},
  {multi: true, upsert:true},
  function(err,result){
  if (err){console.log("ERROR")}
  else{
    console.log('admin usertype has been updated/inserted');
    console.log(result);
  }}
);

//Set type of Active admin for all admin_active users
Revision.update(
  {'user':{'$in':admin_active, '$nin':bot}},
  {$set:{admintype:'active'}},
  {multi: true, upsert:true},
  function(err,result){
  if (err){console.log("ERROR")}
  else{
    console.log('admin_active admintype has been updated/inserted');
    console.log(result);
  }}
);

//Set type of Former admin for all admin_former users
Revision.update(
  {'user':{'$in':admin_former, '$nin':bot}},
  {$set:{admintype:'former'}},
  {multi: true, upsert:true},
  function(err,result){
  if (err){console.log("ERROR")}
  else{
    console.log('admin_former admintype has been updated/inserted');
    console.log(result);
```

```
  }}
);

//Set type of Inactive admin for all admin_inactive users
Revision.update(
  {'user':{'$in':admin_inactive, '$nin':bot}},
  {$set:{admintype:'inactive'}},
  {multi: true, upsert:true},
  function(err,result){
  if (err){console.log("ERROR")}
  else{
    console.log('admin_inactive admintype has been updated/inserted');
    console.log(result);
  }}
);

//Set type of Semi-active admin for all admin_semi_active users
Revision.update(
  {'user':{'$in':admin_semi_active, '$nin':bot}},
  {$set:{admintype:'semi-active'}},
  {multi: true, upsert:true},
  function(err,result){
  if (err){console.log("ERROR")}
  else{
    console.log('admin_semi_active admintype has been updated/inserted');
    console.log(result);
  }}
);

//Update for all bot users with user type being bot
Revision.update(
  {'user':{'$in':bot}},
  {$set:{usertype:'bot',registered:true}},
  {multi: true, upsert:true},
  function(err,result){
  if (err){console.log("ERROR")}
  else{
    console.log('bot usertype has been updated/inserted');
    console.log(result);
  }}
);

//Update for all anonymous users with user type being anon
Revision.update(
  {'anon':{$exists:true}},
  {$set:{usertype:'anon',registered:false}},
  {multi: true, upsert:true},
  function(err,result){
  if (err){console.log("ERROR")}
  else{
    console.log('anon usertype has been updated/inserted');
    console.log(result);
  }}
);
```

```
//Update for all other users with user type being regular user
Revision.update(
  {'user':{'$nin':allRegisteredUser},'anon':{$exists:false}},
  {$set:{usertype:'regular',registered:true}},
  {multi: true, upsert:true},
  function(err,result){
  if (err){console.log("ERROR")}
  else{
    console.log('regular usertype has been updated/inserted');
    console.log(result);
  }}
);

//mongoose.disconnect();
```

File: app/models/registration.js

```
var mongoose = require('./db');
var fs = require('fs');

var RegistrationSchema = new mongoose.Schema({
        email: String,
        password:String,
        firstname:String,
        lastname:String
},{
        versionKey: false
});

// Insert new user
RegistrationSchema.statics.addNewUser = function(parameter, callback){
  return this.insertMany(parameter, function(err, result){
    if(err){console.log(err);}
    callback();
  });
}

// Check if email match with existing email
RegistrationSchema.statics.userExist = function(email, callback){
        return this.find({'email':email}).count().exec(callback)
}

// Check if email and password match with existing users
RegistrationSchema.statics.loginCheck = function(email, password, callback){
  //console.log(email + " " + password);
        return this.find({"email": email, "password": password}).count().exec(callback)
}

var Registration = mongoose.model('Registration', RegistrationSchema, 'registration')
```

```
module.exports = Registration
```

File: app/models/revision.js

```
var mongoose = require('./db');
var fs = require('fs');

var RevisionSchema = new mongoose.Schema({
        title: String,
        timestamp:String,
        user:String,
        anon:String,
        usertype:String,
        registered:Boolean,
        admintype:String
},{
        versionKey: false
});



//Overall #1 : Title of 2 articles with highest number of revisions
RevisionSchema.statics.findMostNumRev = function(number, callback){
        var query = [
                {$group: {_id:"$title", numOfRevisions:{$sum:1}}},
                {$sort: {numOfRevisions:-1}},
                {$limit: number}
        ]
        return this.aggregate(query)
        .exec(callback)
}

//Overall #2 : Title of 2 articles with lowest number of revisions
RevisionSchema.statics.findLeastNumRev = function(number, callback){
        var query = [
                {$group: {_id:"$title", numOfRevisions:{$sum:1}}},
                {$sort: {numOfRevisions:1}},
                {$limit: number}
        ]
        return this.aggregate(query)
        .exec(callback)
}

//Overall #4 : The article edited by largest group of registered users. Each wiki article is edited by a number of users, some making multiple revisions. The number of unique users is a good indicator of an article's popularity.
RevisionSchema.statics.findLargestRegisteredUsers = function(callback){
        var query = [
                {$match:{
        anon:{$exists:false},
        registered:{$exists:true}
```

```
  }},
                {$group:{_id:{title:"$title",user:"$user"}}},
                {$group:{_id:"$_id.title",count:{$sum:1}}},
                {$sort:{count:-1}},
        {$limit:1}
        ]
        return this.aggregate(query)
        .exec(callback)
}

//Overall #5 : The article edited by smallest group of registered users.
RevisionSchema.statics.findSmallestRegisteredUsers = function(callback){
        var query = [
                {$match:
                        {anon:{$exists:false},
        registered:{$exists:true}
                }},
                {$group:{_id:{title:"$title",user:"$user"}}},
                {$group:{_id:"$_id.title",count:{$sum:1}}},
                {$sort:{count:1}},
        {$limit:1}
        ]
        return this.aggregate(query)
        .exec(callback)
}


//Overall #6 : The top 2 articles with the longest history (measured by age)
RevisionSchema.statics.findLongestHistory = function(callback){
        var query = [
                {$group: {_id: "$title", timestamp: {"$min": "$timestamp"}}},
                {$sort: {timestamp: 1}},
          {$limit: 2}
        ]
        return this.aggregate(query)
        .exec(callback)
}

//Overall #7 : Article with the shortest history (measured by age)
RevisionSchema.statics.findYoungestHistory = function(callback){
        var query = [
                {$group: {_id: "$title", timestamp: {"$min": "$timestamp"}}},
                {$sort: {timestamp: -1}},
          {$limit: 1}
        ]
        return this.aggregate(query)
        .exec(callback)
}

//Overall #8 : Bar chart of revision number distribution by year and by user type across the whole data set
RevisionSchema.statics.overallBarChart = function(callback){
        var query = [
                {$project: {
                        year: {$substr: ["$timestamp", 0, 4]},
```

```
                        usertype: "$usertype"
                    }},
    {$group: {_id: {year:"$year", usertype:"$usertype"}, numbOfRev: {$sum: 1}}},
    {$project: {
                        _id: 0,
                        year: "$_id.year",
                        usertype:"$_id.usertype",
     numbOfRev: "$numbOfRev"
    }},
                    {$sort: {"year": 1}}
            ]
            return this.aggregate(query)
            .exec(callback)
}

//Overall #9 : Pie chart of revision number distribution by user type across the whole data set
RevisionSchema.statics.overallPieChart = function(callback){
            var query = [
                    {$group: {_id: {usertype:"$usertype"}, numbOfRev: {$sum: 1}}},
    {$project: {
                                _id: 0,
                                usertype:"$_id.usertype",
       numbOfRev: "$numbOfRev"
    }}
            ]
            return this.aggregate(query)
            .exec(callback)
}

//Overall #9 : Get number of revisions by type of admin
RevisionSchema.statics.overallRevAdminType = function(callback){
            var query = [
                    {$match:{usertype : "admin"}},
    {$group: {_id: "$admintype", numbOfRev: {$sum: 1}}},
    {$project: {
                                admintype:"$admintype",
       numbOfRev: "$numbOfRev"
    }}
            ]
            return this.aggregate(query)
            .exec(callback)
}

//Individual Article: Dropdown List
RevisionSchema.statics.articleDropDownList = function(callback){
            return this.distinct("title")
            .exec(callback)
}

//Individual Article: check latest revision time
RevisionSchema.statics.findTitleLatestRev = function(title, callback){
            return this.find({'title':title}).sort({'timestamp':-1}).limit(1).exec(callback)
}
```

```
//Individual Article #2: Total number of revisions
RevisionSchema.statics.articleRevNumber = function(title, callback){
	return this.find({'title':title}).count().exec(callback)
}

//Individual Article #3: Top 5 regular users ranked by total revision numbers on this article, and the
respective revision numbers
RevisionSchema.statics.top5RegUsers = function(title, callback){
	var query = [
		{$match: {"title": title}},
		{$group: {_id: "$user", numbOfRev:{$sum:1}}},
		{$sort: {numbOfRev:-1}},
		{$limit: 5}
	]
	return this.aggregate(query)
	.exec(callback)
}

//Update data after calling API
RevisionSchema.statics.addData = function(wikiData, callback){
	var empty = true;
	for(var i in wikiData){
		if(wikiData.hasOwnProperty(i)){
			empty = false;
		}
	}
	if(!empty){
		console.log(wikiData);
		return this.insertMany(wikiData, function(err, result){
			if(err){console.log(err);}
			//console.log(result);
			callback();
		});
	}
	else{
		return callback()
	}
}

//Individual Article #4: Bar chart of revision number distributed by year and by user type for this article
RevisionSchema.statics.articleBarChart = function(title, callback){
	var query = [
		{$match: {"title": title}},
		{$project: {
			year: {$substr: ["$timestamp", 0, 4]},
			usertype: "$usertype"
		}},
   {$group: {_id: {year:"$year", usertype:"$usertype"}, numbOfRev: {$sum: 1}}},
   {$project: {
			_id: 0,
			year: "$_id.year",
			usertype:"$_id.usertype",
    numbOfRev: "$numbOfRev"
   }},
```

```
                {$sort: {"year": 1}}
        ]
        return this.aggregate(query)
        .exec(callback)
}

//Individual Article #5: Pie chart of revision number distribution based on user type for this article
RevisionSchema.statics.articlePieChart = function(title, callback){
        var query = [
                {$match: {"title": title}},
                {$group: {_id: {usertype:"$usertype"}, numbOfRev: {$sum: 1}}},
   {$project: {
                                _id: 0,
                                usertype:"$_id.usertype",
      numbOfRev: "$numbOfRev"
   }}
        ]
        return this.aggregate(query)
        .exec(callback)
}

//Individual Article #5: Get number of revisions of this article by type of admin
RevisionSchema.statics.articleAdminType = function(title, callback){
        var query = [
                {$match:{"title": title, usertype : "admin"}},
   {$group: {_id: "$admintype", numbOfRev: {$sum: 1}}},
   {$project: {
                                admintype:"$admintype",
      numbOfRev: "$numbOfRev"
   }}
        ]
        return this.aggregate(query)
        .exec(callback)
}

//Individual Article #6: Bar chart of revision number distributed by year made by one or a few of the top 5 users for this article
RevisionSchema.statics.articleBarChartTop5 = function(title, usersArray, from, to, callback){
        console.log(from + " " + to);
        var query = [
                {$match: {"title": title, "user": {'$in': usersArray}}},
                {$project: {
                        year: {$substr: ["$timestamp", 0, 4]},
    user: "$user"
                }},
                {$match: {year: {$gt:from, $lt:to}}},
                {$group: {_id: {year:"$year", user:"$user"}, numbOfRev: {$sum:1}}},
   {$project:{
    _id:0,
    year: "$_id.year",
    user: "$_id.user",
    numbOfRev: "$numbOfRev"
    }},
                {$sort: {"year": 1}}
```

```
        ]
        return this.aggregate(query)
        .exec(callback)
}

//Individual Article #6: Get year list for year range input
RevisionSchema.statics.articleYearList = function(title, callback){
        var query = [
                {$match: {"title": title}},
   {$project: {year: {$substr: ["$timestamp", 0, 4]}}},
   {$group: {_id: "$year"}},
   {$sort: {"year": 1}}
        ]
        return this.aggregate(query)
        .exec(callback)
}

//Author Analytics: Get article list and number of revision by user
RevisionSchema.statics.authorAnalytics = function(user, callback){
        var query = [
                {$match: {"user": user}},
                {$group: {_id: "$title", numbOfRev: {$sum:1}}},
   {$project: {
        title: "$_id.title",
        numbOfRev: "$numbOfRev"}},
                {$sort: {"_id": 1}}
        ]
        return this.aggregate(query)
        .exec(callback)
}

//Author Analytics: Get timestamp list of an article by user
RevisionSchema.statics.getTimestamp = function(user, title, callback){
        var query = [
                {$match: {"user": user, "title": title}},
   {$project: {
     _id: 0,
     title: "$title",
     timestamp: "$timestamp"}},
                {$sort: {"_id": 1}}
        ]
        return this.aggregate(query)
        .exec(callback)
}

var Revision = mongoose.model('Revision', RevisionSchema, 'revisions')

module.exports = Revision
```

Below are the codes in the app/views

File: app/views/authorAnalyticsResult.ejs

```
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="utf-8">
  <title>Author Analytics Result</title>
  <link rel="stylesheet" href="/css/style.css" />
  <script type="text/javascript" src="/js/author.js"></script>
</head>
<body>
 <div class="subSection">
  <div id="authorerror"><%= allResult.authorErrorMsg %></div>
  <table id="author-table">
   <thead>
    <tr>
     <th>Article's Title</th>
     <th>Number of Revisions</th>
    </tr>
   </thead>
   <tbody>
    <% if (locals.allResult.authorAnalytics) {for(var i=0; i < allResult.authorAnalytics.length; i++) { %>
    <tr>
     <td><%= allResult.authorAnalytics[i]._id %></td>
     <td><%= allResult.authorAnalytics[i].numbOfRev %></td>
    </tr>
    <% }} %>
   </tbody>

  </table>
 </div>

 <div class="subSection timestampsection">
  <label id="timestamplabel">Check Revisions' Timestamps</label><br>
  <div class="select-style">
   <select id="selectAuthorArticle" name="authorarticle">
    <option>Choose an Article</option>
    <% for(var i = 0; i < allResult.authorAnalytics.length;i++){ %>
    <option
value="<%=allResult.authorAnalytics[i]._id%>"><%=allResult.authorAnalytics[i]._id%></option>
    <% } %>
   </select>
   <i class="fa fa-caret-down graphicon" aria-hidden="true"></i>
  </div>
  <input class="select-btn" id="selectAuthorArticleSubmit" type="submit" value="Submit">

  <div id="timestampResult"></div>
 </div>

</body>
</html>
```

File: app/views/finalBarChartTop5.ejs

```
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="utf-8">
  <title>Individual Article Bar Chart Result</title>
  <link rel="stylesheet" href="/css/style.css" />
  <script type="text/javascript" src="/js/articletop5.js"></script>
</head>
<body>
 <script type="text/javascript">
   var articleBarChartTop5 = <%- JSON.stringify(allResult.articleBarChartTop5) %>;
   var usersArray = <%- JSON.stringify(allResult.usersArray) %>;
   //console.log(articleBarChartTop5);
 </script>

<p id="top5error"><%=allResult.top5error%></p>
<div id="articleBarChartTop5Section"></div>

</body>
</html>
```

File: app/views/hightLowRevResult.ejs

```
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="utf-8">
  <title>Highest Number of Revisions</title>
  <link rel="stylesheet" href="/css/style.css" />
</head>
<body>
 <p>Titles of <%= allResult.numberOfArticle %> articles with highest number of revisions :
   <div class="result">&#8594;
   <% if (locals.allResult.mostNumRev) {for(var i=0; i < allResult.mostNumRev.length; i++) { %>
      <%= allResult.mostNumRev[i] %>,
   <% }} %>
 </div>
 </p>
 <p>Titles of <%= allResult.numberOfArticle %> articles with lowest number of revisions :
   <div class="result">&#8594;
   <% if (locals.allResult.leastNumRev) {for(var i=0; i < allResult.leastNumRev.length; i++) { %>
      <%= allResult.leastNumRev[i] %>,
   <% }} %>
 </div>
 </p>

</body>
</html>
```

File: app/views/index.ejs

```
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="utf-8">
  <title>COMP5347 Group Project 6</title>
  <link rel="stylesheet" href="/css/style.css" />
  <script src="https://code.jquery.com/jquery-3.2.1.js"></script>
  <script type="text/javascript" src="/js/index.js"></script>


</head>
<body id="index">
 <div id="loginform" class="loginform form" style="<%=logindisplay%>">
  <form method="POST" action="main">
    <p id="loginerrorresult"><%=allResult.loginerrormsg%></p>
    <input type="email" id="loginemail" name="loginemail" class="loginrequired"
placeholder="Email"><br>
    <input type="password" id="loginpassword" name="loginpassword" class="loginrequired"
placeholder="Password"><br>
    <input name="formtype" value="login" style="display:none">
    <input class="submit-btn" type="submit" value="LOGIN">
    <p id="registerlink" onclick="register()"><span style="color:#515151">Or new user?</span> Create
new Account</p><br><br>
  </form>
 </div>


 <div id="registrationform" class="registrationform form" style="<%=registrationdisplay%>">
  <form method="POST" action="main">
    <p id="errorresult"><%=allResult.errormsg%></p>
    <input type="email" id="email" name="email" class="required" value="<%=allResult.inputemail%>"
placeholder="Email"><br>
    <input type="password" id="password" name="password" class="required"
placeholder="Password"><br>
    <input type="password" id="repassword" name="repassword" class="required"
placeholder="Re-password"><br>
    <input type="text" id="firstname" name="firstname" class="required"
value="<%=allResult.inputfirstname%>" placeholder="First Name"><br>
    <input type="text" id="lastname" name="lastname" class="required"
value="<%=allResult.inputlastname%>" placeholder="Last Name"><br>
    <input name="formtype" value="registration" style="display:none">
    <input class="submit-btn" type="submit" value="REGISTER">
    <button id="loginlink" type="button" onclick="login()">&#8592; Back to Login</button><br>
  </form>
 </div>
</body>
</html>
```

File: app/views/individualArticleResult.ejs

```
<!DOCTYPE html>
```

```
<html>
<head lang="en">
  <meta charset="utf-8">
  <title>Individual Article Result</title>
  <link rel="stylesheet" href="/css/style.css" />
  <script type="text/javascript" src="/js/individual.js"></script>
</head>
<body>
 <script type="text/javascript">
  var articleTitle = <%- JSON.stringify(allResult.articleBarChart) %>;
  var articleBarChart = <%- JSON.stringify(allResult.articleBarChart) %>;
  var articlePieChart = <%- JSON.stringify(allResult.articlePieChart) %>;
  var articleAdminType = <%- JSON.stringify(allResult.articleAdminType) %>;
 </script>

 <p>Title : <span class="result"><%= allResult.individualTitle %></span></p>
 <p>Message : <span class="result"><%= allResult.updateMsg %></span></p>
 <p>Number of revisions : <span class="result"><%= allResult.articleRevNumber %></span></p>
 <p>Top 5 regular users :
  <span class="result">
  <% if (locals.allResult.top5users) {for(var i=0; i < allResult.top5users.length; i++) { %>
     <%= allResult.top5users[i] %>,
  <% }} %>
 </span>
 </p>

 <div class="slidemenu2">

  <!-- Item 3 -->
  <input type="radio" name="slideItem2" id="slide-item-3" class="slide-toggle" checked />
  <label for="slide-item-3">
   <i class="fas fa-chart-bar fa-2x graphicon"></i><span>Bar Chart #1</span>
  </label>
  <!-- Item 4 -->
  <input type="radio" name="slideItem2" id="slide-item-4" class="slide-toggle" />
  <label for="slide-item-4">
   <i class="fas fa-chart-pie fa-2x graphicon"></i><span>Pie Chart</span>
  </label>
  <!-- Item 5 -->
  <input type="radio" name="slideItem2" id="slide-item-5" class="slide-toggle" />
  <label for="slide-item-5">
   <i class="fas fa-chart-bar fa-2x graphicon"></i><span>Bar Chart #2</span>
  </label>
  <div class="clear"></div>
  <!-- Bar -->
  <div class="slider">
   <div class="bar2"></div>
  </div>

 </div>

 <div id="articleBarChartSec"></div>
 <div id="articlePieChartSec"></div>
 <div id="articleBarChartSec2" style="display:none">
```

```
  <div class="col1">
    <div id="topusererror"></div>
    <label>Choose a user or multiple users</label>
    <select class="multiselect" id="selectUser" name="user" multiple>
      <% for(var i = 0; i < allResult.top5users.length;i++){ %>
      <option value="<%=allResult.top5users[i]%>"><%=allResult.top5users[i]%></option>
      <% } %>
    </select>
  </div>

  <div class="col2">
    <div class="select-style yearrange">
      <select id="selectMinYear" name="minyear">
        <option>From</option>
        <% for(var i = 0; i < allResult.articleYearList.length;i++){ %>
        <option
value="<%=allResult.articleYearList[i]._id%>"><%=allResult.articleYearList[i]._id%></option>
        <% } %>
      </select>
      <i class="fa fa-caret-down graphicon" aria-hidden="true"></i>
    </div>

    <div class="select-style yearrange">
      <select id="selectMaxYear" name="maxyear">
        <option>To</option>
        <% for(var i = 0; i < allResult.articleYearList.length;i++){ %>
        <option
value="<%=allResult.articleYearList[i]._id%>"><%=allResult.articleYearList[i]._id%></option>
        <% } %>
      </select>
      <i class="fa fa-caret-down graphicon" aria-hidden="true"></i>
    </div>

    <input class="select-btn" id="selectUserYrSubmit" type="submit" value="Submit">
  </div>


  <div id="individualTitleTop5BarChart"></div>
  <!--<div id="articleBarChartTop5Sec"></div>-->
 </div>

</body>
</html>
```

File: app/views/main.ejs

```
<!DOCTYPE html>
<html>

<head lang="en">
 <meta charset="utf-8">
```

```
  <title>COMP5347 Group Project 6</title>
  <link rel="stylesheet" href="/css/style.css" />
  <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.8.2/css/solid.css"
integrity="sha384-ioUrHig76ITq4aEJ67dHzTvqjsAP/7IzgwE7lgJcg2r7BRNGYSK0LwSmROzYtgzs"
crossorigin="anonymous">
<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.8.2/css/fontawesome.css"
integrity="sha384-sri+NftO+0hcisDKgr287Y/1LVnInHJ1l+XC7+FOabmTTIK0HnE2ID+xxvJ21c5J"
crossorigin="anonymous">

  <script type="text/javascript" src="//cdnjs.cloudflare.com/ajax/libs/jquery/2.0.3/jquery.min.js"></script>
  <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
  <script type="text/javascript" src="/js/main.js"></script>

</head>

<body id="main">
  <script type="text/javascript">
    var overallBarChart = <%- JSON.stringify(allResult.overallBarChart) %>;
    var overallPieChart = <%- JSON.stringify(allResult.overallPieChart) %>;
    var overallRevAdminType = <%- JSON.stringify(allResult.overallRevAdminType) %>;
  </script>

  <div class="container">
    <div id="contentLeft">
      <ul id="leftNavigation">
        <li id="welcomeSec">Welcome,<br> <%= allResult.email%><br><i class="fas fa-smile fa-3x
smileicon"></i></li>
        <li id="overallMenu"> <a href="#"><i class="fas fa-chart-bar leftNavIcon"></i> Overall Analytics
</a></li>
        <li id="individualMenu"> <a href="#"><i class="fas fa-newspaper leftNavIcon"></i> Individual
Analytics </a></li>
        <li id="authorMenu"> <a href="#"><i class="fas fa-user-edit leftNavIcon"></i> Author Analytics
</a></li>
        <li id="logout"> <a href="#"><span style="margin-left: 40px;"> LOGOUT </span></a></li>
      </ul>
    </div>

    <div id="contentRight">
      <div id="overallSection">
        <h2>Overall Analytics</h2>
        <div id="numbererror"></div>
        <div class="subSection">
          <div class="wrap">
            <div class="search">
              <input class="searchTerm" id="number" type="text" name="title" placeholder="Number of
Article">
              <button class="searchButton" id="numOfArticleSubmit" type="submit">
                <i class="fa fa-search"></i>
              </button>
            </div>
          </div>

          <div id="numHighLowResult">
            <p>Titles of 2 articles with highest number of revisions :<div class="result">&#8594; <%=
```

```
allResult.mostNumRev1%>, <%= allResult.mostNumRev2%></div></p>
        <p>Titles of 2 articles with lowest number of revisions : <div class="result">&#8594; <%=
allResult.leastNumRev1%>, <%= allResult.leastNumRev2%></div></p>
      </div>
    </div>

    <div class="subSection">
      <p>The article edited by largest group of registered users : <div class="result">&#8594; <%=
allResult.largestRegUserRev%></div></p>
      <p>The article edited by smallest group of registered users : <div class="result">&#8594; <%=
allResult.smallestRegUserRev%></div></p>
    </div>

    <div class="subSection">
      <p>The top 2 articles with the longest history (measured by age) : <div class="result">&#8594; <%=
allResult.longestHistory1%>, <%= allResult.longestHistory2%></div></p>
      <p>Article with the shortest history (measured by age) : <div class="result">&#8594; <%=
allResult.youngestHistory%></div></p>
    </div>

    <div class="subSection">
      <div class="slidemenu">
        <!-- Item 1 -->
        <input type="radio" name="slideItem" id="slide-item-1" class="slide-toggle" checked />
        <label for="slide-item-1">
          <i class="fas fa-chart-bar fa-2x graphicon"></i><span>Bar Chart</span>
        </label>
        <!-- Item 2 -->
        <input type="radio" name="slideItem" id="slide-item-2" class="slide-toggle" />
        <label for="slide-item-2">
          <i class="fas fa-chart-pie fa-2x graphicon"></i><span>Pie Chart</span>
        </label>
        <div class="clear"></div>
        <!-- Bar -->
        <div class="slider">
          <div class="bar"></div>
        </div>
      </div>
      <div class="graphwrapper">
        <div id="overallBarChartSec"></div>
        <div id="overallPieChartSec"></div>
      </div>
    </div>
  </div>

  <div id="individualSection" style="<%=allResult.individualdisplay%>">
    <h2>Individual Article Analytics</h2>
    <div class="select-style">
      <select id="selectArticle" name="article">
        <option>Choose an Article</option>
        <% for(var i = 0; i < allResult.articleDropDownList.length;i++){ %>
        <option
value="<%=allResult.articleDropDownList[i]%>"><%=allResult.articleDropDownList[i]%></option>
        <% } %>
```

```
        </select>
        <i class="fa fa-caret-down graphicon" aria-hidden="true"></i>
      </div>
      <input class="select-btn" id="selectArticleSubmit" type="submit" value="Select">

      <div id="individualTitle">
        <input id="selectUserYrSubmit" type="submit" value="Submit" style="display:none;">
        <p id="errormsg" style="display:none;"></p>
      </div>
      <div id="individualTitleTop5BarChart"></div>
    </div>


    <div id="authorSection" style="<%=allResult.authordisplay%>">
      <h2>Author Analytics</h2>
      <div class="wrap">
        <div class="search">
          <input class="searchTerm" id="selectAuthor" type="text" placeholder="Author's name">
          <button class="searchButton" id="selectAuthorSubmit" type="submit" value="Submit">
            <i class="fa fa-search"></i>
          </button>
        </div>
      </div>

      <div id="authorAnalyticsResult">
        <input id="selectAuthorArticleSubmit" type="submit" value="Submit" style="display:none;">
      </div>
    </div>
  </div>
 </div>


</body>
</html>
```

File: app/views/timeStampResult.ejs

```
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="utf-8">
  <title>Timestamps Result</title>
  <link rel="stylesheet" href="/css/style.css" />
</head>
<body>
 <table id="timestamp-table">
   <thead>
     <tr>
       <th>#</th>
       <th>Timestamps</th>
     </tr>
   </thead>
```

```
  <tbody>
   <% if (locals.allResult.timestampResult) {for(var i=0; i < allResult.timestampResult.length; i++) { %>
    <tr>
     <td><%= i+1 %></td>
     <td><%= allResult.timestampResult[i].timestamp %></td>
    </tr>
   <% }} %>
  </tbody>

 </table>

</body>
</html>
```

Below are codes in public/js

File: public/js/articleTop5.js

```
google.charts.load('current', {packages: ['corechart']});
google.charts.setOnLoadCallback(drawArticleBarChartTop5Users);

function drawArticleBarChartTop5Users(){
 var array = [['Year', 'Number of revisions']];
 if (typeof articleBarChartTop5 === 'undefined'){
  //console.log("do nothing");
 }
 else{
  var jsondata = {};
  var arr = [];
  arr.push('Year');
  usersArray.forEach(function(field){
   arr.push(field);
  });

  articleBarChartTop5.forEach(function(field){
   if(!jsondata[field.year]){
    jsondata[field.year] = {};
   }
   //console.log(field.year + " " + field.user + " " + field.numbOfRev);
   jsondata[field.year][field.user] = parseInt(field.numbOfRev);
  })

  var bar = [];
  bar.push(arr);
  //console.log(bar);
  for(var row in jsondata){
   var barRow = [row];
   for(var i=0;i<usersArray.length;i++){
    barRow.push(jsondata[row][usersArray[i]]);
   }
   bar.push(barRow);
```

```
    }

    var data = google.visualization.arrayToDataTable(bar);

    var options = {
      'width': 900,
      'height': 500,
      colors: ['#bf5f5f', '#ffafaf', '#ff9b41', '#bb245f', '#7e5dc3'],
      'title': "Revision number distributed by year made by one or a few of the top 5 users for this article",
      hAxis: {
        title: 'timestamp'
      }
    };
    var chart = new google.visualization.ColumnChart($("#articleBarChartTop5Section")[0]);
    chart.draw(data, options);
  }
}
```

File: public/js/author.js

```
$(document).ready(function(){
    //When submitted article's title of an author, respond with list of revisions' timestamps
    $('#selectAuthorArticleSubmit').on('click', function(e){
      var author = $('#selectAuthor').val();
      var encodedAuthor = encodeURIComponent(author);
      var title = $('#selectAuthorArticle').val();
      var encodedTitle = encodeURIComponent(title);
      console.log(title);
      $.get("/main/author/showTimestamp?user=" + encodedAuthor + "&title=" + encodedTitle,
function(result) {
        //console.log(result);
        $('#timestampResult').html(result)
      });
    });
});
```

File: public/js/index.js

```
//Hide login form and show registration form
function register(){
  document.getElementById("registrationform").style.display = "block";
  document.getElementById("loginform").style.display = "none";
}

//Hide registration form and show login form
function login(){
  document.getElementById("registrationform").style.display = "none";
  document.getElementById("loginform").style.display = "block";
}


//function to determine if a field is blank
```

```
function isBlank(inputField){
  if(inputField.type=="checkbox"){
              if(inputField.checked){return false;}
               return true;
    }
    if (inputField.value==""){return true;}
  return false;
}

//Validate Email
function validateEmail(email){
  var email = document.getElementById(email);
  var emailregex = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/;
  if(email.value.match(emailregex)){
    return true;
  }
  else{
    return false;
  }
}

//Validate for strong password: must have at least 8 characters, contain at least 1 upper and 1 lower case
letter and 1 number
function validatePassword(){
  var password = document.getElementById("password");
  var pwregex = new RegExp("^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.*[!@#\$%\^&\*])(?=.{8,})");
  if(password.value.match(pwregex)){
    return true;
  }
  else{
    return false;
  }
}

//check if password match
function passwordMatch(){
  var password = document.getElementById("password");
  var repassword = document.getElementById("repassword");
  if(password.value == repassword.value){
    return true;
  }
  else{
    return false;
  }
}

//check if name contains special character
function checkName(){
  var firstname = document.getElementById("firstname");
  var lastname = document.getElementById("lastname");
  //letter only and should be between 2-30 characters
  var letteronlyreg = /^[a-zA-Z ]{2,30}$/;
  if(!firstname.value.match(letteronlyreg)){
    errorresult.innerHTML = "Invalid First name. First Name must not contain any special character or
```

```
number and should be from 2-30 characters.";
    return false;
  }
  else if(!lastname.value.match(letteronlyreg)){
    errorresult.innerHTML = "Invalid Last name. Last Name must not contain any special character or number
and should be from 2-30 characters.";
    return false;
  }
  else{
    return true;
  }
}

//Check all validations
function runValidation(){
  if(!validateEmail("email")){
    errorresult.innerHTML = "Invalid email. Please input email in the correct format.";
    return false;
  }
  else if(!validatePassword()){
    errorresult.innerHTML = "Password is not strong enough. Password must be 8 characters or longer, and
must contain at least 1 uppercase letter, 1 lowercase letter, 1 numeric character, and 1 special character.";
    //alert("Password is not strong enough. Password must be 8 characters or longer, and must contain at
least 1 uppercase letter, 1 lowercase letter, 1 numeric character, and 1 special character.");
    return false;
  }
  else if(!passwordMatch()){
    errorresult.innerHTML = "Password does not match. Plese input the password again.";
    return false;
  }
  else if(!checkName()){
    return false;
  }
  else{
    return true;
  }
}

$(document).ready(function(){
  var loginform = document.getElementById("loginform");
  var loginRequiredInputs = document.querySelectorAll(".loginrequired");
  var registrationform = document.getElementById("registrationform");
  var requiredInputs = document.querySelectorAll(".required");

  //When login form is submitted
  loginform.onsubmit = function(e){
    console.log("Login submitted on frontend");
    for (var i=0; i < loginRequiredInputs.length; i++){
      if(isBlank(loginRequiredInputs[i])){
        e.preventDefault();
        loginerrorresult.innerHTML = "Please input all the required fields";
      }
      else if(!validateEmail("loginemail")){
        e.preventDefault();
```

```
      loginerrorresult.innerHTML = "Invalid email. Please input email in the correct format.";
      return false;
    }
          else{
      loginerrorresult.innerHTML = "";
      //loginerrorresult.style.color = "#4e9947";
      console.log("pass!");
          }
  }

 }

 //When registration form is submitted
 registrationform.onsubmit = function(e){
  //var requiredInputs = document.querySelectorAll(".required");
  for (var i=0; i < requiredInputs.length; i++){
   if(isBlank(requiredInputs[i])){
    e.preventDefault();
    errorresult.innerHTML = "Please input all the required fields";
   }
   //if not pass validation
   else if(!runValidation()){
    e.preventDefault();
   }
          else{
    errorresult.innerHTML = "";
    //errorresult.style.color = "#4e9947";
    console.log("pass!");
          }
  }
 }

});
```

File: public/js/individual.js

```
google.charts.load('current', {packages: ['corechart']});
google.charts.setOnLoadCallback(drawArticleBarChart);
google.charts.setOnLoadCallback(drawArticlePieChart);

$(document).ready(function(){
 //Display individual bar chart #1
 $('#slide-item-3').on('click', function(e){
  $('#articleBarChartSec').css("display", "block");
  $('#articlePieChartSec').css("display", "none");
  $('#articleBarChartSec2').css("display", "none");
 });

 //Display individual pie chart
 $('#slide-item-4').on('click', function(e){
  $('#articleBarChartSec').css("display", "none");
  $('#articlePieChartSec').css("display", "block");
  $('#articleBarChartSec2').css("display", "none");
```

```
  });

  //Display individual bar chart 2
  $('#slide-item-5').on('click', function(e){
    $('#articleBarChartSec').css("display", "none");
    $('#articlePieChartSec').css("display", "none");
    $('#articleBarChartSec2').css("display", "block");
  });

    //When submitted user lists and year range of the article, respond with bar chart
    $('#selectUserYrSubmit').on('click', function(e){
      var article = $('#selectArticle').val();
      var topusers = $('#selectUser').val();
      var from = $('#selectMinYear').val();
      var to = $('#selectMaxYear').val();
      if(from >= to){
        console.log("invalid range");
        $('#topusererror').html("Invalid year range. Please select the year range again.")
        e.preventDefault();
      }
      else if(topusers == null || from == "From" || to == "To"){
        console.log("not all selected");
        $('#topusererror').html("Some fields are not selected. Please select all fields.")
        e.preventDefault();
      }
      else{
        var encodedArticle = encodeURIComponent(article);
        var encodedUser = encodeURIComponent(topusers);
        var route = "/main/article/getBar?title=" + encodedArticle + "&topusers=" + encodedUser + "&from="
+ from + "&to=" + to;
        console.log(from + " " + to + " " + topusers);
        //console.log(typeof(user));
        $.get(route, function(result) {
          console.log(result);
          $('#topusererror').html("");
          $('#individualTitleTop5BarChart').html(result)
        });
      }
    });
});


function drawArticleBarChart(){
  var jsondata = {};
  if (typeof articleBarChart === 'undefined'){
    //console.log("do nothing");
  }
  else{
    articleBarChart.forEach(function(field){
      if(!jsondata[field.year]){
        jsondata[field.year] = {};
      }
      jsondata[field.year][field.usertype] = parseInt(field.numbOfRev);
    })
```

```
    var bar = [['Year', 'Administrator', 'Anonymous', 'Bot', 'Regular']];
    for(var row in jsondata){
      var barRow = [row];
      barRow.push(jsondata[row].admin);
      barRow.push(jsondata[row].anon);
      barRow.push(jsondata[row].bot);
      barRow.push(jsondata[row].regular);
      bar.push(barRow);
    }
    var data = google.visualization.arrayToDataTable(bar);
    var options = {
        'title':"Revision number distributed by year and by user type of the article: " + $('#selectArticle').val(),
        'width':900,
        'height':500,
        colors: ['#bf5f5f', '#ffafaf', '#ff9b41', '#bb245f']
    };
    var chart = new google.visualization.ColumnChart($("#articleBarChartSec")[0]);
    chart.draw(data, options);
  }
}

function drawArticlePieChart(){
  if (typeof articlePieChart != 'undefined'){
    var usertype = [['user type', 'number of revision']];
    var pieindex;
    var i = 0;
    articlePieChart.forEach(function(field){
      var piesection = [field.usertype, field.numbOfRev];
      usertype.push(piesection);
      if(field.usertype == "admin"){pieindex = i;}
      i++;
    })
    var str = "";
    var tooltip = [];
    articleAdminType.forEach(function(field){
      str = str + field._id + " " + field.numbOfRev + "<br>";
    })
    tooltip.push(str);
    var data = google.visualization.arrayToDataTable(usertype);
    var options = {
        'title':"Revision number distribution by user type of the article: " + $('#selectArticle').val(),
        'width':900,
        'height':550,
        colors: ['#bf5f5f', '#ee9c9c', '#f2bc8b', '#de7e34'],
        tooltip: { isHtml: true }
    };
    var chart = new google.visualization.PieChart($("#articlePieChartSec")[0]);

    var sliceid = 0;
    function eventHandler(e){
      chart.setSelection([e]);
      try {
        selection = chart.getSelection();
```

```
      sliceid = selection[0].row - pieindex;
    }
    catch(err) {
      ;
    }
    //$(".google-visualization-tooltip-item-list li:eq(0)").css("font-weight", "bold");
    $(".google-visualization-tooltip-item-list li:eq(1)").html(tooltip[sliceid]).css("font-family", "Arial");
  }
  google.visualization.events.addListener(chart, 'onmouseover', eventHandler);
  var container = document.getElementById('articlePieChartSec');
  google.visualization.events.addListener(chart, 'ready', function () {
    container.style.display = 'none';
  });
  chart.draw(data, options);
 }
}
```

File: public/js/main.js

```
google.charts.load('current', {packages: ['corechart']});
//google.load("visualization", "1", {packages:["corechart"]});
google.charts.setOnLoadCallback(drawOverallBarChart);
google.charts.setOnLoadCallback(drawOverallPieChart);

$(document).ready(function(){
  //When submitted number of article, respond with number of least/most revisions for overall data
  $('#numOfArticleSubmit').on('click', function(e){
    var getnumber = $('#number').val();
    console.log(parseInt(getnumber));
    if(isNaN(parseInt(getnumber))){
      console.log("is not number");
      $('#numbererror').html("Invalid input. Please input number.")
      e.preventDefault();
    }
    else{
      var parameters = {numberofarticle: $('#number').val() };
      $.get('/main/getHighLowRev', parameters, function(result) {
        $('#numbererror').html("");
        $('#numHighLowResult').html(result);
      });
    }
  });

  //When submitted article title, respond with that article's data analytics
  $('#selectArticleSubmit').on('click', function(e){
    var article = $('#selectArticle').val();
    console.log(article);
    var encodedArticle = encodeURIComponent(article);
    $.get("/main/article?title=" + encodedArticle, function(result) {
      //console.log(result);
      $('#individualTitle').html(result);
    });
  });
```

```
  //When submitted author's name, respond with that author's data analytics result
  $('#selectAuthorSubmit').on('click', function(e){
    var author = $('#selectAuthor').val();
    var encodedAuthor = encodeURIComponent(author);
    console.log(author);
    $.get("/main/author?user=" + encodedAuthor, function(result) {
      //console.log(result);
      $('#authorerror').html("");
      $('#authorAnalyticsResult').html(result);
    });
  });

  //Display Overall section
  $('#overallMenu').on('click', function(e){
    $('#overallSection').css("display", "block");
    $('#individualSection').css("display", "none");
    $('#authorSection').css("display", "none");
  });

  //Display Individual section
  $('#individualMenu').on('click', function(e){
    $('#overallSection').css("display", "none");
    $('#individualSection').css("display", "block");
    $('#authorSection').css("display", "none");
  });

  //Display Author section
  $('#authorMenu').on('click', function(e){
    $('#overallSection').css("display", "none");
    $('#individualSection').css("display", "none");
    $('#authorSection').css("display", "block");
  });

  //Display overall bar chart
  $('#slide-item-1').on('click', function(e){
    $('#overallBarChartSec').css("display", "block");
    $('#overallPieChartSec').css("display", "none");
  });

  //Display overall pie chart
  $('#slide-item-2').on('click', function(e){
    $('#overallBarChartSec').css("display", "none");
    $('#overallPieChartSec').css("display", "block");
  });

  //When user logout
  $('#logout').on('click', function(e){
    console.log("Log out");
    $.get("/logout", function(result) {
      $("html").html(result);
    });
  });
});
```

```
function drawOverallBarChart(){
  var jsondata = {};
  //console.log(overallBarChart);
  overallBarChart.forEach(function(field){
    if(!jsondata[field.year]){
      jsondata[field.year] = {};
    }
    //console.log(field.numbOfRev);
    jsondata[field.year][field.usertype] = parseInt(field.numbOfRev);
  })

  var bar = [['Year', 'Administrator', 'Anonymous', 'Bot', 'Regular']];
  for(var row in jsondata){
    var barRow = [row];
    barRow.push(jsondata[row].admin);
    barRow.push(jsondata[row].anon);
    barRow.push(jsondata[row].bot);
    barRow.push(jsondata[row].regular);
    bar.push(barRow);
  }
  var data = google.visualization.arrayToDataTable(bar);
  var options = {
      'title':"Revision number distribution by year and by user type across the whole dataset",
      'width':900,
      'height':500,
      colors: ['#bf5f5f', '#ffafaf', '#ff9b41', '#bb245f']
  };
  var chart = new google.visualization.ColumnChart($("#overallBarChartSec")[0]);
  chart.draw(data, options);
}

function drawOverallPieChart(){
  var usertype = [['user type', 'number of revision']];
  var pieindex;
  var i = 0;
  //console.log(overallPieChart);
  overallPieChart.forEach(function(field){
    //console.log(field.usertype + field.numbOfRev);
    var piesection = [field.usertype, field.numbOfRev];
    usertype.push(piesection);
    if(field.usertype == "admin"){pieindex = i;}
    i++;
  })
  var str = "";
  var tooltip = [];
  //console.log(overallRevAdminType);
  overallRevAdminType.forEach(function(field){
    //console.log(field._id + field.numbOfRev);
    str = str + field._id + " " + field.numbOfRev + "<br>";
  })
  tooltip.push(str);
```

```
  var data = google.visualization.arrayToDataTable(usertype);
  var options = {
     'title':"Revision number distribution by user type across the whole data set",
     'width':900,
     'height':550,
     colors: ['#bf5f5f', '#ee9c9c', '#f2bc8b', '#de7e34'],
     tooltip: { isHtml: true }
  };
  var chart = new google.visualization.PieChart($("#overallPieChartSec")[0]);

  var sliceid = 0;
  function eventHandler(e){
    chart.setSelection([e]);
    try {
      selection = chart.getSelection();
      //console.log(selection);
      sliceid = selection[0].row - pieindex;
      //console.log("sliceid: " + sliceid);
    }
    catch(err) {
      ;
    }
    //$(".google-visualization-tooltip-item-list li:eq(0)").css("font-weight", "bold");
    $(".google-visualization-tooltip-item-list li:eq(1)").html(tooltip[sliceid]).css("font-family", "Arial");
  }
  google.visualization.events.addListener(chart, 'onmouseover', eventHandler);
  var container = document.getElementById('overallPieChartSec');
  google.visualization.events.addListener(chart, 'ready', function () {
    container.style.display = 'none';
  });
  chart.draw(data, options);
}
```