Technical report

COMP5437 Group 9

Introduction

WikiLyse is a web application designed to help users analyse the revisions made to Wikipedia articles. The dataset used for this application contains information describing the articles, the users editing these articles, as well as a log of comments. WikiLyse users can perform calculations on specific articles or usernames as per their specific requirements, such as the article title with the most revisions. This web application also provides visualisations in the form of pie and bar graphs, allowing for further insights into the revision data. The security of this data is ensured by way of authorisation - users are required to register (providing their email address) in order to use the application.

The following report will illustrate how our group implemented these functionalities. Both functional and non-functional requirements will be explained. The system architecture and the interaction patterns between different components will also be described. This report will also detail the measures taken to improve the performance and security of WikiLyse.

Functional Properties

The architecture of WikiLyse is built based on a Model-View-Controller (MVC) Framework, and the source files are structured accordingly. Breaking down the application into these smaller parts has allowed for greater maintainability during production, while also improving clarity and ease of viewing of source code.

The basic principle upon which the analytical functionality of this application is built is:

1. A request is made through the view for a specific set of data (either as specified by user or as a set function of the page)
2. The controller parses this request and feeds the appropriate information into the model
3. The model connects to the database and retrieves the relevant data, passing it back to the controller
4. The controller passes this data back to the view
5. Data is arranged and displayed as per application specifications for the user to view and interact with.

View

Landing Page

The landing page of WikiLyse (**/views/index.ejs**) features an introduction to the web application, as well as links to three functional options (Overall Analysis, Individual Analysis, and Author Analysis) in the navigation bar. Users need to be authorised by either Logging In or Signing up (available in the navigation bar) before they can follow these three links. See the Security section below for details on implementation of the log-in and sign-up functionalities.

The nav-bar sits permanently at the top of the page, when scrolled it shrinks using a simple javascript function which adjusts the size of bar. The items are organised horizontally using flexboxes.

Analytics

Although they are split into 3 separate links in the navigation bar, the analytical functions are all kept within the same page (see Figure 7.1). When a user clicks one of the three options, the 'href' attribute of the link passes an index into the url. This index is passed into the javascript (see Figure 5.3), and therefore only the relevant portions of the '/article' page are rendered.

A combination of client-side and server-side scripting is used to perform the necessary calculations and rendering of the analytics for Wikilyse. The javascript functions responsible for these tasks are contained in the **/public/javascript/** directory.

For those functions requiring no user input (eg. the charts on the Overall Analysis page and the population of the Article title list on the Individual Analysis page), separate 'GET' requests are sent to query the database upon loading of the page. These are routed to different URLs contained within the page, and so they are rendered on the page asynchronously as the requests are completed without having to wait for everything to load and display at once.

For queries where user input is required (eg. specifying which article and year range to view analytics for on the Individual Analysis page), event handlers are implemented to collect these user-determined values once the relevant 'search' buttons are clicked. These values are then passed on to the controller via 'POST' requests. Various measures have been put into place via client-side scripting to ensure that user inputs are valid (eg. notification windows appear if an invalid range of years is entered) and that searches returning no results do not cause unsightly outputs (by listening for the controller results and editing the relevant HTML elements accordingly).

As analytics for individual articles are generated and displayed, the server application itself queries the WikiPedia API to ensure the catalog of revisions is up to date. The article title is sent to a route with the article title as a request parameter. The final fetch is completed using AXIOS - a promise based HTTP request client/module for NodeJS. This was because the package simplifies the process to a set of three simple commands, and using a promise based system allows the process to occur asynchronously: processing up to 500 revisions to the database can take time, and therefore, a promise based system allows the calculation and display of other analytics while the process continues in the background. AXIOS also is compatible with all browsers, since async/await aren't compatible with internet explorer and other older browsers.

Once the database has been updated, the user is prompted to refresh the page to view an updated version of the analytics.

For the Individual Article Analysis page, a GET request is used to obtain all the different article titles as the page loads, storing it as a Datalist attached to a textbox. This is used as a guide for the user as they search for an article to display analytics on. When an article is chosen various requests are made asynchronously to the server application to calculate the return the results of various queries. These results can then be processed further into a easy to read display for the end-user.

The charts are generated using Google Charts, and they utilise the google.visualization.ColumnChart and google.visualization.PieChart functions respectively (see Figures 5.2 and 5.4). The data received from the controller is converted to a DataTable, which the Google Charts service uses to draw these charts.

Controller

Requests from the View are routed (via the files contained in the **/routes/** directory) to the relevant Controllers (found in the **/controllers/** directory). These call static methods that have been defined in the Model (see below for details), and the resulting data is returned by a callback function. This is then passed back to the view for rendering. Figure 3.2 shows an example of the controller calling a static method in the Model to retrieve data from the database.

Model

The Model files are used to interact with the database, and they rely heavily on the Mongoose library for Node.js and MongoDB. The tasks performed within this section include connecting to the database (Figure 4.1), the creation of schema matching the structure of the revision data files (Figure 4.3), as well as a schema for the representation of WikiLyse users for log-in/sign-up purposes (Figure 4.4). Static methods were also constructed in the Model portion of our application to send queries to the database (Figure 4.3). The use of static methods in particular allows for greater versatility in using the controller to dynamically retrieve data for specific queries.

Database

A MongoDB database is used for storing the revision data. We use an NPM Script to execute a bashscript which imports the data using default mongo functions, and then process them into the needed format. For instance, the JSON files contain timestamps as a string - and those need to be converted to ISODates. Furthermore, a usertype attribute needs to be added to each entity in accordance with the type of user who made the revision - bots, admin, or otherwise.

Non-functional Properties

Security

Sign up/Log in

Security for Wikilyse is ensured by requiring users to sign-up (providing their name, email, and a password) and log in to be able to access the analytical functionalities. Since only authorised users can access WikiLyse, using the URL to directly visit the analytics pages is blocked. Therefore before loading up these pages of the application, an express session needs to be created. An express session is an internal identifier between different pages rendering. As can be seen in Figure 3.3, the session of the login/signup user is checked by querying the user database. If the user logs in with a valid combination of username and password, the user attribute of the current session will be added along with the username. Users can then access all the functionalities of WikiLyse for the life cycle of this session.

Performance

AJAX

Asynchronous Javascript and XML (AJAX) is used to improve the application's performance and user experience. This allows for the page elements to load asynchronously and display as soon as the relevant data has been retrieved from the database, rather than blocking the display of the page until after everything has loaded. It also allows the application to react dynamically to the user's requests and only 'refresh' relevant parts of the page as new requests come in.

Data import

To enhance the performance of importing the data from file to database, we created a shell script and saved that into the node models folder. Thus an npm command could be used to import the data. We originally tried to import the data by using a .js script (Figure 5.5) which was a much slower process due to the huge amount of callback functions executed.

The import script went through multiple iterations to ensure the most effective and efficient method which allows access to all needed data. For instance, usertype queries. As a NoSQL database system, MongoDB's ability for joins is rather limited and inefficient, so it became evident we would need to attach the usertype to the original revisions themselves. The %lookup functionality is quite inefficient as well, both as a preprocessing step and a processing step. Therefore, we came collectively to the conclusion to just include the usertype as a attribute of each revision, no subdocument. For further information see Figures 1.3 and 1.4.

Appendix

Codeblock Figures

1. Root Directory

1.1 /App.JS

```
require('./models/db');
const createError = require('http-errors');
const express = require('express');
const session = require('express-session');
const path = require('path');
const cookieParser = require('cookie-parser');
const logger = require('morgan');
const bodyParser = require('body-parser');
const indexRouter = require('./routes/index');
const loginRouter = require('./routes/login');
const signupRouter =require('./routes/signup');
const articleRouter =require('./routes/articleAnalyze');
const app = express();

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');

app.use(cookieParser('user-Privilege'));
app.use(session({
  secret: 'user-Privilege',
  resave: true,
  saveUninitialized: true,
}));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended: false}));
app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({extended: false}));
app.use(cookieParser());
```

```
app.use(express.static(path.join(__dirname, 'public')));

app.use('/', indexRouter);
app.use('/login', loginRouter);
app.use('/signup', signupRouter);
app.use('/article', articleRouter);

// catch 404 and forward to error handler
app.use(function(req, res, next) {
  next(createError(404));
});

// error handler
app.use(function(err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get('env') === 'development' ? err : {};

  // render the error page
  res.status(err.status || 500);
  res.render('error');
});

module.exports = app;
```

1.2 /package.json

```
{
  "name": "web-dev-9",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www",
    "lint": "eslint \"./**/*.js\"",
    "import": "./importbash"
  },
  "dependencies": {
    "axios": "^0.18.0",
    "babel-eslint": "^10.0.1",
    "cookie-parser": "~1.4.3",
```

```
    "debug": "~2.6.9",
    "ejs": "~2.5.7",
    "eslint": "^5.16.0",
    "eslint-config-google": "^0.12.0",
    "express": "~4.16.0",
    "express-session": "^1.16.1",
    "http-errors": "~1.6.2",
    "jquery": "^3.4.0",
    "mongoose": "~5.5.1",
    "morgan": "~1.9.0"
  }
}
```

1.3 /importbash.sh

```
#!/bin/bash
# get all filename in specified path

path=$1
files=$(ls Dataset_25_March_2019/revisions)
echo -e "================\nImporting Revision Database\n================\n"
for filename in $files
do
   mongoimport --db "revision" --collection "revisions" --type json --file
"Dataset_25_March_2019/revisions/"$filename --jsonArray
done
echo -e "================\nFixing Revision Timestamps\n================\n"
mongo revision --eval "db.revisions.find({}).forEach(function(doc)
{doc.timestamp = new Date(doc.timestamp); db.revisions.save(doc)})"
echo -e "================\nAdding UserType Field\n================\n"
mongo revision --eval "db.revisions.updateMany({'anon':{\$exists:false}},
{\$set:{'usertype':'user'}})"
echo "================\nFixing User Types\n================\n"
mongo revision typecorrection.js
```

1.4 /typecorrection.js

```
var file = cat('Dataset_25_March_2019/admin_active.txt');
var users = file.split('\n');
for(var i = 0; i < users.length; i++) {
  db.revisions.updateMany({'user':users[i]}, {$set: {'usertype':'admin'}});
}

file = cat('Dataset_25_March_2019/admin_former.txt');
users = file.split('\n');
for(var i = 0; i < users.length; i++) {
  db.revisions.updateMany({'user':users[i]}, {$set: {'usertype':'admin'}});
}

file = cat('Dataset_25_March_2019/admin_inactive.txt');
users = file.split('\n');
for(var i = 0; i < users.length; i++) {
  db.revisions.updateMany({'user':users[i]}, {$set: {'usertype':'admin'}});
}

file = cat('Dataset_25_March_2019/admin_semi_active.txt');
users = file.split('\n');
for(var i = 0; i < users.length; i++) {
  db.revisions.updateMany({'user':users[i]}, {$set: {'usertype':'admin'}});
}

file = cat('Dataset_25_March_2019/bot.txt');
users = file.split('\n');
for(var i = 0; i < users.length; i++) {
  db.revisions.updateMany({'user':users[i]}, {$set: {'usertype':'bot'}});
}
```

2. Bin Directory

2.1 /bin/www

```
#!/usr/bin/env node

/**
 * Module dependencies.
 */

var app = require('../app');
var debug = require('debug')('web-dev-9:server');
var http = require('http');

/**
 * Get port from environment and store in Express.
 */

var port = normalizePort(process.env.PORT || '3000');
app.set('port', port);

/**
 * Create HTTP server.
 */

var server = http.createServer(app);

/**
 * Listen on provided port, on all network interfaces.
 */

server.listen(port);
server.on('error', onError);
server.on('listening', onListening);

/**
 * Normalize a port into a number, string, or false.
 */
```

```
function normalizePort(val) {
  var port = parseInt(val, 10);

  if (isNaN(port)) {
    // named pipe
    return val;
  }

  if (port >= 0) {
    // port number
    return port;
  }

  return false;
}

/**
 * Event listener for HTTP server "error" event.
 */

function onError(error) {
  if (error.syscall !== 'listen') {
    throw error;
  }

  var bind = typeof port === 'string'
    ? 'Pipe ' + port
    : 'Port ' + port;

  // handle specific listen errors with friendly messages
  switch (error.code) {
    case 'EACCES':
      console.error(bind + ' requires elevated privileges');
      process.exit(1);
      break;
    case 'EADDRINUSE':
      console.error(bind + ' is already in use');
      process.exit(1);
      break;
    default:
      throw error;
  }
```

```
}

/**
 * Event listener for HTTP server "listening" event.
 */

function onListening() {
  var addr = server.address();
  var bind = typeof addr === 'string'
    ? 'pipe ' + addr
    : 'port ' + addr.port;
  debug('Listening on ' + bind);
}
```

3. Controllers Directory

3.1 /controllers/articleAnalyzeHandler.js

```
const revisions = require('../models/revision');
const axios = require('axios');

exports.default = function(req, res, next) {
  if(req.session.user!=null){
    res.render('articleAnalyze.ejs', {flag:1,name:req.session.user.name});
  }
  else{
    res.render('error',{message:"Please login first."});
  }
};

exports.setRevision_High = function(req, res, next) {
  var number = parseInt(req.body.number);
  revisions.getMostRevision(number,function(err,data){
    if(err){
      console.log(err);
    }

    var articlelist=[];
    for(var i=0;i<number;i++){
      articlelist.push(data[i]._id);
    }
    res.send({articlelist: articlelist, length: data.length});
  })
};

exports.setRevision_Low = function(req, res, next) {
  var number = parseInt(req.body.number);
  revisions.getLeastRevision(number,function(err,data){
    var articlelist=[];
    for(var i=0;i<number;i++){
      articlelist.push(data[i]._id);
    }
    res.send({articlelist: articlelist, length: data.length});
```

```
  })
};

exports.gethistory_H = function(req, res, next) {
  revisions.getMostHistory(function(err,data){
    console.log(data);
    var articlelist=[];
    for(var i=0;i<2;i++){
      articlelist.push(data[i]._id);
    }
    res.send({articlelist: articlelist});
  })
};
exports.gethistory_L = function(req, res, next) {
  revisions.getLeastHistory(function(err,data){
    var articlelist=[];
    articlelist.push(data[0]._id);
    res.send({articlelist: articlelist});
  })
};


exports.getUserNum_H = function(req, res, next) {
 revisions.getMostRegister(function(err,data){
    var articlelist=[];
    console.log(data);
    articlelist.push(data[0]._id);
    res.send({articlelist: articlelist});
 })
};

exports.getUserNum_L = function(req, res, next) {
  revisions.getLeastRegister(function(err,data){
     var articlelist=[];
     articlelist.push(data[0]._id);
     res.send({articlelist: articlelist});
  })
 };

 exports.getDistinctTitlesList = function(req, res, next) {
   revisions.getDistinctTitles(function(err,data){
     var articlelist=[];
```

```
    for (i = 0; i < data.length; i ++) {
      articlelist.push([data[i]._id, data[i].numOfEdits]);
    }

    res.send({articlelist: articlelist});
  })
};

exports.titleTotalRevisions = function(req, res, next) {
  var title = req.body.title;
  var from = req.body.from;
  var to = req.body.to;
   revisions.getTotalRevForTitle(title, from, to, function(err,data){
     if(err){
         console.log(err);
       }
       var articlelist=[];

       articlelist.push(data);

      res.send({articlelist: articlelist});
   })
};


exports.articleTopAuthors = function(req,res,next) {
  var title = req.body.title;
  var from = req.body.from;
  var to = req.body.to;
  revisions.articleTopFiveAuthors(title, from, to, function(err,data){
    if(err){
      console.log(err);
    }
    var authorList=[];

    for (i = 0; i < data.length; i++) {
      authorList.push([data[i]._id, data[i].numOfEdits]);
    }
    res.send({authorList: authorList, length: data.length});
  })
};
```

```
exports.update = function(req, res, next) {
  let responseMessage = 0;
  const articleTitle = req.params.title;
  let timestamp = new Date();
  revisions.getMostRecentRevisionDate(articleTitle, function(err, data) {
    if (err) {
      console.log(err);
    }
    timestamp = data[0].timestamp;
  });
  const mostRecentRevisionURL =
`https://en.wikipedia.org/w/api.php?action=query&format=json&prop=revisions
&titles=${articleTitle}&rvprop=timestamp%7Cuser%7Csha1%7Cuserid%7Csize%7Cid
s%7Cparsedcomment&rvlimit=500&rvstart=${timestamp.toISOString()}&rvdir=olde
r`;
  axios.get(mostRecentRevisionURL).then((response) => {
    const json = response.data.query.pages;
    let article = {};
    for (key in json) {
      article = json[key];
    }
    const articleTimestamp = new Date(article.revisions[0].timestamp);
    if (timestamp < articleTimestamp) {
      responseMessage = 1;
      console.log('Updating: ' + articleTitle);
      const revs = article.revisions;
      for (let i = 0; i < revs.length; i++) {
        const currRevision = revs[i];
        let data = {};
        data['revid'] = currRevision.revid;
        data['parent'] = currRevision.parentid;
        data['user'] = currRevision.user;
        data['userid'] = currRevision.userid;
        data['timestamp'] = currRevision.timestamp;
        data['size'] = currRevision.size;
        data['sha1'] = currRevision.sha1;
        data['parsedcomment'] = currRevision.parsedcomment;
        data['title'] = articleTitle;
        data['usertype'] = 'user';
        if (currRevision['minor']) {
          data['minor'] = '';
```

```
        }
        if (currRevision['anon']) {
          data['anon'] = '';
        }
        const newRevision = new revisions(data);
        newRevision.save((err, rev) => {
          if (err) console.log(err.errors.message);
        });
      }
    }
  }).catch((error) => {
    console.log(error);
  }).finally(function() {
    res.send({'update': responseMessage});
  });
};
exports.userPie = function(req,res,next) {
      revisions.userTypePieChart(function(err, data) {

            var userCount = [];
            if(err) {
                  console.log(err)
            }
            userCount.push(data[0].Unregistered);
            userCount.push(data[0].Regular);
            userCount.push(data[0].Admin);
            userCount.push(data[0].Bot);

            res.send({userCount: userCount});

      })
};

exports.individualPie = function(req,res,next) {
        var title = req.body.title;
        var from = req.body.from;
        var to = req.body.to;
        revisions.individualPieChart(title, from, to, function(err,data){
          if(err){
            console.log(err);
          }
          var userCount = [];
```

```
          userCount.push(data[0].Unregistered);
            userCount.push(data[0].Regular);
            userCount.push(data[0].Admin);
            userCount.push(data[0].Bot);
            res.send({userCount: userCount});
        })
      };

exports.userBar = function(req,res,next) {
      revisions.userTypeBarChart(function(err, data) {

            var userCount = [];
            if(err) {
                  console.log(err)
            }

            userCount.push(data[0].Unregistered);
            userCount.push(data[0].Regular);
            userCount.push(data[0].Admin);
            userCount.push(data[0].Bot);

            res.send({userCount: userCount});

      })
};

exports.articleYearBar = function(req,res,next) {
      var title = req.body.title;
      var from = req.body.from;
    var to = req.body.to;
      revisions.articleUserBarChart(title, from, to, function(err,data){
            var userCount = [];
            if(err) {
                  console.log(err)
            }

            userCount.push(data[0].Unregistered);
            userCount.push(data[0].Regular);
            userCount.push(data[0].Admin);
            userCount.push(data[0].Bot);

            res.send({userCount: userCount});
```

```
      })
}

exports.YearlyUserBar = function(req,res,next){
  var user= req.body.author;
  var title = req.body.title;
  var from = req.body.from;
  var to = req.body.to;
  revisions.userArticlesBarChart(user,title,from,to,function(err,data){
     res.send({dataset:data});
  })



}

exports.authorTitleStats = function(req, res, next) {

      var author = req.body.author;

      revisions.authorTitles(author, function(err, data) {
            if(err){
                  console.log(err);
                }
          var authorTitles=[];
                for (i = 0; i < data.length; i++) {
                  authorTitles.push([data[i]._id, data[i].numOfEdits,
data[i].timestamps]);
                }

                res.send({authorTitles: authorTitles, length:
data.length});
      })
}
```

3.2 /controllers/index.js

```
exports.index = function(req, res, next) {
  if(req.session.user!=null){
    res.render('index', {title:
```

```
'WikiLyse',flag:1,name:req.session.user.name});
  }
  res.render('index', {title: 'WikiLyse',flag:0});

};
```

3.3 /controllers/loginController.js

```
const crypto =require('crypto');
const User =require('../models/user');

exports.login = function(req, res, next) {
  const md5 =crypto.createHash('md5');
  const newPas = md5.update(req.body.password).digest('hex');
  const postData ={
    username: req.body.username,
    password: newPas,
  };
  console.log(postData.username);
  User.findOne({username: postData.username}, function(err, data) {
    if (data) {
      const user={
        name: data.firstname
      };
      req.session.user =user;
      res.render('articleAnalyze.ejs',{flag:1,name:req.session.user.name});
    } else {
      res.render('error',{message:"account or password is wrong, Please
login again"});
    }
  });
};
```

3.4 /controllers/signupController.js

```
const User = require('../models/user');
const crypto =require('crypto');
```

```
exports.Signup = function(req, res, next) {
  const md5 =crypto.createHash('md5');
  const newPas = md5.update(req.body.password).digest('hex');
  const postData ={
    username: req.body.email,
    password: newPas,
    firstname: req.body.firstname,
    secondname: req.body.secondname,
  };
  User.findOne({username: postData.username}, function(err, data) {
    if (data) {

      res.render('error',{message:"This email has already been
used.",flag:0});
    } else {
      User.create(postData, function(err, data) {
        if (err) throw err;
        res.render('error',{message:"Sign-up successful.",flag:0});
      });
    }
  });
};
```

4. Models Directory

4.1 /models/db.js

```
const mango = require('mongoose');
// mango.set('useCreateIndex', true);
mango.connect('mongodb://localhost/revision', {useNewUrlParser: true});

const db = mango.connection;
db.on('error', function callback(error) {
  console.log('Connection error'+error);
});

db.once('open', function callback() {
```

```
  console.log('connected');
});
```

4.2 /models/overViewArticle.js

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

const overViewSchema = new mongoose.Schema({
  revisionNum: Number,
  title: String,
  history: Date,
  registeredNum: Number,
});

module.exports = mongoose.model('overView', overViewSchema);
```

4.3 /models/revision.js

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

const revisionSchema = new mongoose.Schema({
  revid: Number,
  parentid: Number,
  minor: Boolean,
  user: String,
  userid: Number,
  timestamp: Date,
  size: Number,
  sha1: String,
  parsedcomment: String,
  title: String,
  usertype: String,
});

// methods go here
revisionSchema.static('getMostRevision',function(number,callback){
  return this.aggregate([{$group:{_id:"$title", numOfEdits:{$sum:1}}},
{$sort:{numOfEdits:-1}}, {$limit:number}],callback);
})
```

```
revisionSchema.static('getLeastRevision',function(number,callback){
  return this.aggregate([{$group:{_id:"$title", numOfEdits:{$sum:1}}},
{$sort:{numOfEdits:1}}, {$limit:number}],callback);
})

revisionSchema.static('getMostHistory',function(callback){
      var mostHistory = this.aggregate([{$project:{_id:"$title",
age:{$subtract:[new Date(), "$timestamp"]}}}, {$sort:{age:-1}},
{$group:{_id:"$_id", age:{$first:"$age"}}}, {$sort:{age:-1}},
{$limit:2}]).allowDiskUse(true).exec(callback);
      return mostHistory;
})

revisionSchema.static('getLeastHistory',function(callback){
      var leastHistory = this.aggregate([{$project:{_id:"$title",
age:{$subtract:[new Date(), "$timestamp"]}}}, {$sort:{age:-1}},
{$group:{_id:"$_id", age:{$first:"$age"}}}, {$sort:{age:1}},
{$limit:2}]).allowDiskUse(true).exec(callback);
      return leastHistory;
})

revisionSchema.static('getMostRegister',function(callback){
  return this.aggregate([{$match:{anon:{$exists:false}}},
{$group:{_id:"$title", numOfEdits:{$sum:1}}}, {$sort:{numOfEdits:-1}},
{$limit:1}],callback)
})
revisionSchema.static('getLeastRegister',function(callback){
  return this.aggregate([{$match:{anon:{$exists:false}}},
{$group:{_id:"$title", numOfEdits:{$sum:1}}}, {$sort:{numOfEdits:1}},
{$limit:1}],callback)
})
revisionSchema.static('getDistinctTitles',function(callback){
  return this.aggregate([{$group:{_id:"$title", numOfEdits:{$sum:1}}},
{$sort:{_id:1}}],callback);
})

revisionSchema.static('getTotalRevForTitle',function(title,from,to,callback
){
      return this.aggregate([{$match:{"title": title}},
                  {$match:{timestamp:{$gte:new Date(from)}}},
                  {$match:{timestamp:{$lte:new Date(to)}}},
```

```
                  {$group:{_id:"$title", numOfEdits:{$sum:1}}},
                  {$sort:{numOfEdits:-1}},
                  {$limit:5}]
                 ,callback)
})

revisionSchema.static('articleTopFiveAuthors',function(title,from,to,callba
ck){
  return this.aggregate([{$match:{"title": title}},
                                    {$match:{timestamp:{$gte:new
Date(from)}}},
                                    {$match:{timestamp:{$lte:new
Date(to)}}},
                                    {$group:{_id:"$user",
numOfEdits:{$sum:1}}},
                                    {$sort:{numOfEdits:-1}},
                                    {$limit:5}]
                                   ,callback)
})

revisionSchema.static('userTypePieChart', function(callback) {

      return this.aggregate([
      {$facet:
            {"Unregistered":[{$match: {anon:{$exists:true}}},
                             {$count: "Unregistered"}],
        "Regular":[ {$match: {anon:{$exists:false}}},
                    {$match: {usertype:"user"}},
                    {$count: "Regular"}
                     ],
        "Admin":[ {$match: {anon:{$exists:false}}},
                  {$match: {usertype:"admin"}},
                  {$count: "Admin"}
                   ],
        "Bot":[ {$match: {anon:{$exists:false}}},
                {$match: {usertype:"bot"}},
                {$count: "Bot"}
                 ]}},
    {$project:
      {"Unregistered":
            {$arrayElemAt:["$Unregistered.Unregistered", 0]},
        "Regular":
```

```
            {$arrayElemAt:["$Regular.Regular", 0]},
          "Admin":
            {$arrayElemAt:["$Admin.Admin", 0]},
          "Bot":
            {$arrayElemAt:["$Bot.Bot", 0]}
}}], callback)

})

revisionSchema.static('individualPieChart', function(title, from, to,
callback) {

      return this.aggregate([
            {$match:{timestamp:{$gte:new Date(from)}}},
          {$match:{timestamp:{$lte:new Date(to)}}},
            {$match:{"title":title}},
            {$facet:
            {"Unregistered":[{$match: {anon:{$exists:true}}},
                                {$count: "Unregistered"}],
          "Regular":[ {$match: {anon:{$exists:false}}},
                      {$match: {usertype:"user"}},
                      {$count: "Regular"}
                       ],
          "Admin":[ {$match: {anon:{$exists:false}}},
                    {$match: {usertype:"admin"}},
                    {$count: "Admin"}
                     ],
          "Bot":[ {$match: {anon:{$exists:false}}},
                  {$match: {usertype:"bot"}},
                  {$count: "Bot"}
                   ]}},
    {$project:
      {"Unregistered":
            {$arrayElemAt:["$Unregistered.Unregistered", 0]},
          "Regular":
            {$arrayElemAt:["$Regular.Regular", 0]},
          "Admin":
            {$arrayElemAt:["$Admin.Admin", 0]},
          "Bot":
            {$arrayElemAt:["$Bot.Bot", 0]}
}}], callback)
```

```
})


revisionSchema.static('userTypeBarChart', function(callback){
      return this.aggregate([

            {$facet:
                  {"Unregistered":[{$match: {anon:{$exists:true}}},
                              {$project:{year:{$year:"$timestamp"}}},
                              {$group:{_id:"$year",
"Unregistered":{$sum:1}}},
                              {$sort:{_id:1}}
                              ],
                   "Regular":[ {$match: {anon:{$exists:false}}},
                                       {$match: {usertype:"user"}},
                              {$project:{year:{$year:"$timestamp"}}},
                              {$group:{_id:"$year", "Regular":{$sum:1}}},
                              {$sort:{_id:1}}
                                ],
             "Admin":[ {$match: {anon:{$exists:false}}},
                            {$match: {usertype:"admin"}},
                            {$project:{year:{$year:"$timestamp"}}},
                            {$group:{_id:"$year", "Admin":{$sum:1}}},
                            {$sort:{_id:1}}
                               ],
             "Bot":[ {$match: {anon:{$exists:false}}},
                           {$match: {usertype:"bot"}},
                           {$project:{year:{$year:"$timestamp"}}},
                           {$group:{_id:"$year", "Bot":{$sum:1}}},
                           {$sort:{_id:1}}
                            ]
                  }}
       ], callback)

})

revisionSchema.static('articleUserBarChart', function(title, from, to,
callback){
      return this.aggregate([
            {$match:{"title":title}},
            {$match:{timestamp:{$gte:new Date(from)}}},
          {$match:{timestamp:{$lte:new Date(to)}}},
```

```
            {$facet:
                  {"Unregistered":[{$match: {anon:{$exists:true}}},
                              {$project:{year:{$year:"$timestamp"}}},
                              {$group:{_id:"$year",
"Unregistered":{$sum:1}}},
                              {$sort:{_id:1}}
                              ],
                   "Regular":[ {$match: {anon:{$exists:false}}},
                                        {$match: {usertype:"user"}},
                               {$project:{year:{$year:"$timestamp"}}},
                               {$group:{_id:"$year", "Regular":{$sum:1}}},
                               {$sort:{_id:1}}
                               ],
                   "Admin":[ {$match: {anon:{$exists:false}}},
                              {$match: {usertype:"admin"}},
                              {$project:{year:{$year:"$timestamp"}}},
                              {$group:{_id:"$year", "Admin":{$sum:1}}},
                              {$sort:{_id:1}}
                            ],
                   "Bot":[ {$match: {anon:{$exists:false}}},
                           {$match: {usertype:"bot"}},
                           {$project:{year:{$year:"$timestamp"}}},
                           {$group:{_id:"$year", "Bot":{$sum:1}}},
                           {$sort:{_id:1}}
                            ]}}
       ], callback)
})

revisionSchema.static('getDistinctUser',function(callback){
      return
this.aggregate([{$match:{anon:{$exists:false}}},{$group:{_id:"$user",
numOfEdits:{$sum:1}}}, {$sort:{_id:1}}],callback);
})

revisionSchema.static('userArticlesBarChart',
function(user,title,from,to,callback){
      return this.aggregate([
            {$match:{'title':title}},
            {$match:{timestamp:{$gte:new Date(from)}}},
           {$match:{timestamp:{$lte:new Date(to)}}},
            {
                  $facet:
```

```
                  { "1":[{$match: {'user':user[0]}},
                        {$project:{year:{$year:"$timestamp"}}},
                        {$group:{_id:"$year", "count":{$sum:1}}},
                        {$sort:{_id:1}}
                        ],
                        "2":[{$match: {'user':user[1]}},
                        {$project:{year:{$year:"$timestamp"}}},
                        {$group:{_id:"$year", "count":{$sum:1}}},
                        {$sort:{_id:1}}
                        ],
                        "3":[{$match: {'user':user[2]}},
                        {$project:{year:{$year:"$timestamp"}}},
                        {$group:{_id:"$year", "count":{$sum:1}}},
                        {$sort:{_id:1}}
                        ],
                        "4":[{$match: {'user':user[3]}},
                        {$project:{year:{$year:"$timestamp"}}},
                        {$group:{_id:"$year", "count":{$sum:1}}},
                        {$sort:{_id:1}}
                        ],
                        "5":[{$match: {'user':user[4]}},
                        {$project:{year:{$year:"$timestamp"}}},
                        {$group:{_id:"$year", "count":{$sum:1}}},
                        {$sort:{_id:1}}
                        ]

                  }
                  }
      ],callback)

})


revisionSchema.static('authorTitles', function(author, callback){
      return this.aggregate([{$match:{"user":author}},
{$group:{_id:"$title", numOfEdits:{$sum:1},
timestamps:{$push:"$timestamp"}}}, {$sort:{numOfEdits:-1}}], callback)
})

revisionSchema.static('getNumberRevisionsForYearAnon', function(year,
callback) {
  return this.aggregate([{$match: {'timestamp': {$gte: new
```

```
ISODate(`${year}-01-01T00:00:00Z`), $lte: new
ISODate(`${year}-12-31T23:59:59Z`)}, 'anon': {$exists: true}}}, {$count}],
callback);
});

revisionSchema.static('getNumberRevisionsForYearUsertype', function(year,
usertype, callback) {
  return this.aggregate([{$match: {'timestamp': {$gte: new
ISODate(`${year}-01-01T00:00:00Z`), $lte: new
ISODate(`${year}-12-31T23:59:59Z`)}, 'usertype': usertype}}, {$count}],
callback);
});

revisionSchema.static('getMostRecentRevisionDate', function(title,
callback) {
  return this.aggregate([{$match: {'title': title}}, {$project:
{'timestamp': 1}}, {$sort: {'timestamp': -1}}, {$limit: 1}], callback);
});

module.exports = mongoose.model('revision', revisionSchema, 'revisions');
```

4.4 /models/user.js

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

const userSchema = new mongoose.Schema({
  firstname: String,
  secondname: String,
  username: String,
  password: String,

});

// methods here

module.exports = mongoose.model('user', userSchema);
```

5. Public Directory

5.1 /public/javascript/articleAnalyze.js

```
var flag =0;
var flag_over =0;
setRevision = function() {
  const number = document.getElementById('articalNum').value;
  if(number>0) {
        showRevision_H(number);
        showRevision_L(number);
  } else {
        window.alert("Please enter a number greater than 0");
  }
};

function showRevision_H(number){
  const xhrH =new XMLHttpRequest();
  xhrH.open('post', 'article/revision/most', true);
  xhrH.setRequestHeader('Content-Type', 'application/json');
  xhrH.responseType ='json';
  var number ={
    'number': number
  }
  xhrH.send(JSON.stringify(number));
  xhrH.onreadystatechange= function() {
    if (xhrH.status==200 && xhrH.readyState===4) {
      const titles = xhrH.response.articlelist;
      const length =xhrH.response.length;
      const listHighest = document.getElementById('revision list highest');
      let html ='';
      for (let i=0; i<length; i++) {
        html = html+'<li>'+titles[i]+'</li>';
      }
      listHighest.innerHTML=html;
    }
  };
}
function showRevision_L(number){
```

```
  const xhrH =new XMLHttpRequest();
  xhrH.open('post', 'article/revision/least', true);
  xhrH.setRequestHeader('Content-Type', 'application/json');
  xhrH.responseType ='json';
  var number ={
    'number': number
  }
  xhrH.send(JSON.stringify(number));
  xhrH.onreadystatechange= function() {
    if (xhrH.status==200 && xhrH.readyState===4) {
      const titles = xhrH.response.articlelist;
      const length =xhrH.response.length;
      const listHighest = document.getElementById('revision list lowest');
      let html ='';
      for (let i=0; i<length; i++) {
        html = html+'<li>'+titles[i]+'</li>';
      }
      listHighest.innerHTML=html;
    }
  };
}


function showHistory_H() {
  const xhr = new XMLHttpRequest();
  xhr.open('get', '/article/history/most', true);
  xhr.responseType ='json';
  xhr.send();
  xhr.onreadystatechange = function() {
    if (xhr.status==200 && xhr.readyState==4) {
      const titles = xhr.response.articlelist;
      const listLongestHist = document.getElementById('longest history');
      let html='';
      for (let i=0; i<2; i++) {
        html = html+'<li>'+titles[i]+'</li>';
      }
      listLongestHist.innerHTML=html;
    }
  };
}
function showHistory_L() {
  const xhr = new XMLHttpRequest();
```

```
  xhr.open('get', '/article/history/least', true);
  xhr.responseType ='json';
  xhr.send();
  xhr.onreadystatechange = function() {
    if (xhr.status==200 && xhr.readyState==4) {
      const titles = xhr.response.articlelist;
      const listShortestHist = document.getElementById('shortest history');
      let html='';
      html = html+'<li>'+titles[0]+'</li>';
      listShortestHist.innerHTML=html;
    }
  };
}

function showUserNumber_H() {
  const xhr = new XMLHttpRequest();
  xhr.open('get', '/article/User/most', true);
  xhr.responseType ='json';
  xhr.send();
  xhr.onreadystatechange = function() {
    if (xhr.status==200 && xhr.readyState==4) {
      const titles = xhr.response.articlelist;
      const listMostRegister = document.getElementById('most
registeredNum');
      let html='';
      html = html+'<li>'+titles[0]+'</li>';
      listMostRegister.innerHTML=html;
    }
  };
}


function showUserNumber_L() {
  const xhr = new XMLHttpRequest();
  xhr.open('get', '/article/User/least', true);
  xhr.responseType ='json';
  xhr.send();
  xhr.onreadystatechange = function() {
    if (xhr.status==200 && xhr.readyState==4) {
      const titles = xhr.response.articlelist;
      const listLeastRegister = document.getElementById('least
registeredNum');
```

```
      let html='';
      html = html+'<li>'+titles[0]+'</li>';
      listLeastRegister.innerHTML=html;
    }
  };
}

var title_for_chart='';
function populateDatalist() {
  const xhr = new XMLHttpRequest();
  xhr.open('get', '/article/title/list', true);
  xhr.responseType='json';
  xhr.send();

  xhr.onreadystatechange = function() {
    if (xhr.status==200 && xhr.readyState==4) {
      const titles = xhr.response.articlelist;
      const dropdown = document.getElementById('searchTitle');
      let html='';

      for (let i = 0; i < titles.length; i++) {
        html = html + '<option value="' + titles[i][0] + '">[<i>' +
titles[i][1] + ' total revisions</i>]</option>';
      }

      dropdown.innerHTML = html;

      document.getElementById('individualSearch').addEventListener('click',
function() {
        // chuck in query here
        var fromYear = document.getElementById('fromYear').value;
          var toYear = document.getElementById('toYear').value;
          document.getElementById('UserBarChart').innerHTML = '';
          if (parseInt(fromYear) >= 1000 && parseInt(fromYear) <= 9999 &&
parseInt(toYear) >= 1000 && parseInt(toYear) <= 9999) {
              if (parseInt(fromYear) <= parseInt(toYear)) {
                    if(document.getElementsByName('findTitle')[0].value !=
'') {
                document.getElementById('individualAlert').innerHTML = '';
                const wikipediaGet = new XMLHttpRequest();
                wikipediaGet.responseType = 'json';
                wikipediaGet.open('get',
```

```
`/article/revision/update/${document.getElementsByName('findTitle')[0].valu
e}`, true);
                wikipediaGet.onreadystatechange = function() {
                  if(this.readyState == 4 && this.status == 200) {
                    if (this.response.update == 1) {
                      const updateAlert =
document.getElementById('individualAlert');
                      updateAlert.innerHTML = '<h3>There were updates made
to our Database. You may need to refresh for them to take effect.</h3>';
                    }
                  }
                }
                console.log('sending wikipedia request');
                wikipediaGet.send();
                          fromYear = fromYear + "-01-01";
                          toYear = toYear + "-12-31";
                        const title =
document.getElementsByName('findTitle')[0].value;
                        title_for_chart =
document.getElementsByName('findTitle')[0].value;
                        const individualTitle =
document.getElementById('individualTitle');
                        totalForTitle(title, fromYear, toYear);
                        let titleHtml=title;
                        individualTitle.innerHTML = titleHtml;
                        articleTopAuthors(title, fromYear, toYear);
                        drawIndividualPie(title, fromYear, toYear);
                    drawArticleYearBar(title, fromYear, toYear);

document.getElementById('indi_graphs_buttons').style.display='';
                    flag=1;
                      }
                } else {
                    window.alert("'From' year should be before 'To' year");
                }
            } else {
                window.alert("Year should be a 4-digit number");
            }

        });

    }
```

```
  };
}


function totalForTitle(title, from, to) {

  const xhr = new XMLHttpRequest();
  xhr.open('post', 'article/title/total', true);

  xhr.setRequestHeader('Content-Type', 'application/json');
    xhr.responseType ='json';

  xhr.send(JSON.stringify({'title':title, 'from':from, 'to':to}));
  xhr.onreadystatechange = function() {

    if (xhr.status==200 && xhr.readyState==4) {

      const totals = xhr.response.articlelist;
      try {
        const totalField = document.getElementById('individualTotal');
          let html='';
          html = html+"Number of revisions = " + totals[0][0].numOfEdits;
          totalField.innerHTML = html;

          console.log(totals[0]);
      } catch(err) {
        document.getElementById('individualTotal').innerHTML = 'No results
found';
        document.getElementById('topAuthors').innerHTML = '';
      }

    }
  };
}
var topFive =[];

function articleTopAuthors(title, from, to) {
  const xhr = new XMLHttpRequest();
  xhr.open('post', 'article/title/authors', true);
  xhr.setRequestHeader('Content-Type', 'application/json');
    xhr.responseType ='json';
```

```
    xhr.send(JSON.stringify({'title':title, 'from':from, 'to':to}));
  xhr.onreadystatechange = function() {
    if(xhr.status==200 && xhr.readyState==4) {
      topFive = [];
      const topAuthors = xhr.response.authorList;
      const length = xhr.response.length;
      const authorField = document.getElementById('topAuthors');
      let html='';
      for(i = 0; i < length; i++){
        topFive.push(topAuthors[i][0]);
        html = html + '<li>' + topAuthors[i][0] + ': ' + topAuthors[i][1] +
' revisions</li>';
      }



      if (length > 0) {
        authorField.innerHTML = 'Top ' + length + ' authors on this article
are: <ol>' + html + '</ol>';
        userList();
      }

    }
  };
}

authorSearch = function() {
        const author = document.getElementById('findAuthor').value;
        document.getElementById('authorTimes').innerHTML = '';
        authorTitles(author);
      };

function authorTitles(author) {
        const xhr = new XMLHttpRequest();
        xhr.open('post', 'article/author/titles', true);
        xhr.setRequestHeader('Content-Type', 'application/json');
          xhr.responseType ='json';
          var author ={
            'author': author
          }
        xhr.send(JSON.stringify(author));
        xhr.onreadystatechange = function() {
```

```
          if(xhr.status==200 && xhr.readyState==4) {
            const articleList = xhr.response.authorTitles;
            const length = xhr.response.length;
            const authorTitleField =
document.getElementById('authorTitles');
            let html='';
            for(i = 0; i < length; i++){
              html = html + '<li id=' + articleList[i][0] + '>' +
articleList[i][0] + ': ' + articleList[i][1] + ' revisions</li>';
              console.log("articleList[i][0] = " + articleList[i][0])
            }
            authorTitleField.innerHTML = 'Articles edited by ' +
author.author + ' are: <ol>' + html + '</ol>';


document.getElementById('displayTimestamps').addEventListener('click',
function() {
              console.log("length = " + length);
              var timestamps = document.getElementById('authorTimes');
              var html = '';

                    for(i=0; i < length; i++) {

                          if (articleList[i][2].length > 1) {
                                html= html + '<li>' +  articleList[i][0] +
'</li><ul>';
                                console.log("articleList[i].length = " +
articleList[i].length);

                                for(j = 0; j < articleList[i][2].length;
j++) {
                                      html = html + '<li>' +
articleList[i][2][j] + '</li>';
                                }

                          }
                          html = html + '</ul>'
                    }
              timestamps.innerHTML = '<ol>' + html + '</ol>';
                  });
          }
       };
```

```
}



function userList(){
    var users = topFive;
    console.log("topFive = " + topFive);
    const checkboxes = document.getElementById('searchUser');
    const button  = document.getElementById('graphButton');
    let html = '';
    for(let i = 0; i<users.length;i++){
      if(i == 0){
        html = '<p> View the number of revisions (by year) from these top
users:</p>';
        let create_button = '<br><input type="button" id="graphbutton" name
= "graphbutton" value= "Show graphs">'
        button.innerHTML = create_button;
      }
      html  = html + '<input type = "checkbox" name= "topfive" value ="'+
users[i]+'">' + users[i]+'<br>'
    }
    checkboxes.innerHTML = html;


document.getElementById("graphbutton").addEventListener('click',function(){
      document.getElementById("indi_graphs").style.display='none';
      document.getElementById('UserBarChart').style.display='';
      var userSearchList=[];
      var index=[];
      var checkboxs = document.getElementsByName('topfive');
      for(var i=0; i< checkboxs.length; i++){
        if(checkboxs[i].checked){
          userSearchList.push(topFive[i]);
          index.push((i+1).toString());
        }
      }
      if(userSearchList.length==0){
        alert("Please select user(s)");
      }
      else{
        var fromYear = document.getElementById('fromYear').value +
"-01-01";
```

```
          var toYear = document.getElementById('toYear').value + "-12-31";
        drawUserBar(topFive,title_for_chart,index,fromYear,toYear);
      }
    })

}
```

5.2 public/javascript/barGraph.js

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




function drawOverallBar(){

      var overallBarOptions = {'title':"Distribution of All Revisions by
User Type and Year",
                  'width':1200,
                'height':600,
                'vAxis':{title:'Number of Revisions'},
                'hAxis':{title:'Year'},
                'chartArea':{left:40}
      };

      const xhr = new XMLHttpRequest();
      xhr.open('get', '/article/overall/bar', true);
      xhr.responseType = 'json';
      xhr.send();
      xhr.onreadystatechange=function(){
            if(xhr.status==200 && xhr.readyState==4) {
                  var barDataArray = [];
                  barDataArray.push(['Year', 'Unregistered', 'Regular',
'Admin', 'Bot']);
                  const userTypes = xhr.response.userCount;
                  var yearRange = [];

                  for(i = 0; i < userTypes[0].length; i++) {
                        yearRange.push(userTypes[0][i]._id);
                  }
                  for(i = 0; i < userTypes[1].length; i++) {
```

```
                        yearRange.push(userTypes[1][i]._id);
                  }
                  for(i = 0; i < userTypes[2].length; i++) {
                        yearRange.push(userTypes[2][i]._id);
                  }
                  for(i = 0; i < userTypes[3].length; i++) {
                        yearRange.push(userTypes[3][i]._id);
                  }

                  for(i = Math.min(...yearRange); i <=
Math.max(...yearRange); i++) {
                        barDataArray.push([i.toString(),0,0,0,0]);
                  }

                  for (i = 0; i < barDataArray.length - 1; i++) {
                        for(j = 0; j < userTypes[0].length; j++) {
                              if (parseInt(barDataArray[i+1][0]) ==
userTypes[0][j]._id) {
                                    barDataArray[i+1][1] =
userTypes[0][j].Unregistered;
                              }
                        }

                        for(j = 0; j < userTypes[1].length; j++) {
                              if (parseInt(barDataArray[i+1][0]) ==
userTypes[1][j]._id) {
                                    barDataArray[i+1][2] =
userTypes[1][j].Regular;
                              }
                        }

                        for(j = 0; j < userTypes[2].length; j++) {
                              if (parseInt(barDataArray[i+1][0]) ==
userTypes[2][j]._id) {
                                    barDataArray[i+1][3] =
userTypes[2][j].Admin;
                              }
                        }

                        for(j = 0; j < userTypes[3].length; j++) {
                              if (parseInt(barDataArray[i+1][0]) ==
userTypes[3][j]._id) {
```

```
                                    barDataArray[i+1][4] =
userTypes[3][j].Bot;
                                }
                            }

                    }
                    var overallBarData =
google.visualization.arrayToDataTable(barDataArray);
                    var chart = new
google.visualization.ColumnChart(document.getElementById('overallBarChart')
);
                    chart.draw(overallBarData, overallBarOptions);
            }
      }
}

function drawArticleYearBar(title, from, to) {
      var overallBarOptions = {'title':"Distribution of Revisions by User
Type and Year",
                    'width':1200,
                'height':600,
                'vAxis':{title:'Number of Revisions'},
                'hAxis':{title:'Year'},
                'chartArea':{left:40}
      };


       const xhr = new XMLHttpRequest();
        xhr.open('post', 'article/individual/bar', true);
        xhr.setRequestHeader('Content-Type', 'application/json');
          xhr.responseType ='json';

          xhr.send(JSON.stringify({'title':title, 'from':from, 'to':to}));
        xhr.onreadystatechange = function() {
          if(xhr.status==200 && xhr.readyState==4) {
            var barDataArray = [];
                    barDataArray.push(['Year', 'Unregistered', 'Regular',
'Admin', 'Bot']);
                    const userTypes = xhr.response.userCount;
                    var yearRange = [];

                    for(i = 0; i < userTypes[0].length; i++) {
```

```
                        yearRange.push(userTypes[0][i]._id);
                  }
                  for(i = 0; i < userTypes[1].length; i++) {
                        yearRange.push(userTypes[1][i]._id);
                  }
                  for(i = 0; i < userTypes[2].length; i++) {
                        yearRange.push(userTypes[2][i]._id);
                  }
                  for(i = 0; i < userTypes[3].length; i++) {
                        yearRange.push(userTypes[3][i]._id);
                  }
                  for(i = Math.min(...yearRange); i <=
Math.max(...yearRange); i++) {
                        barDataArray.push([i.toString(),0,0,0,0]);
                  }
                  for (i = 0; i < barDataArray.length - 1; i++) {
                        for(j = 0; j < userTypes[0].length; j++) {
                              if (parseInt(barDataArray[i+1][0]) ==
userTypes[0][j]._id) {
                                    barDataArray[i+1][1] =
userTypes[0][j].Unregistered;
                              }
                        }

                        for(j = 0; j < userTypes[1].length; j++) {
                              if (parseInt(barDataArray[i+1][0]) ==
userTypes[1][j]._id) {
                                    barDataArray[i+1][2] =
userTypes[1][j].Regular;
                              }
                        }

                        for(j = 0; j < userTypes[2].length; j++) {
                              if (parseInt(barDataArray[i+1][0]) ==
userTypes[2][j]._id) {
                                    barDataArray[i+1][3] =
userTypes[2][j].Admin;
                              }
                        }

                        for(j = 0; j < userTypes[3].length; j++) {
                              if (parseInt(barDataArray[i+1][0]) ==
```

```
userTypes[3][j]._id) {
                                    barDataArray[i+1][4] =
userTypes[3][j].Bot;
                                }
                            }

                    }

                    var overallBarData =
google.visualization.arrayToDataTable(barDataArray);
                    var chart = new
google.visualization.ColumnChart(document.getElementById('individualBarChar
t'));
                    chart.draw(overallBarData, overallBarOptions);

            }
        }
}
function drawUserBar(user,title,index,from,to){
      var UserBarOptions = {'title':"Yearly Distribution of Revisions by
Top Users",
                    'width':1200,
                'height':600,
                'vAxis':{title:'Number of Revisions'},
                'hAxis':{title:'Year'},
                'chartArea':{left:40}
      };

      const xhr = new XMLHttpRequest();
      xhr.open('post', 'article/user/bar', true);
      xhr.setRequestHeader('Content-Type', 'application/json');
        xhr.responseType ='json';

        var author ={
            'author': user,
            'title': title,
            'from':from,
            'to':to
        }
      xhr.send(JSON.stringify(author));
      xhr.onreadystatechange=function(){
            if(xhr.status==200 && xhr.readyState==4) {
```

```
                  const dataset = xhr.response.dataset;
                  var barDataArray = [];
                  barDataArray.push(['Year']);
                  var selected_data =[];
                  for(var i =0;i<index.length;i++){
                        barDataArray[0].push(user[index[i]-1]);
                        selected_data.push(dataset[0][index[i]]);
                  }
                  var yearRange = new Set();
                  for(var i=0;i<selected_data.length;i++){
                        var temp =selected_data[i];
                        for(var j = 0; j<temp.length;j++){
                              yearRange.add(temp[j]._id);
                        }
                  }
                  var yearArray = Array.from(yearRange);
                  var minYear = Math.min(...yearArray);
                  var maxYear = Math.max(...yearArray);

                  for(i = minYear; i <=maxYear; i++) {
                        var temp =[];
                        temp.push(i.toString());
                        for(var j=0;j<index.length;j++){
                              temp.push(0);
                        }
                        barDataArray.push(temp);
                  }


                  for(var i =1; i<barDataArray.length;i++){
                        var temp = barDataArray[i];
                        for(var j=0;j<selected_data.length;j++){
                              var data = selected_data[j];
                              for(var x =0;x<data.length;x++){
                                    if(parseInt(data[x]._id) ==temp[0]){

barDataArray[i][j+1]=data[x].count;
                                    }
                              }
                        }
                  }
                  var BarData =
```

```
google.visualization.arrayToDataTable(barDataArray);
                    var chart = new
google.visualization.ColumnChart(document.getElementById('UserBarChart'));
                    chart.draw(BarData, UserBarOptions);
                }
        }
}
```

5.3 /public/javascript/content.js

```
function display_article(){
    var stateObj = {foo:'bar'};
    history.pushState(stateObj,'page 1','article');

    document.getElementById("default").style.display='none';
    document.getElementById("article analyze").style.display='';
    setoverviewGraph();
    document.getElementById("individual analytics").style.display='none';
    document.getElementById('author analytics').style.display='none';
}
function setoverviewGraph() {
    if(flag_over==0){
        document.getElementById("overgraphs").style.display='none';
    }
    document.getElementById("showPie").addEventListener('click',function(){
        flag_over=1;
        document.getElementById("overgraphs").style.display='';
        document.getElementById("overallBarChart").style.display='none';
        document.getElementById("overallPieChart").style.display='';
    })
    document.getElementById("showBar").addEventListener('click',function(){
        flag_over=1;
        document.getElementById("overgraphs").style.display='';
        document.getElementById("overallBarChart").style.display='';
        document.getElementById("overallPieChart").style.display='none';
    })

}

function display_individual(){
    var stateObj = {foo:'bar'};
```

```
    history.pushState(stateObj,'page 1','article');
    document.getElementById("default").style.display='none';
    document.getElementById("article analyze").style.display='none';
    document.getElementById("individual analytics").style.display='';
    if(flag==0){

document.getElementById('indi_graphs_buttons').style.display='none';
    }
    else{
        document.getElementById('indi_graphs_buttons').style.display='';
    }
    setindividualGraph();
    document.getElementById('author analytics').style.display='none';
}
function setindividualGraph(){
    if(flag==0){
        document.getElementById("indi_graphs").style.display='none';
    }

document.getElementById("showPie_indi").addEventListener('click',function()
{
        document.getElementById('UserBarChart').style.display='none';
        document.getElementById("indi_graphs").style.display='';
        document.getElementById("individualBarChart").style.display='none';
        document.getElementById("individualPieChart").style.display='';
    })

document.getElementById("showBar_indi").addEventListener('click',function()
{
        document.getElementById('UserBarChart').style.display='none';
        document.getElementById("indi_graphs").style.display='';
        document.getElementById("individualBarChart").style.display='';
        document.getElementById("individualPieChart").style.display='none';
    })

}
function display_author() {
    var stateObj = {foo:'bar'};
    history.pushState(stateObj,'page 1','article');
    document.getElementById("default").style.display='none';
    document.getElementById("article analyze").style.display='none';
    document.getElementById("individual analytics").style.display='none';
```

```
    document.getElementById('author analytics').style.display='';
}

function getQueryVariable(variable)
{
       var query = window.location.search.substring(1);
       var vars = query.split("&");
       for (var i=0;i<vars.length;i++) {
               var pair = vars[i].split("=");
               if(pair[0] == variable){return pair[1];}
       }
       return(false);
}

window.onload =function() {
    if(!getQueryVariable("index")){
        document.getElementById("article analyze").style.display='none';
        document.getElementById("individual
analytics").style.display='none';
        document.getElementById('author analytics').style.display='none';
    }
    else{
        document.getElementById("default").style.display='none';
        switch(getQueryVariable("index")){
            case "1":
                display_article();
                break;
            case "2":
                display_individual();
                break;
            case "3":
                display_author();
                break;
        }
    }


}
```

5.4 /public/javascript/pieGraphs.js

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


function drawOverallPie(){


      var overallPieOptions = {'title':"Distribution of Revisions by User
Type",
                  'width':800,
                  'height':450
            };



      const xhr = new XMLHttpRequest();
      xhr.open('get', '/article/overall/pie', true);
      xhr.responseType = 'json';
      xhr.send();

      xhr.onreadystatechange=function(){
            if(xhr.status==200 && xhr.readyState==4) {
                  overallPieData = new google.visualization.DataTable();
                  overallPieData.addColumn('string', 'UserType');
                  overallPieData.addColumn('number', 'NumEdits');

                  const userTypes = xhr.response.userCount;

                  overallPieData.addRow(["Unregistered", userTypes[0]]);
                  overallPieData.addRow(["Regular", userTypes[1]]);
                  overallPieData.addRow(["Admin", userTypes[2]]);
                  overallPieData.addRow(["Bot", userTypes[3]]);
                  var chart = new
google.visualization.PieChart(document.getElementById('overallPieChart'));
                  chart.draw(overallPieData, overallPieOptions);

            }
      }
}
```

```
function drawIndividualPie(title, from, to) {

      var individualPieOptions = {'title':"Distribution of Revisions by
User Type",
                'width':700,
                'height':450
            };


        const xhr = new XMLHttpRequest();
        xhr.open('post', 'article/individual/pie', true);
        xhr.setRequestHeader('Content-Type', 'application/json');
          xhr.responseType ='json';

        xhr.send(JSON.stringify({'title':title, 'from':from, 'to':to}));
        xhr.onreadystatechange = function() {
          if(xhr.status==200 && xhr.readyState==4) {
            individualPieData = new google.visualization.DataTable();
            individualPieData.addColumn('string', 'UserType');
            individualPieData.addColumn('number', 'NumEdits');

                  const userTypes = xhr.response.userCount;

                  individualPieData.addRow(["Unregistered", userTypes[0]]);
                  individualPieData.addRow(["Regular", userTypes[1]]);
                  individualPieData.addRow(["Admin", userTypes[2]]);
                  individualPieData.addRow(["Bot", userTypes[3]]);
                  var chart = new
google.visualization.PieChart(document.getElementById('individualPieChart')
);
                  chart.draw(individualPieData, individualPieOptions);
          }

        };
}
```

5.5 /public/javascript/importdata.js

```
const mango = require('mongoose');
mango.connect('mongodb://localhost/revision', {useNewUrlParser: true});
const revisions = require('../../models/revision');
```

```
const fs = require('fs');
const folder =
'/Users/XUYIFEI/Desktop/Web_Dev_9/Dataset_25_March_2019/revisions';
fs.readdir(folder, (err, files)=>{
    console.log("importing");
    files.forEach((file) =>{
      var filepath = folder+'/'+file;
      fs.readFile(filepath, function(err, data) {
        if (err) {
          return console.error(err);
        }
        let rev = data.toString();
        rev =JSON.parse(rev);
        for(var i=0;i< rev.length;i++){
            var revid = rev[i].revid;
            var parentid =rev[i].parentid;
            var minor =rev[i].minor;
            var user = rev[i].user;
            var userid = rev[i].userid;
            var timestamp =rev[i].timestamp;
            var size = rev[i].size;
            var sha1 = rev[i].sha1;
            var parsedcomment = rev[i].parsedcomment;
            var title = rev[i].title;

            var revision = {
                revid :revid,
                parentid:parentid,
                minor:minor,
                user:user,
                userid:userid,
                timestamp:timestamp,
                size:size,
                sha1:sha1,
                parsedcomment:parsedcomment,
                title:title
              }
              // revisions.create(revision,function(err,data){
              //   console.log("123");

              // })
              revisions.create(revision,function(err,data){
```

```
                if(err){
                  console.log(err);
                }
              })
        }
    })
    console.log(file+" will be imported");
    });
  });
  console.log("Finish");
```

5.6 /public/javascripts/nav-bar.js

```
window.onscroll = function() {
  const logo = document.getElementById('logo');
  if (document.body.scrollTop > 100 ||
     document.documentElement.scrollTop > 100) {
    logo.style.padding = '1vh 1vw 1vh 1vw';
    logo.style.fontSize = '16px';
  } else {
    logo.style.padding = '10vh 2vw 10vh 2vw';
    logo.style.fontSize = '5vh';
  }
};
```

5.7 /public/javascripts/render.js

```
function addCookie(index){
    $.cookie("index",index);
    console.log($.cookie("index"));
}
```

5.8 /public/stylesheets/style.css

```
body {
  font: 18px "Lucida Grande", Helvetica, Arial, sans-serif;
  margin: 0px;
  z-index: 0;
}
```

```css
a {
  color: whitesmoke;
  text-decoration: none;
}

img {
  height: 45vh;
  width: 100%;
  object-fit: contain;
}

#default{
  text-align: center;
}
#default a{
  color:black;
  margin-left: 10px;
}


.nav {
  background-color: rgb(0, 80, 146);
  position: fixed;
  width: 100%;
  top: 0;
  display: flex;
  flex-direction: row;
  justify-content: space-between;
  color: whitesmoke;
  z-index: 1;
}

.nav-item {
  padding-top: 1vh;
  padding-bottom: 1vh;
  padding-right: 1vw;
  padding-left: 1vw;
  transition: 0.5s;
}

.nav-content {
```

```
  font-size: 16px;
  display: flex;
  justify-content: left;
  width: 33%;
  align-items: center;
}

.nav-logo {
  display: flex;
  justify-content: center;
  width: 34%;
}

#logo {
  padding-top: 10vh;
  padding-bottom: 10vh;
  padding-left: 2vw;
  padding-right: 2vw;
  font-size: 5vh;
  font-weight: bold;
  transition: 0.5s;
}

.nav-user {
  font-size: 16px;
  display: flex;
  justify-content: right;
  width: 33%;
  align-items: center;
}

.content-wrapper{
  display: flex;
  justify-content: center;
  margin-top: 30vh;
  margin-bottom: 25vh;
}

.content {
  width: 50%;
}
```

```css
#individualAlert {
  color: red;
}

.landingpageContainer {
  display: flex;
  flex-wrap: wrap;
  align-items: center;
}

.landingpageContent {
  flex: 1 0 40%;
  margin: 5vh;
  font-size: 20px;
}
```

5.9 /public/stylesheets/login.css

```css
h1 {
    font-size: 24px;
}

label {
    font-size: 20px;
}
```

5.10 /public/stylesheets/signup.css

```css
h1 {
    font-size: 24px;
}

label {
    font-size: 20px;
}
```

6. Route Directory

6.1 /routes/articleAnalyze.js

```
const express =require('express');
const controller =require('../controllers/articleAnalyzeHandler');
const bodyParser = require('body-parser');
const urlencodedParser = bodyParser.urlencoded({extended: false});
const router = new express.Router();

router.get('/', controller.default);
router.post('/revision/most', urlencodedParser,
controller.setRevision_High);
router.post('/revision/least', urlencodedParser,
controller.setRevision_Low);
router.get('/history/most', urlencodedParser, controller.gethistory_H);
router.get('/history/least', urlencodedParser, controller.gethistory_L);
router.get('/User/most', urlencodedParser, controller.getUserNum_H);
router.get('/User/least', urlencodedParser, controller.getUserNum_L);
router.get('/title/list', urlencodedParser,
controller.getDistinctTitlesList);
router.post('/title/total', urlencodedParser,
controller.titleTotalRevisions);
router.post('/title/authors', urlencodedParser,
controller.articleTopAuthors);
router.get('/revision/update/:title', urlencodedParser, controller.update);

router.get('/overall/pie', urlencodedParser, controller.userPie);
router.get('/overall/bar', urlencodedParser, controller.userBar);
router.post('/user/bar',urlencodedParser, controller.YearlyUserBar);

router.post('/individual/pie', urlencodedParser, controller.individualPie);
router.post('/individual/bar', urlencodedParser,
controller.articleYearBar);
router.post('/author/titles', urlencodedParser,
controller.authorTitleStats);
//router.post('/author/timestamps', urlencodedParser,
controller.authorTitleTimestamps);
module.exports = router;
```

6.2 /routes/index.js

```
const express = require('express');
const controller = require('../controllers/index.js');
const router = new express.Router();

/* GET home page. */
router.get('/', controller.index);

module.exports = router;
```

6.3 /routes/login.js

```
const express = require('express');
const router = new express.Router();
const controller =require('../controllers/loginController');


/* GET users listing. */
router.get('/', function(req, res, next) {
  if(req.session.user!=null){
    res.render('error', {message:"You have logged
in.",flag:1,name:req.session.user.name});
  }
  return res.render('../views/login.ejs',{flag:0});
});

router.post('/', controller.login);

module.exports = router;
```

6.4 /routes/logout.js

```
const express = require('express');
const controller = require('../controllers/logoutController');
const router = new express.Router();

/* GET home page. */
router.get('/', controller.logout);
```

```
module.exports = router;
```

6.5 /routes/signup.js

```
const express =require('express');
const controller = require('../controllers/signupController');
const router = new express.Router;

router.get('/', function(req, res, next) {
  if(req.session.user!=null){
    res.render('signup.ejs', {flag:1,name:req.session.user.name});
  }
  return res.render('../views/signup.ejs',{flag:0});
});

router.post('/', controller.Signup);

module.exports =router;
```

7. Views Directory

7.1 /views/articleAnalyze.ejs

```
<!DOCTYPE html>
<html>
  <head>
    <title>WikiLyse</title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
    <script src='/javascripts/nav-bar.js'></script>

<script'type="text/javascript"'src="http://libs.baidu.com/jquery/1.9.1/jque
ry.min.js"></script>
    <script type="text/javascript"
src="/javascript/articleAnalyze.js"></script>
    <script type="text/javascript" src="/javascript/content.js"></script>
    <script type="text/javascript"
src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript" src="/javascript/pieGraphs.js"></script>
    <script type="text/javascript" src="/javascript/barGraphs.js"></script>
</head>
  <body>
    <div class='nav'>
            <div class='nav-content'>
                    <% if (flag) { %>
                        <div class='nav-item'><a href="javascript:"
onclick=display_article() >Overall Analytics</a></div>
                        <div class='nav-item'><a href="javascript:"
onclick=display_individual() >Article Analytics</a></div>
                        <div class='nav-item'><a href="javascript:"
onclick=display_author() >Author Analytics</a></div>
                    <% } %>
            </div>
            <div class='nav-logo'>
                <div class='nav-item' id='logo'><a
href='/'>WikiLyse</a></div>
            </div>
            <div class='nav-user'>
                <% if (flag) { %>
```

```
                    <div class='nav-item'>Hello <%=name%></a></div>
                    <div class='nav-item'><a href='/logout'>Log
out</a></div>
                <% } else { %>
                    <div class='nav-item'><a href='/login'>Login</a></div>
                    <div class='nav-item'><a
href='/signup'>Signup</a></div>
                <% } %>
      </div>
        </div>
    <div class='content-wrapper'>
        <div class='content'>
            <div id ='article analyze'>
                <h1>Overall analysis</h1>
                <script>
                    showHistory_H()
                    showRevision_H(2);
                    showRevision_L(2);
                    showHistory_L();
                    showUserNumber_H();
                    showUserNumber_L();
                    populateDatalist();
                    totalForTitle();
                    articleTopAuthors();
                    drawOverallPie();
                    drawOverallBar();
                    authorSearch();
                    authorTitles();
                    authorsTimestamps();
                    drawArticleYearBar();
                </script>
                <label for='number'>Enter a number to view articles with
the highest/lowest number of revisions:</label>
                <input type="text" id="articalNum" name="articalNum"
value="2">
                <input type="button" value="Go" id ="submit"
onclick="setRevision();"/>
                <h3>Articles with the Highest Number of Revisions</h3>
                    <ul id="revision list highest">

                    </ul>
                <h3>Articles with the Lowest Number of Revisions</h3>
```

```
                    <ul id="revision list lowest">

                    </ul>
                <h3>Articles with the Longest History</h3>
                    <ul id="longest history">

                    </ul>
                <h3>Article with the Shortest History</h3>
                    <ul id="shortest history">

                    </ul>
                <h3>Article edited by the Most Registered Users</h3>
                    <ul id="most registeredNum">

                    </ul>
                <h3>Article edited by the Least Registered Users</h3>
                    <ul id="least registeredNum">

                    </ul>
                <h2>Graphs</h2>
                <p>Choose a graph to display</p>
                <input type="button" id="showPie" value="Pie chart">
                <input type="button" id="showBar" value="Bar chart">
                <div id='overgraphs'>
                    <p id="overallPieChart" class="chart"></p>
                    <p id="overallBarChart" class="chart"></p>
                </div>
            </div>

            <div id='individual analytics'>
                 <h1>Individual Article Analysis</h1>
                Select an article:
                <input type="text" name="findTitle" list="searchTitle"/>
                <datalist id="searchTitle"></datalist>
                <br>
                  Select a range of years to see stats for (enter Year as a
4-digit number):<br>
                  From: <input type="number" id="fromYear"
value="2000"><br>
                  To: <input type="number" id="toYear" value="2019"><br>
                <input type="button" value="Search" id="individualSearch">
                <div id="individualAlert"></div>
```

```
                    <h2 id="individualTitle"></h2>
                    <h3 id="individualTotal"></h3>
                    <h3 id="topAuthors"></h3>

                <div id ="indi_graphs_buttons">
                    <h2>Graphs</h2>
                    <p>Choose a graph to show</p>
                    <input type="button" id="showPie_indi" value="Pie
chart">
                    <input type="button" id="showBar_indi" value="Bar
chart">
                </div>
                <div id='indi_graphs'>
                    <p id="individualPieChart" class="chart"></p>
                    <p id="individualBarChart" class="chart"></p>
                </div>

                <div id="searchUser">
                </div>
                <div id='graphButton' >

                </div>
                <p id = UserBarChart></p>

            </div>
            <div id='author analytics'>
                    <h1>Author Analytics</h1>

                    <label for='author'>Enter an Author username:</label>
                <input type="text" id="findAuthor" name="findAuthor">
                <input type="button" value="Seach for Titles" id ="submit"
onclick="authorSearch();"/>



                    <h2 id="authorTitles"></h2>

                    <button type="button" id ="displayTimestamps">Display
Timestamps</button>
                    <h3 id="authorTimes"></h2>
            </div>
```

```
        </div>
    </div>

    <div id= 'default'>
        <h2>Please make a choice</h2>
        <a href="javascript:;" onclick=display_article()><u>Article
analyze</u></a>
        <a href="javascript:;" onclick=display_individual()><u>individual
analyze</u></a>
        <a href="javascript:;" onclick=display_author()><u>Author
analyze</u></a>
    </div>


  </body>
</html>
```

7.2 /views/error.ejs

```
<!DOCTYPE html>
<html>
  <head>
    <title>WikiLyse</title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
    <script src='/javascripts/nav-bar.js'></script>
    <script src='/javascripts/render.js'></script>
  </head>
  <body>

    <div class='nav'>
      <div class='nav-content'>
          <div class='nav-item'><a href="/article?index=1"  >Article
analyze</a></div>
          <div class='nav-item'><a href="/article?index=2"  >individual
analyze</a></div>
          <div class='nav-item'><a href="/article?index=3"  >Author
analyze</a></div>
      </div>
      <div class='nav-logo'>
        <div class='nav-item' id='logo'><a href='/'>WikiLyse</a></div>
```

```
      </div>
      <div class='nav-user'>
            <% if (flag) { %>
                <div class='nav-item'>Hello <%=name%></a></div>
                <div class='nav-item'><a href='/logout'>Log out</a></div>
            <% } else { %>
                <div class='nav-item'><a href='/login'>Login</a></div>
                <div class='nav-item'><a href='/signup'>Signup</a></div>
            <% } %>
      </div>
    </div>
    <div class='content-wrapper'>
      <div class='content'>
        <%=message%>
    </div>

  </body>
</html>
```

7.3 /views/index.ejs

```
<!DOCTYPE html>
<html>
  <head>
    <title>WikiLyse</title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
    <script src='/javascripts/nav-bar.js'></script>
    <script src='/javascripts/render.js'></script>
    <script src='/javascripts/logout.js'></script>
  </head>
  <body>
    <div class='nav'>
      <div class='nav-content'>
          <% if (flag) { %>
          <div class='nav-item'><a href="/article?index=1"  >Overall
Analytics</a></div>
          <div class='nav-item'><a href="/article?index=2"  >Article
Analytics</a></div>
          <div class='nav-item'><a href="/article?index=3"  >Author
Analytics</a></div>
          <% } %>
```

```
      </div>
      <div class='nav-logo'>
        <div class='nav-item' id='logo'><a href='/'>WikiLyse</a></div>
      </div>
      <div class='nav-user'>
          <% if (flag) { %>
            <div class='nav-item'>Hello <%=name%></a></div>
            <div class='nav-item'><a href='/logout'>Log out</a></div>
            <% } else { %>
            <div class='nav-item'><a href='/login'>Login</a></div>
            <div class='nav-item'><a href='/signup'>Signup</a></div>
          <% } %>
      </div>
    </div>
    <div class='content-wrapper'>
      <div class='content'>
        <h1>Welcome to <%= title %></h1>
        <br><br><br>
        <div class="landingpageContainer">
          <div class="landingpageContent">
            <p>WikiLyse is a platform for generating data analytics from
select Wikipedia Articles.</p>
          </div>
          <div class="landingpageContent">
            <img src="/images/piegraph.png">
          </div>
          <div class="landingpageContent">
            <img src="/images/articledistribution.png">
          </div>
          <div class="landingpageContent">
            <p>WikiLyse can help you visiualise how articles have changed
over time, how they grow.</p>
          </div>
          <div class="landingpageContent">
            <p>View dynamic statistics, updated in real time, for real
people.</p>
          </div>
          <div class="landingpageContent">
            <img src="/images/usertypeyear.png" >
          </div>
        </div>
        </div>
```

```
    </div>

  </body>
</html>
```

7.4 /views/login.ejs

```
<!DOCTYPE html>
<html>
  <head>
    <title>Login</title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
    <link rel='stylesheet' href='/stylesheets/login.css' />
    <script src='/javascripts/nav-bar.js'></script>
  </head>
  <body>
    <div class='nav'>
      <div class='nav-content'>
          <% if (flag) { %>
            <div class='nav-item'><a href="/article?index=1"  >Overall
Analytics</a></div>
            <div class='nav-item'><a href="/article?index=2"  >Article
Analytics</a></div>
            <div class='nav-item'><a href="/article?index=3"  >Author
Analytics</a></div>
          <% } %>
      </div>
      <div class='nav-logo'>
        <div class='nav-item' id='logo'><a href='/'>WikiLyse</a></div>
      </div>
      <div class='nav-user'>
          <% if (flag) { %>
            <div class='nav-item'>Hello <%=name%></a></div>
            <div class='nav-item'><a href='/logout'>Log out</a></div>
            <% } else { %>
            <div class='nav-item'><a href='/login'>Login</a></div>
            <div class='nav-item'><a href='/signup'>Signup</a></div>
          <% } %>
      </div>
    </div>
    <div class='content-wrapper'>
```

```
      <div class='content'>
        <h1>Login</h1>
        <form action="/login", method="POST">
          <p>
            <label for='email'>Email address</label>
            <input type="text" id ='username' name ="username"
placeholder="Email Address" >
          </p>
          <p>
            <label for='password'>Password</label>
            <input type="password" id ='password' name ="password"
placeholder="Password" >
          </p>
          <p>
            <input type="submit" value="Login">
          </p>
        </form>
      </div>
    </div>
  </body>
</html>
```

7.5 /views/signup.ejs

```
<!DOCTYPE html>
<html>
  <head>
    <title>Sign up</title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
    <link rel='stylesheet' href='/stylesheets/singup.css' />
    <script src='/javascripts/nav-bar.js'></script>
  </head>
  <body>
    <div class='nav'>
      <div class='nav-content'>
          <% if (flag) { %>
            <div class='nav-item'><a href="/article?index=1"  >Overall
Analytics</a></div>
            <div class='nav-item'><a href="/article?index=2"  >Article
Analytics</a></div>
            <div class='nav-item'><a href="/article?index=3"  >Author
```

```
Analytics</a></div>
          <% } %>
      </div>
      <div class='nav-logo'>
        <div class='nav-item' id='logo'><a href='/'>WikiLyse</a></div>
      </div>
      <div class='nav-user'>
          <% if (flag) { %>
            <div class='nav-item'>Hello <%=name%></a></div>
            <div class='nav-item'><a href='/logout'>Log out</a></div>
            <% } else { %>
            <div class='nav-item'><a href='/login'>Login</a></div>
            <div class='nav-item'><a href='/signup'>Signup</a></div>
          <% } %>
      </div>
    </div>
    <div class='content-wrapper'>
      <div class='content'>
        <h1>Sign up</h1>
        <form action="/signup", method="POST">
          <p>
              <label for='First name'>First name</label>
              <input type="text" id ='firstname' name ="firstname"
placeholder="First name" >
          </p>
          <p>
              <label for='Second name'>Second name</label>
              <input type="text" id ='secondname' name ="secondname"
placeholder="Second name" >
          </p>
          <p>
              <label for='email'>Email address</label>
              <input type="email" id ='email' name ="email"
placeholder="Email Address" >
          </p>
          <p>
              <label for='password'>Password</label>
              <input type="password" id ='password' name ="password"
placeholder="Password" >
          </p>
          <p>
              <input type="submit" value="Sign up">
```

```
          </p>
        </form>
      </div>
    </div>
  </body>
</html>
```