Assignment 2 Technical Report

(mmun6233, nali3710 and tcha9078)
Michael Muni (450441905), Nafis Ali (480211468) and Tushar Chawla (480193153)

1. Introduction

Article Insight is a web application providing various analytics about Wikipedia articles and their associated revisions. These analytics can be divided into three categories: overall (capturing statistics across all the revisions in the database - e.g. which article has the most revisions), article-specific (capturing statistics relevant to a particular article - e.g. for article 'Australia', who made the most revisions in total) and author-specific (capturing statistics relevant to a particular author - e.g. for an author, show me a list of all revisions made by them). All statistics are generated in real time based on the information currently available in the database. In order to access this information, users must either register an account or login via an already registered account.

2. Technical Specifications

The application is developed using the Javascript language. The backend server is developed using the Express web framework built on top of Node JS. The database platform is MongoDB. The frontend is developed using NUXT.js which is a JS-based framework. The backend and frontend are separate applications where the backend can run independently, and the frontend connects to the backend for its functionalities using Axios, a HTTP client.

3. Application Structure

The application is designed based on the MVC (Model-View-Controller) structure, with the models and controllers residing in the server-side of the application and the view consisting of the entire client-side. The model defines how the data is structured in the database, the controllers are the functionalities of the application which can be accessed via HTTP requests, and the views are the various Vue pages and components which the client interacts with in the browser.

A) Server

The server.js file is the main file that initialized the application, with the config.js file configuring the application startup.

Models

Our application has two models: User and Revision.

i) User

The user model represents the users of the website. All user information is stored in the database based on this model (refer to Fig. 1 in Appendix 1 for full screenshot)

ii) Revision

The revision model represents the article revisions that are stored in the database. A revision in the collection corresponds to a change or "revision" made to a given Wikipedia article at a particular time by a particular user (refer to Fig. 2 in Appendix 1 for full screenshot)

Controllers

There are two controllers representing the two models, entitled revision.controller and user.controller respectively.

i) user.controller

The user controller provides functionalities for registering a site account and logging into the site via valid, authenticated account. Authentication is managed using jsonwebtoken library that encrypts sensitive data such as passwords. It connects to the database via the User model, and it relays the output of its functionality to the Login and Registration Vue pages (which serve as views).

ii) revision.controller

The revision controller is responsible for fetching all the required analytics to be displayed on the website. It interacts with the Revision model through Mongoose queries which in turns interacts with the underlying MongoDB database, and in turn, the output of these functions is relayed to the Overview, Author and Article Vue pages (which serve as views). Sometimes the output is displayed in a text format (see Fig. 3), and in other times, it forms the basis of charts created via vue-charts (a Vue-based wrapper for charts.js) (see Fig. 4).

Routes

There are two routes used to direct HTTP requests to the appropriate controller method, namely revision.routes and user.routes

i) user.routes

This invokes the appropriate methods in user.controller when the user registers an account or tries to log in.

ii) revision.routes

This invokes the appropriate methods in revision.controller to fetch the analytics when the user is on the Overview, Article or Author page.

B) Client

The client is built using the NUXT.js which is an extension to Vue.js framework, where each page corresponds to a view. Each UI elements is a separate custom component. Different pages uses these components to display information. Vuetify was a framework used to theme and enhance the design of the website, and vue-charts (a Vue-wrapper for charts.js) was used to render the necessary charts. The client is a single page application which means the page is loaded once and later dynamically transform the page contents

Components

- ArticleTile (Used to display minimal text-based information for an article)
- HorizontalCards (A group box to hold multiple ArticleTiles of similar type)
- ArticleContainer (Container to hold detailed information about single Article)
- BarChart (used to construct the bar chart for Overview and Article pages)
- PieChart (used to construct the pie chart for Overview and Article pages)

- ChartContainer (encapsulates BarChart and PieChart together to display on Overview and Article pages)
- DatePicker (defines the date filter seen on the Article page)
- Login (defines the login form on the Login page)
- Navbar (defines the navigation bar seen on the top of all site pages)
- Registration (defines the Registration form on the Register page)

Pages

- Article (where user can search for specific article and see all analytics about that article)
- Author (where user can search for specific user and see all analytics about that user)
- Index (home page)
- Login (where user enters account credentials to gain site access)
- Overview (where user can look at analytics pertaining to entire revision database)
- Register (where user can supply details to make a site-associated account)

Store

The store is used to preserve data states between pages. The application has a user store to persist user authentication data.

Middleware

Middlewares are used to provide additional services between the pages. Here, auth.js and guest.js middleware is used to implement access restriction to pages. It ensures only logged in users can access the features whilst others can only access home page, registration and login.

4. Design Decisions

A) Security

The server application has CORS enabled to only let whitelisted domains connect.
User authentication uses *bcrypt* to encrypt sensitive data (can be seen in the bottom of Fig. 1) and *jsonwebtoken* to manage user's state (see Fig. 5).

B) Performance

The application uses asynchronous functions to ensure that it is not blocking the processing thread (see Fig. 6). Every feature has its own separate controller method which makes sure only the minimum amount of data transfer occurs. Indexes are also used to speed up the processing time for certain complex queries (can be seen in User Model Fig. 1).

C) Maintainability

The project is highly modularized by files on both the client and server sides (see Fig. 7 and 8). This makes it easier to identify what each individual component in the application does, and also aids in maintainability and extensibility.

D) Ease of use

The application is designed to simplify tasks for the user wherever possible. When registering an account, if at any point the user has done something wrong (e.g. leave a field

blank), they are alerted immediately (see Fig. 9). Combo boxes are employed to assist users in searching for articles which exist in the database (see Fig. 10).

5. Instructions

A) Pre-requisites

For the website to work, the following needs to be installed on the machine:

1. NodeJS
2. MongoDB

To import the data into MongoDB, use the command line to navigate into the given dataset folder and enter the following command:

```
@echo off
for %%f in ("*.json") do (
"C:\Program Files\MongoDB\Server\4.0\bin\mongoimport.exe" --jsonArray --db insightdb --collection revisions --file "%%~nf.json"
)
```

B) Step-by-step instructions

1. After downloading the master-folder from the GitHub repository, extract it to a directory of your choice.
2. Open up Terminal (if you're using Mac/Linux) or Command Prompt (if you're using Windows)
3. Navigate to the server folder (by typing cd filepathtoDIR/Web_dev_4-master/server where DIR is the directory you extracted to).

   Type the following commands:

   a) npm install
   b) npm install -g nodemon
   c) npm run dev.

4. Open another terminal window and navigate to the client folder (/client instead of/ server in the cd command) and type the same 3 commands as for the server.
5. Type localhost:4200 in the browser to access the site.

Appendix 1: Images

```
const UserSchema = new Schema({
  firstName: {
    type: String,
    required: true,
    trim: true
  },
  lastName: {
    type: String,
    required: true,
    trim: true
  },
  email: {
    type: String,
    unique: true,
    required: true,
    trim: true
  },
  password: {
    type: String,
    required: true,
    trim: true
  }
});

UserSchema.pre("save", function(next) {
  this.password = bcrypt.hashSync(this.password, config.SALT_ROUNDS);
  next();
});
```

Figure 1: the User model

```
const RevisionSchema = new Schema({
  revid: {
    type: Number,
    unique: true,
    required: true
  },
  parentid: {
    type: Number,
    required: true
  },
  minor: {
    type: Boolean
  },
  user: {
    type: String,
    trim: true
  },
  anon: {
    type: Boolean,
    trim: true
  },
  userid: {
    type: Number
  },
  timestamp: {
    type: String,
    trim: true
  },
  size: {
    type: Number,
    trim: true
  },
  sha1: {
    type: String,
    trim: true
  },
  parsedcomment: {
    type: String,
    trim: true
  },
  title: {
    type: String,
    required: true,
    trim: true,
    index: true
  }
});

RevisionSchema.index({ title: 1, timestamp: -1 });
```

Fig 2. the Revision model

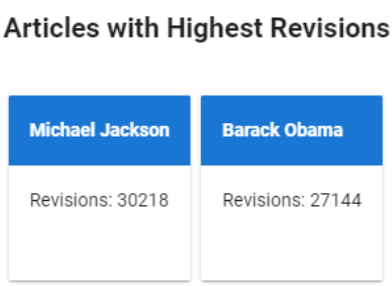

Fig. 3 - example of analytics fetched from revision.controller displayed as text

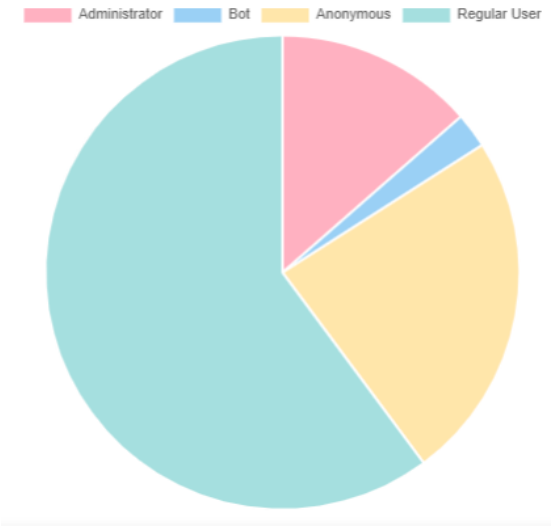

Fig. 4 - example of analytics fetched from revision.controller displayed as a chart

```
const jwt = require("jsonwebtoken");

module.exports = (request, response, next) => {
  const token = request.headers["x-access-token"];

  if (!token) {
    return response.json({ status: "error", message: "No token provided", data: null });
  }

  jwt.verify(token, request.app.get("secretKey"), (error, decoded) => {
    if (error) {
      return response.json({ status: "error", message: "Token authentication failed", data: null });
    }

    request.body.userId = decoded.id;

    next();
  });
};
```

Fig. 5 - using jsonwebtoken to manage user state on server side

```
  countAll: async (request, response, next) => {
    await revisionModel.countDocuments({}, function(err, result) {
      //log error to json response if one occurs
      if (err) {
        response.json({ status: "error", message: "Could not retrieve results", data: null });

        next();
        //log results to json response if successful
      } else {
        response.json({ status: "success", message: "Fetched count", data: result });

        next();
      }
    });
  },
```

Fig. 6 - example of using asynchronous methods to ensure non-blocking

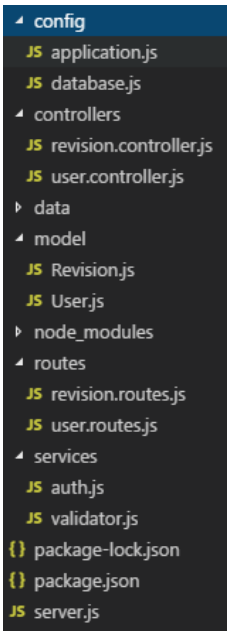

Fig. 7 - the modular file structure of the server-side

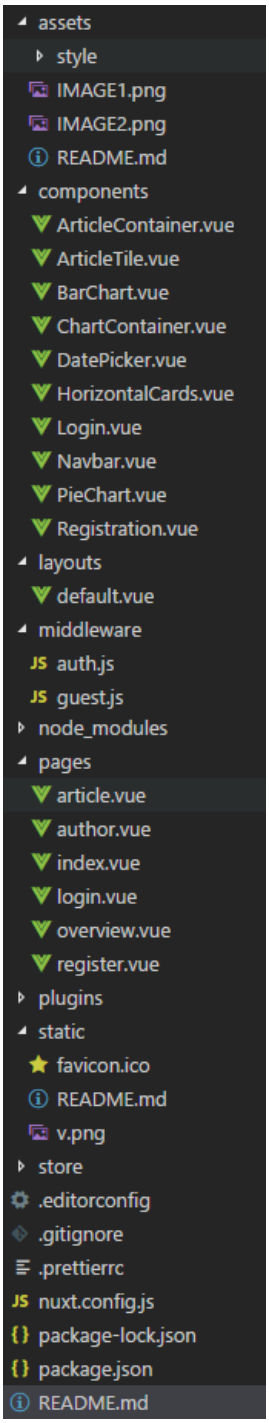

Fig. 8 - the modular file structure on the client side

Fig. 9 - alert which crops up telling user email entered isn't of valid format

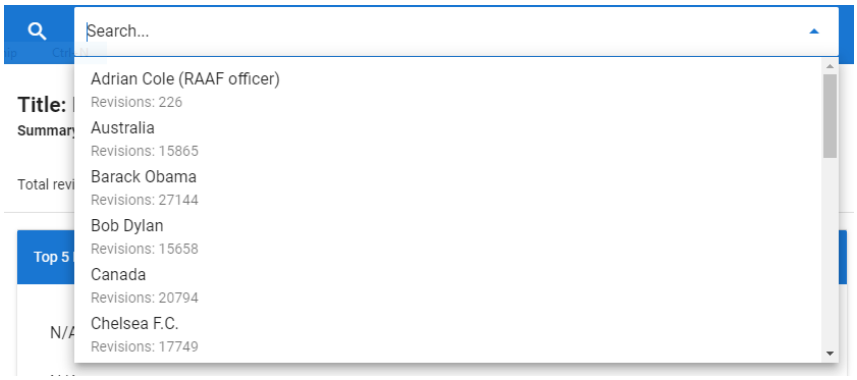

Fig. 10 - combo box to aid user in finding articles

Appendix 2: Source Code

Data Loading

import_data.bat

```
@echo off
for %%f in ("*.json") do (
"C:\Program Files\MongoDB\Server\4.0\bin\mongoimport.exe" --jsonArray --db insightdb --collection revisions --file "%%~nf.json"
)
```

Server

server\model\Revision.js

```
const mongoose = require("mongoose");
const config = require("../config/application");

const Schema = mongoose.Schema;

const RevisionSchema = new Schema({
 revid: {
   type: Number,
   unique: true,
   required: true
 },
```

```
  parentid: {
    type: Number,
    required: true
  },
  minor: {
    type: Boolean
  },
  user: {
    type: String,
    trim: true
  },
  anon: {
    type: Boolean,
    trim: true
  },
  userid: {
    type: Number
  },
  timestamp: {
    type: String,
    trim: true
  },
  size: {
    type: Number,
    trim: true
  },
  sha1: {
    type: String,
    trim: true
  },
  parsedcomment: {
    type: String,
    trim: true
  },
  title: {
    type: String,
    required: true,
    trim: true,
    index: true
  }
});
```

```
RevisionSchema.index({ title: 1, timestamp: -1 });
//RevisionSchema.index({ title: 1, timestamp: 1 });

module.exports = mongoose.model("Revision", RevisionSchema, "revisions");
```

server\model\User.js

```
const mongoose = require("mongoose");
const bcrypt = require("bcrypt");
const config = require("../config/application");

const Schema = mongoose.Schema;

const UserSchema = new Schema({
 firstName: {
   type: String,
   required: true,
   trim: true
 },
 lastName: {
   type: String,
   required: true,
   trim: true
 },
 email: {
   type: String,
   unique: true,
   required: true,
   trim: true
 },
 password: {
   type: String,
   required: true,
   trim: true
 }
});

UserSchema.pre("save", function(next) {
```

```
 this.password = bcrypt.hashSync(this.password, config.SALT_ROUNDS);
 next();
});

module.exports = mongoose.model("User", UserSchema);
```

server\routes\revision.routes.js

```
const revisionController = require("../controllers/revision.controller");
const express = require("express");
const router = express.Router();

//OVERALL ANALYTICS

//fetches total number of revisions in the database
router.get("/", revisionController.countAll);
//fetches the top x articles with the highest number of revisions
router.get("/getHighestRevisionsWithValue",
revisionController.getHighestRevisionsWithValue);
//fetches the top x articles with the fewest number of revisions
router.get("/getLowestRevisionsWithValue",
revisionController.getLowestRevisionsWithValue);
//gets the article in the database with the most users contributing to its
revisions
router.get("/getMostRegUsers", revisionController.getMostRegisteredUsers);
//gets the article in the database with the fewest users contributing to its
revisions
router.get("/getLeastRegUsers", revisionController.getLeastRegisteredUsers);
//gets the x articles (x is a param) whose first revision in the database is
the earliest of any other
router.get("/getOldestArticle", revisionController.getOldestArticle);
//gets the article whose first revision in the database is the latest of any
other
router.get("/getYoungestArticle", revisionController.getYoungestArticle);
//breaks down how many revisions were done by each user type {admin, anon,
bot, regular}
router.get("/getRevisionsByUserType",
revisionController.getRevisionsByUserType);
//breaks down how many revisions were done by each user type for each given
```

```
year
router.get("/getRevisionDistributionByYearUser",
revisionController.getRevisionDistributionByYearUser);

//INDIVIDUAL ANALYTICS

//displays metrics for given article, including total no. of revisions
associated with it,
//and the top 5 users in terms of revision contributions along with how many
revisions they have contributed
router.get("/displaySummaryInfo",
revisionController.displaySummaryInformation);
//breaks down how many revisions were done for specified article by each
user type
router.get("/getArticleRevsByUserType",
revisionController.getArticleRevisionsByUserType);
//breaks down how many revisions were done for specified article by each
user type for each year
router.get("/getArticleRevsByUserTypeAndYear",
revisionController.getArticleRevsByUserTypeAndYear);
//fetches number of revisions for specified article
router.get("/countTitle", revisionController.countTitle);
//fetches all the article titles featured in the revisions database
router.get("/uniqueTitles", revisionController.getUniqueTitles);
//fetches the latest revision for a given article
router.get("/latestRevision", revisionController.getLatestRevision);
//fetches the earliest revision for a given article
router.get("/oldestRevision", revisionController.getOldestRevision);

//AUTHOR ANALYTICS

//fetches all articles which given author has contributed to
router.get("/getArticlesByAuthor", revisionController.getArticlesByAuthor);
//fetches timestamps of revisions made to specified article by specified
author
router.get("/trackArticleRevsByAuthor",
revisionController.trackArticleRevisionsByAuthor);
//get list of authors
router.get("/getAllAuthors", revisionController.getListOfAuthors);

module.exports = router;
```

server\routes\user.routes.js

```
const userController = require("../controllers/user.controller");
const express = require("express");
const router = express.Router();

router.post("/register", userController.create);
router.post("/authenticate", userController.authenticate);

module.exports = router;
```

server\services\auth.js

```
const jwt = require("jsonwebtoken");

module.exports = (request, response, next) => {
 const token = request.headers["x-access-token"];

 if (!token) {
   return response.json({ status: "error", message: "No token provided",
data: null });
 }

 jwt.verify(token, request.app.get("secretKey"), (error, decoded) => {
   if (error) {
     return response.json({ status: "error", message: "Token authentication
failed", data: null });
   }

   request.body.userId = decoded.id;

   next();
 });
};
```

server\services\validator.js

```
module.exports = {
   isEmail: data => {
```

```
    const regex =
/^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-
9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-
Z]{2,}))$/;
    return regex.test(
      String(data)
        .toLowerCase()
        .trim()
    );
  },
  isName: data => {
    const regex = /^[a-zA-Z. ]+$/;
    return regex.test(String(data).trim());
  },
  isPassword: data => {
    return String(data).length >= 8 ? true : false;
  },
  isNumber: data => {
    const regex = /^\d+$/;
    return regex.test(String(data).trim());
  }
};
```

server\config\application.js

```
module.exports = {
 ENV: process.env.NODE_ENV || "development",
 PORT: process.env.PORT || 3000,
 URL: process.env.BASE_URL || "http://localhost:3000",
 MONGODB: process.env.MONGODB_URI || "mongodb://localhost:27017/insightdb",
 SALT_ROUNDS: 10,
 JWT_SECRET: process.env.JWT_SECRET || "articleInsight"
};
```

server\config\database.js

```
const config = require("../config/application");
const mongoose = require("mongoose");

mongoose.connect(config.MONGODB, {
```

```
 useNewUrlParser: true,
 useCreateIndex: true
});

module.exports = mongoose;
```

server\controllers\user.controller.js

```
const userModel = require("../model/User");
const bcrypt = require("bcrypt");
const jwt = require("jsonwebtoken");
const validator = require("../services/validator");

module.exports = {
 // Create a new user at registration
 create: async (request, response, next) => {
   try {
     // Validate registration form data
     if (
       !validator.isName(request.body.firstName) ||
       !validator.isName(request.body.lastName) ||
       !validator.isEmail(request.body.email) ||
       !validator.isPassword(request.body.password)
     ) {
       return response.json({ status: "error", message: "Invalid input
data", data: null });
     }

     const { firstName, lastName, email, password } = request.body;

     const result = await userModel.create({ firstName, lastName, email,
password });

     if (result) {
       response.json({ status: "success", message: "User added
successfully!", data: null });

       next();
     }
   } catch (error) {
```

```
      response.json({ status: "error", message: "Email already in use. Try a
different email.", data: null });

      next();
    }
  },
  // Authenticate user at login
  authenticate: async (request, response, next) => {
    try {
      // Validate login form data and existing user
      const user = validator.isEmail(request.body.email) ? await
userModel.findOne({ email: request.body.email }) : null;

      if (!user) {
        // response.json({ status: "error", message: "User does not exist.
Please register to continue.", data: null });

        // next();
        throw Error("User does not exist. Please register to continue");
      }
      // Perform password decryption and check
      const isMatch = await bcrypt.compare(request.body.password,
user.password);

      if (!isMatch) {
        // response.json({ status: "error", message: "Invalid password.",
data: null });

        // next();
        throw Error("Invalid password");
      }
      // Create token at successful authentication
      const token = jwt.sign({ id: user._id }, request.app.get("secretKey"),
{
        expiresIn: "1h"
      });

      return response.json({
        status: "success",
        data: {
          userName: user.firstName + " " + user.lastName,
```

```
          userEmail: user.email,
          token: token
        }
      });

      //next();
    } catch (error) {
      response.status(500).json({ status: "error", message: error.message });

      next();
    }
  }
};
```

server\controllers\revision.controller.js

```
const revisionModel = require("../model/Revision");
const validator = require("../services/validator");
const fs = require("fs");
const async = require("async");

module.exports = {
 // fetch a count of all records
 countAll: async (request, response, next) => {
   await revisionModel.countDocuments({}, function(err, result) {
     //log error to json response if one occurs
     if (err) {
       response.json({ status: "error", message: "Could not retrieve results",
data: null });

       next();
       //log results to json response if successful
     } else {
       response.json({ status: "success", message: "Fetched count", data:
result });

       next();
     }
   });
 },
```

```
 //fetch a count of records which match the given title
 countTitle: async (request, response, next) => {
   reqTitle = request.query.title;

   await revisionModel.countDocuments({ title: reqTitle }, function(err,
result) {
     //log error to json response if one occurs
     if (err) {
       response.json({ status: "error", message: "Could not retrieve results",
data: null });

       next();
       //log results to json response if successful
     } else {
       response.json({ status: "success", message: "Fetched count", data:
result });

       next();
     }
   });
 },

 //get list of unique titles in the revisions collection
 getUniqueTitles: async (request, response, next) => {
   await revisionModel.aggregate(
     [
       {
         $group: {
           _id: "$title",
           revisions: {
             $sum: 1
           }
         }
       },
       {
         $sort: {
           _id: 1
         }
       }
     ],
```

```
    function(err, result) {
      //log error to json response if one occurs
      if (err) {
        response.json({ status: "error", message: "Could not retrieve titles",
data: null });

        next();
        //log results to json response if successful
      } else {
        response.json({ status: "success", message: "Fetched list of titles",
data: result });

        next();
      }
    }
  );
 },

 //for a given title, find the latest revision done to that article
 getLatestRevision: async (request, response, next) => {
  reqTitle = request.query.title;

  await revisionModel
    .find({ title: reqTitle })
    .sort({ timestamp: -1 })
    .limit(1)
    .exec(function(err, result) {
      if (err) {
        response.json({ status: "error", message: "Could not retrieve
revision", data: null });

        next();
        //log results to json response if successful
      } else {
        response.json({ status: "success", message: "Fetched latest revision",
data: result });

        next();
      }
    });
 },
```

```
 //Get Highest Revisions based on user value
 getHighestRevisionsWithValue: async (request, response, next) => {
   const limit = Number(request.query.limit);

   await revisionModel.aggregate([{ $group: { _id: "$title", count: { $sum: 1 } } }, { $sort: { count: -1 } }, { $limit: limit }], function(err, result) {
     if (err) {
       response.json({ status: "error", message: "Could not retrieve revision", data: null });

       next();
       //log results to json response if successful
     } else {
       response.json({ status: "success", message: "Fetched highest " + limit + " revisions", data: result });

       next();
     }
   });
 },
 // Get Lowest Revisions based on user Value
 getLowestRevisionsWithValue: async (request, response, next) => {
   const limit = Number(request.query.limit);

   await revisionModel.aggregate([{ $group: { _id: "$title", count: { $sum: 1 } } }, { $sort: { count: 1 } }, { $limit: limit }], function(err, result) {
     if (err) {
       response.json({ status: "error", message: "Could not retrieve revision", data: null });

       next();
       //log results to json response if successful
     } else {
       response.json({ status: "success", message: "Fetched lowest " + limit + " revisions", data: result });

       next();
     }
   });
 },
```

```
 //for a given title, find the earliest revision done to that article
 getOldestRevision: async (request, response, next) => {
   reqTitle = request.query.title;

   await revisionModel
     .find({ title: reqTitle })
     .sort({ timestamp: 1 })
     .limit(1)
     .exec(function(err, result) {
       if (err) {
         response.json({ status: "error", message: "Could not retrieve
revision", data: null });

         next();
         //log results to json response if successful
       } else {
         response.json({ status: "success", message: "Fetched oldest revision",
data: result });

         next();
       }
     });
 },

 getMostRegisteredUsers: async (request, response, next) => {
   var mostRegisteredUsersPipeline = [
     { $group: { _id: { title: "$title", userid: "$userid" }, distinctUsers: {
$sum: 1 } } },
     { $group: { _id: "$_id.title", distinctUsers: { $sum: 1 } } },
     { $sort: { distinctUsers: -1 } },
     { $limit: 1 }
   ];

   await revisionModel.aggregate(mostRegisteredUsersPipeline, function(err,
result) {
     if (err) {
       response.json({ status: "error", message: "Problem with executing
aggregate function", data: null });

       next();
     } else {
```

```
        response.json({ status: "success", message: "Fetched article edited by
most users", data: result });

        next();
      }
    });
  },

  getLeastRegisteredUsers: async (request, response, next) => {
    var leastRegisteredUsersPipeline = [
      { $group: { _id: { title: "$title", userid: "$userid" }, distinctUsers: {
$sum: 1 } } },
      { $group: { _id: "$_id.title", distinctUsers: { $sum: 1 } } },
      { $sort: { distinctUsers: 1 } },
      { $limit: 1 }
    ];

    await revisionModel.aggregate(leastRegisteredUsersPipeline, function(err,
result) {
      if (err) {
        response.json({ status: "error", message: "Problem with executing
aggregate function", data: null });

        next();
      } else {
        response.json({ status: "success", message: "Fetched article edited by
fewest users", data: result });

        next();
      }
    });
  },

  getOldestArticle: async (request, response, next) => {
    const limit = parseInt(request.query.limit);

    var oldestArticlePipeline = [
      { $group: { _id: "$title", oldest: { $min: "$timestamp" } } },
      { $project: { title: 1, olddate: { $dateFromString: { dateString:
"$oldest" } } } },
      { $project: { title: 1, age: { $subtract: [new Date(), "$olddate"] } } },
```

```
    { $sort: { age: -1 } },
    { $limit: limit }
  ];

  await revisionModel.aggregate(oldestArticlePipeline, function(err, result) {
    if (err) {
      response.json({ status: "error", message: "Problem with executing
aggregate function", data: null });

      next();
    } else {
      var data = result;
      // converting milliseconds to days
      //data[0].age = Math.floor(parseInt(data[0].age) / (1000 * 60 * 60 *
24));
      response.json({ status: "success", message: "Fetched oldest article",
data: data });

      next();
    }
  });
 },

 getYoungestArticle: async (request, response, next) => {
  const limit = parseInt(request.query.limit);

  var youngestArticlePipeline = [
    { $group: { _id: "$title", oldest: { $min: "$timestamp" } } },
    { $project: { title: 1, olddate: { $dateFromString: { dateString:
"$oldest" } } } },
    { $project: { title: 1, age: { $subtract: [new Date(), "$olddate"] } } },
    { $sort: { age: 1 } },
    { $limit: limit }
  ];

  await revisionModel.aggregate(youngestArticlePipeline, function(err, result)
{
    if (err) {
      response.json({ status: "error", message: "Problem with executing
aggregate function", data: null });
```

```
        next();
      } else {
        var data = result;
        // converting milliseconds to days
        //data[0].age = Math.floor(parseInt(data[0].age) / (1000 * 60 * 60 *
24));
        response.json({ status: "success", message: "Fetched youngest article",
data: data });

        next();
      }
    });
  },

  getRevisionsByUserType: async (request, response, next) => {
    function readAsync(file, callback) {
      fs.readFile(file, "utf8", callback);
    }

    const files = ["data/bot.txt", "data/admin_active.txt",
"data/admin_inactive.txt", "data/admin_semi_active.txt",
"data/admin_former.txt"];

    async.map(files, readAsync, async (error, contents) => {
      var userBots = contents[0].toString().split("\r\n");
      var userAdminActive = contents[1].toString().split("\r\n");
      var userAdminInactive = contents[2].toString().split("\r\n");
      var userAdminSemi = contents[3].toString().split("\r\n");
      var userAdminFormer = contents[4].toString().split("\r\n");
      var allAdmins =
userAdminActive.concat(userAdminInactive.concat(userAdminSemi.concat(userAdminF
ormer)));

      async.parallel(
        {
          bot: function(cb) {
            revisionModel.aggregate(
              [
                {
                  $project: {
                    title: "$title",
```

```
            user: "$user",
            anon: "$anon",
            year: {
              $year: {
                $dateFromString: {
                  dateString: "$timestamp"
                }
              }
            }
          }
        },
        {
          $match: {
            user: { $in: userBots }
          }
        },
        {
          $group: {
            _id: "Bot",
            bot_revisions: {
              $sum: 1
            }
          }
        }
      ],
      function(err, results) {
        if (err) {
          cb(err);
        } else {
          cb(null, results);
        }
      }
    );
  },

  admin: function(cb) {
    revisionModel.aggregate(
      [
        {
          $project: {
            title: "$title",
```

```
            user: "$user",
            anon: "$anon",
            year: {
              $year: {
                $dateFromString: {
                  dateString: "$timestamp"
                }
              }
            }
          }
        },
        {
          $match: {
            user: { $in: allAdmins }
          }
        },
        {
          $group: {
            _id: "Administrator",
            admin_revisions: {
              $sum: 1
            }
          }
        }
      ],
      function(err, results) {
        if (err) {
          cb(err);
        } else {
          cb(null, results);
        }
      }
    );
  },

  anon: function(cb) {
    revisionModel.aggregate(
      [
        {
          $project: {
            title: "$title",
```

```
            user: "$user",
            anon: "$anon",
            year: {
              $year: {
                $dateFromString: {
                  dateString: "$timestamp"
                }
              }
            }
          }
        },
        {
          $match: {
            anon: { $exists: true }
          }
        },
        {
          $group: {
            _id: "Anonymous",
            anon_revisions: {
              $sum: 1
            }
          }
        }
      ],
      function(err, results) {
        if (err) {
          cb(err);
        } else {
          cb(null, results);
        }
      }
    );
  },

  regular: function(cb) {
    revisionModel.aggregate(
      [
        {
          $project: {
            title: "$title",
```

```
                    user: "$user",
                    anon: "$anon",
                    year: { $year: { $dateFromString: { dateString: "$timestamp"
} } }
                  }
                },
                {
                  $match: {
                    $and: [{ user: { $nin: allAdmins } }, { user: { $nin:
userBots } }, { anon: { $exists: false } }]
                  }
                },
                {
                  $group: {
                    _id: "Regular User",
                    reg_revisions: {
                      $sum: 1
                    }
                  }
                }
              ],
              function(err, results) {
                if (err) {
                  cb(err);
                } else {
                  cb(null, results);
                }
              }
            );
          }
        },

        function(err, results) {
          if (err) {
            response.json({ status: "error", message: "didn't get stuff", data:
err });
            next();
          } else {
            json1 = results.admin;
            json2 = results.bot;
            json3 = results.anon;
```

```
            json4 = results.regular;
            // mapped_res = json1.map(x => Object.assign(x, json2.find(y =>
y._id == x._id)));
            // mapped_res = mapped_res.map(x => Object.assign(x, json3.find(y =>
y._id == x._id)));
            // mapped_res = mapped_res.map(x => Object.assign(x, json4.find(y =>
y._id == x._id)));

            data = json1.concat(json2.concat(json3.concat(json4)));

            response.json({ status: "success", message: "got stuff", data: data
});
            next();
          }
        }
      );
    });
  },

  getRevisionDistributionByYearUser: async (request, response, next) => {
    function readAsync(file, callback) {
      fs.readFile(file, "utf8", callback);
    }

    const files = ["data/bot.txt", "data/admin_active.txt",
"data/admin_inactive.txt", "data/admin_semi_active.txt",
"data/admin_former.txt"];

    async.map(files, readAsync, async (error, contents) => {
      var userBots = contents[0].toString().split("\r\n");
      var userAdminActive = contents[1].toString().split("\r\n");
      var userAdminInactive = contents[2].toString().split("\r\n");
      var userAdminSemi = contents[3].toString().split("\r\n");
      var userAdminFormer = contents[4].toString().split("\r\n");
      var allAdmins =
userAdminActive.concat(userAdminInactive.concat(userAdminSemi.concat(userAdminF
ormer)));

      async.parallel(
        {
          bot: function(cb) {
```

```
            revisionModel.aggregate(
              [
                {
                  $project: {
                    title: "$title",
                    user: "$user",
                    anon: "$anon",
                    year: { $year: { $dateFromString: { dateString: "$timestamp"
} } }
                  }
                },
                {
                  $match: {
                    user: { $in: userBots }
                  }
                },
                {
                  $group: {
                    _id: "$year",
                    bot_revisions: { $sum: 1 }
                  }
                },
                {
                  $sort: {
                    _id: 1
                  }
                }
              ],
              function(err, results) {
                if (err) {
                  cb(err);
                } else {
                  cb(null, results);
                }
              }
            );
          },

          admin: function(cb) {
            revisionModel.aggregate(
              [
```

```
            {
              $project: {
                title: "$title",
                user: "$user",
                anon: "$anon",
                year: { $year: { $dateFromString: { dateString: "$timestamp"
} } }
              }
            },
            {
              $match: {
                user: { $in: allAdmins }
              }
            },
            {
              $group: {
                _id: "$year",
                admin_revisions: { $sum: 1 }
              }
            },
            {
              $sort: {
                _id: 1
              }
            }
          ],
          function(err, results) {
            if (err) {
              cb(err);
            } else {
              cb(null, results);
            }
          }
        );
      },

      anon: function(cb) {
        revisionModel.aggregate(
          [
            {
              $project: {
```

```
                  title: "$title",
                  user: "$user",
                  anon: "$anon",
                  year: { $year: { $dateFromString: { dateString: "$timestamp"
} } }
                }
              },
              {
                $match: {
                  anon: { $exists: true }
                }
              },
              {
                $group: {
                  _id: "$year",
                  anon_revisions: { $sum: 1 }
                }
              },
              {
                $sort: {
                  _id: 1
                }
              }
            ],
            function(err, results) {
              if (err) {
                cb(err);
              } else {
                cb(null, results);
              }
            }
          );
        },

        regular: function(cb) {
          revisionModel.aggregate(
            [
              {
                $project: {
                  title: "$title",
                  user: "$user",
```

```
                    anon: "$anon",
                    year: { $year: { $dateFromString: { dateString: "$timestamp"
} } }
                  }
                },
                {
                  $match: {
                    $and: [{ user: { $nin: allAdmins } }, { user: { $nin:
userBots } }, { anon: { $exists: false } }]
                  }
                },
                {
                  $group: {
                    _id: "$year",
                    reg_revisions: { $sum: 1 }
                  }
                },
                {
                  $sort: {
                    _id: 1
                  }
                }
              ],
              function(err, results) {
                if (err) {
                  cb(err);
                } else {
                  cb(null, results);
                }
              }
            );
          }
        },

        function(err, results) {
          if (err) {
            response.json({ status: "error", message: "didn't get stuff", data:
err });
            next();
          } else {
            json1 = results.admin;
```

```
            json2 = results.bot;
            json3 = results.anon;
            json4 = results.regular;

            mapped_res = json1.concat(json2.concat(json3.concat(json4)));
            var result = mapped_res.filter(function(v) {
              return this[v._id] ? !Object.assign(this[v._id], v) : (this[v._id]
= v);
            }, {});

            response.json({ status: "success", message: "got stuff", data:
result });
            next();
          }
        }
      );
    });
  },

  displaySummaryInformation: async (request, response, next) => {
    function readAsync(file, callback) {
      fs.readFile(file, "utf8", callback);
    }

    reqTitle = request.query.title;
    reqFrom = request.query.fromyear != null ? request.query.fromyear : "1970";
    reqTo = request.query.toyear != null ? request.query.toyear : new
Date().getFullYear().toString();

    stringFrom = reqFrom.toString().concat("-01-01T00:00:00Z");
    stringTo = reqTo.toString().concat("-12-31T23:59:59Z");

    const files = ["data/bot.txt", "data/admin_active.txt",
"data/admin_inactive.txt", "data/admin_semi_active.txt",
"data/admin_former.txt"];

    async.map(files, readAsync, async (error, contents) => {
      var userBots = contents[0].toString().split("\r\n");
      var userAdminActive = contents[1].toString().split("\r\n");
      var userAdminInactive = contents[2].toString().split("\r\n");
      var userAdminSemi = contents[3].toString().split("\r\n");
```

```
      var userAdminFormer = contents[4].toString().split("\r\n");
      var allAdmins =
userAdminActive.concat(userAdminInactive.concat(userAdminSemi.concat(userAdminF
ormer)));

      var summaryPipeline = [
        {
          $match: {
            title: reqTitle,
            $expr: {
              $and: [
                {
                  $gte: ["$timestamp", stringFrom]
                },
                {
                  $lte: ["$timestamp", stringTo]
                }
              ]
            }
          }
        },
        {
          $facet: {
            Total: [{ $count: "Total" }],
            TopFive: [
              { $match: { $and: [{ user: { $nin: allAdmins } }, { user: { $nin:
userBots } }, { anon: { $exists: false } }] } },
              { $group: { _id: "$user", usercount: { $sum: 1 } } },
              { $sort: { usercount: -1 } },
              { $limit: 5 }
            ]
          }
        }
      ];

      await revisionModel.aggregate(summaryPipeline, function(err, result) {
        if (err) {
          response.json({ status: "error", message: "Problem with fetching
individual article summary", data: null });

          next();
```

```
      } else {
        response.json({ status: "success", message: "Fetched individual
summary article for " + reqTitle, data: result });

        next();
      }
    });
  });
 },

 getArticleRevisionsByUserType: async (request, response, next) => {
   function readAsync(file, callback) {
     fs.readFile(file, "utf8", callback);
   }

   reqTitle = request.query.title;
   reqFrom = request.query.fromyear != null ? request.query.fromyear : "1970";
   reqTo = request.query.toyear != null ? request.query.toyear : new
Date().getFullYear().toString();

   stringFrom = reqFrom.toString().concat("-01-01T00:00:00Z");
   stringTo = reqTo.toString().concat("-12-31T23:59:59Z");

   const files = ["data/bot.txt", "data/admin_active.txt",
"data/admin_inactive.txt", "data/admin_semi_active.txt",
"data/admin_former.txt"];

   async.map(files, readAsync, async (error, contents) => {
     var userBots = contents[0].toString().split("\r\n");
     var userAdminActive = contents[1].toString().split("\r\n");
     var userAdminInactive = contents[2].toString().split("\r\n");
     var userAdminSemi = contents[3].toString().split("\r\n");
     var userAdminFormer = contents[4].toString().split("\r\n");
     var allAdmins =
userAdminActive.concat(userAdminInactive.concat(userAdminSemi.concat(userAdminF
ormer)));

     async.parallel(
       {
         bot: function(cb) {
           revisionModel.aggregate(
```

```
            [
              {
                $project: {
                  title: "$title",
                  user: "$user",
                  anon: "$anon",
                  timestamp: "$timestamp"
                }
              },
              {
                $match: {
                  $and: [{ user: { $in: userBots } }, { title: reqTitle }, {
timestamp: { $gte: stringFrom, $lte: stringTo } }]
                }
              },
              {
                $group: {
                  _id: "Bot",
                  bot_revisions: {
                    $sum: 1
                  }
                }
              }
            ],
            function(err, results) {
              if (err) {
                cb(err);
              } else {
                cb(null, results);
              }
            }
          );
        },

        admin: function(cb) {
          revisionModel.aggregate(
            [
              {
                $project: {
                  title: "$title",
                  user: "$user",
```

```
                anon: "$anon",
                timestamp: "$timestamp"
              }
            },
            {
              $match: {
                $and: [{ user: { $in: allAdmins } }, { title: reqTitle }, {
timestamp: { $gte: stringFrom, $lte: stringTo } }]
              }
            },
            {
              $group: {
                _id: "Administrator",
                admin_revisions: {
                  $sum: 1
                }
              }
            }
          ],
          function(err, results) {
            if (err) {
              cb(err);
            } else {
              cb(null, results);
            }
          }
        );
      },

      anon: function(cb) {
        revisionModel.aggregate(
          [
            {
              $project: {
                title: "$title",
                user: "$user",
                anon: "$anon",
                timestamp: "$timestamp"
              }
            },
            {
```

```
                $match: {
                  $and: [{ anon: { $exists: true } }, { title: reqTitle }, {
timestamp: { $gte: stringFrom, $lte: stringTo } }]
                }
              },
              {
                $group: {
                  _id: "Anonymous",
                  anon_revisions: {
                    $sum: 1
                  }
                }
              }
            ],
            function(err, results) {
              if (err) {
                cb(err);
              } else {
                cb(null, results);
              }
            }
          );
        },

        regular: function(cb) {
          revisionModel.aggregate(
            [
              {
                $project: {
                  title: "$title",
                  user: "$user",
                  anon: "$anon",
                  timestamp: "$timestamp"
                }
              },
              {
                $match: {
                  $and: [{ user: { $nin: allAdmins } }, { user: { $nin:
userBots } }, { anon: { $exists: false } }, { title: reqTitle }, { timestamp: {
$gte: stringFrom, $lte: stringTo } }]
                }
```

```
            },
            {
              $group: {
                _id: "Regular User",
                regular_revisions: {
                  $sum: 1
                }
              }
            }
          ],
          function(err, results) {
            if (err) {
              cb(err);
            } else {
              cb(null, results);
            }
          }
        );
      }
    },

    function(err, results) {
      if (err) {
        response.json({ status: "error", message: "didnt get article
revisions by user type", data: err });
        next();
      } else {
        json1 = results.admin;
        json2 = results.bot;
        json3 = results.anon;
        json4 = results.regular;
        // mapped_res = json1.map(x => Object.assign(x, json2.find(y =>
y._id == x._id)));
        // mapped_res = mapped_res.map(x => Object.assign(x, json3.find(y =>
y._id == x._id)));
        // mapped_res = mapped_res.map(x => Object.assign(x, json4.find(y =>
y._id == x._id)));

        data = json1.concat(json2.concat(json3.concat(json4)));

        response.json({ status: "success", message: "got article revisions
```

```
by user type", data: data });
            next();
          }
        }
      );
    });
  },

  getArticleRevsByUserTypeAndYear: async (request, response, next) => {
    function readAsync(file, callback) {
      fs.readFile(file, "utf8", callback);
    }

    const files = ["data/bot.txt", "data/admin_active.txt",
"data/admin_inactive.txt", "data/admin_semi_active.txt",
"data/admin_former.txt"];

    reqTitle = request.query.title;
    reqFrom = request.query.fromyear != null ? request.query.fromyear : "1970";
    reqTo = request.query.toyear != null ? request.query.toyear : new
Date().getFullYear().toString();

    stringFrom = reqFrom.toString().concat("-01-01T00:00:00Z");
    stringTo = reqTo.toString().concat("-12-31T23:59:59Z");

    async.map(files, readAsync, async (error, contents) => {
      var userBots = contents[0].toString().split("\r\n");
      var userAdminActive = contents[1].toString().split("\r\n");
      var userAdminInactive = contents[2].toString().split("\r\n");
      var userAdminSemi = contents[3].toString().split("\r\n");
      var userAdminFormer = contents[4].toString().split("\r\n");
      var allAdmins =
userAdminActive.concat(userAdminInactive.concat(userAdminSemi.concat(userAdminF
ormer)));

      async.parallel(
        {
          bot: function(cb) {
            revisionModel.aggregate(
              [
                {
```

```
                $project: {
                  title: "$title",
                  user: "$user",
                  anon: "$anon",
                  timestamp: "$timestamp",
                  year: { $year: { $dateFromString: { dateString: "$timestamp"
} } }
                }
              },
              {
                $match: {
                  $and: [{ user: { $in: userBots } }, { title: reqTitle }, {
timestamp: { $gte: stringFrom, $lte: stringTo } }]
                }
              },
              {
                $group: {
                  _id: "$year",
                  bot_revisions: { $sum: 1 }
                }
              },
              {
                $sort: {
                  _id: 1
                }
              }
            ],
            function(err, results) {
              if (err) {
                cb(err);
              } else {
                cb(null, results);
              }
            }
          );
        },

        admin: function(cb) {
          revisionModel.aggregate(
            [
              {
```

```
                $project: {
                  title: "$title",
                  user: "$user",
                  anon: "$anon",
                  timestamp: "$timestamp",
                  year: { $year: { $dateFromString: { dateString: "$timestamp"
} } }
                }
              },
              {
                $match: {
                  $and: [{ user: { $in: allAdmins } }, { title: reqTitle }, {
timestamp: { $gte: stringFrom, $lte: stringTo } }]
                }
              },
              {
                $group: {
                  _id: "$year",
                  admin_revisions: { $sum: 1 }
                }
              },
              {
                $sort: {
                  _id: 1
                }
              }
            ],
            function(err, results) {
              if (err) {
                cb(err);
              } else {
                cb(null, results);
              }
            }
          );
        },

        anon: function(cb) {
          revisionModel.aggregate(
            [
              {
```

```
                $project: {
                  title: "$title",
                  user: "$user",
                  anon: "$anon",
                  timestamp: "$timestamp",
                  year: { $year: { $dateFromString: { dateString: "$timestamp"
} } }
                }
              },
              {
                $match: {
                  $and: [{ anon: { $exists: true } }, { title: reqTitle }, {
timestamp: { $gte: stringFrom, $lte: stringTo } }]
                }
              },
              {
                $group: {
                  _id: "$year",
                  anon_revisions: { $sum: 1 }
                }
              },
              {
                $sort: {
                  _id: 1
                }
              }
            ],
            function(err, results) {
              if (err) {
                cb(err);
              } else {
                cb(null, results);
              }
            }
          );
        },

        regular: function(cb) {
          revisionModel.aggregate(
            [
              {
```

```
                $project: {
                  title: "$title",
                  user: "$user",
                  anon: "$anon",
                  timestamp: "$timestamp",
                  year: { $year: { $dateFromString: { dateString: "$timestamp"
} } }
                }
              },
              {
                $match: {
                  $and: [{ user: { $nin: allAdmins } }, { user: { $nin:
userBots } }, { anon: { $exists: false } }, { title: reqTitle }, { timestamp: {
$gte: stringFrom, $lte: stringTo } }]
                }
              },
              {
                $group: {
                  _id: "$year",
                  reg_revisions: { $sum: 1 }
                }
              },
              {
                $sort: {
                  _id: 1
                }
              }
            ],
            function(err, results) {
              if (err) {
                cb(err);
              } else {
                cb(null, results);
              }
            }
          );
        }
      },

      function(err, results) {
        if (err) {
```

```
            response.json({ status: "error", message: "failed to get article
revisions by user type and year", data: err });
            next();
          } else {
            json1 = results.admin;
            json2 = results.bot;
            json3 = results.anon;
            json4 = results.regular;

            mapped_res = json1.concat(json2.concat(json3.concat(json4)));
            var result = mapped_res.filter(function(v) {
              return this[v._id] ? !Object.assign(this[v._id], v) : (this[v._id]
= v);
            }, {});

            response.json({ status: "success", message: "got article revisions
by user type and year", data: result });
            next();
          }
        }
      );
    });
  },

  getArticlesByAuthor: async (request, response, next) => {
    reqAuthor = request.query.author;

    var RevisionsByAuthorPipeline = [
      { $match: { user: reqAuthor } },
      { $group: { _id: "$title", no_revisions: { $sum: 1 }, timestamp_list: {
$addToSet: "$timestamp" } } },
      { $sort: { no_revisions: -1 } }
    ];

    await revisionModel.aggregate(RevisionsByAuthorPipeline, function(err,
result) {
      if (err) {
        response.json({ status: "error", message: "Couldn't fetch articles
revised by " + reqAuthor, data: null });

        next();
```

```
    } else {
      response.json({ status: "success", message: "Fetched articles revised by
author " + reqAuthor, data: result });

      next();
    }
  });
},

trackArticleRevisionsByAuthor: async (request, response, next) => {
  reqAuthor = request.query.author;
  reqTitle = request.query.title;

  await revisionModel.find({ title: reqTitle, user: reqAuthor }, { title: 1,
user: 1, timestamp: 1, _id: 0 }).exec(function(err, result) {
    if (err) {
      response.json({ status: "error", message: "Could not retrieve timestamps
for revisions by author " + reqAuthor + " to article " + reqTitle, data: null
});

      next();
      //log results to json response if successful
    } else {
      response.json({ status: "success", message: "Retrieved timestamps for
revisions by author " + reqAuthor + " to article " + reqTitle, data: result });

      next();
    }
  });
},

getListOfAuthors: async (request, response, next) => {
  await revisionModel.aggregate(
    [
      {
        $match: { $and: [{ anon: { $exists: false } }, { user: { $exists: true
} }] }
      },
      {
        $group: {
          _id: "$user",
```

```
            user: {
              $sum: 1
            }
          }
        }
      ],
      function(error, result) {
        if (error) {
          response.json({ status: "error", message: "Didnt get stuff", data:
null });

          next();
        } else {
          response.json({ status: "success", message: "got stuff", data: result
});

          next();
        }
      }
    );
  }
};
```

Client

client\store\user.js

```
export const state = () => ({
 authUser: null,
 loggedIn: false
});

export const mutations = {
 SET_USER: (state, user) => {
   state.authUser = user;
 },
 SET_LOGGEDIN: (state, flag) => {
   state.loggedIn = flag;
 }
};
```

```
export const actions = {
 // nuxtServerInit is called by Nuxt.js before server-rendering every page
 // nuxtServerInit({ commit }, { req }) {
 //   if (req.session && req.session.authUser) {
 //     commit("SET_USER", req.session.authUser);
 //   }
 // },
 async login({ commit }, { email, password }) {
   try {
     const data = await this.$axios.$post("user/authenticate", {
       email,
       password
     });

     if (!data || data.status === "error") {
       throw new Error("Bad credentials");
     }

     commit("SET_USER", data);
     commit("SET_LOGGEDIN", true);
     localStorage.setItem("store", JSON.stringify(this.state));
   } catch (error) {
     if (error.response && error.response.status === 401) {
       throw new Error("Bad credentials");
     }
     throw error;
   }
 },

 async logout({ commit }) {
   commit("SET_USER", null);
   commit("SET_LOGGEDIN", false);
   localStorage.clear();
 },

 initializeStore(state) {
   if (localStorage.getItem("store")) {
     this.replaceState(
       Object.assign(state, JSON.parse(localStorage.getItem("store")))
     );
   } else {
```

```
      this.$router.push("/");
    }
  }
};

export const getters = {
  isAuthenticated: state => {
    return state.loggedIn;
  },
  loggedInUser: state => {
    return state.authUser;
  }
};
```

client\components\ArticleContainer.vue

```
<template>
  <v-card>
    <v-card-title primary-title>
      <v-layout column>
        <h2>Title: {{ articleTitle ? articleTitle : "N/A" }}</h2>
        <h4>Summary Between: {{ yearFrom ? yearFrom : "---" }} to {{ yearTo ? yearTo : "---" }}</h4>
      </v-layout>
    </v-card-title>
    <v-card-text v-if="show">Total revisions: {{ revisionCount }}</v-card-text>
    <v-divider></v-divider>
    <v-card-text>
      <v-divider></v-divider>
      <v-card v-if="show" width="100%">
        <v-card-title class="primary white--text">Top 5 Regular Users</v-card-title>
        <v-card-text>
          <v-list>
            <template v-for="item in regUsers">
              <v-list-tile :key="item._id">
                <v-list-tile-title v-html="item._id"></v-list-tile-title>
                <v-list-tile-sub-title v-html="item.usercount"></v-list-tile-sub-title>
```

```
            </v-list-tile>
          </template>
        </v-list>
      </v-card-text>
    </v-card>
  </v-card-text>
  <!-- <ChartContainer v-if="show" type="individual" :title="articleTitle"
:yearFrom="yearFrom" :yearTo="yearTo"/> -->
 </v-card>
</template>

<script>
import ChartContainer from "~/components/ChartContainer";

export default {
 components: {
   ChartContainer
 },
 data: () => ({
   revisionCount: 0,
   regUsers: [],
   show: false
 }),
 props: ["articleTitle", "yearFrom", "yearTo", "update"],
 methods: {
   async getArticleSummary() {
     if (this.articleTitle !== null && this.update === true) {
       this.show = true;
       const data = await this.$axios.$get("revisions/displaySummaryInfo", {
         headers: {
           "x-access-token": this.$store.state.user.authUser.data.token
         },
         params: {
           title: this.articleTitle,
           fromyear: this.yearFrom ? this.yearFrom : "1970-01-01",
           toyear: this.yearTo
             ? this.yearTo
             : new Date().toISOString().substring(0, 10)
         }
       });
```

```
        this.revisionCount =
          data.data[0].Total.length == 1 ? data.data[0].Total[0].Total : 0;
        this.regUsers = [];

        var topfivelength = data.data[0].TopFive.length;

        for (var i = 0; i < topfivelength; i++) {
          this.regUsers.push(data.data[0].TopFive[i]);
        }

        for (var i = 0; i < 5 - topfivelength; i++) {
          this.regUsers.push({ _id: "N/A", usercount: 0 });
        }

        this.$emit("done", true);
      }
    }
  },
  watch: {
    update: function(value) {
      if (value === true) {
        this.getArticleSummary();
      }
    },
    articleTitle: function(nvalue, ovalue) {
      if (nvalue !== ovalue) {
        this.getArticleSummary();
      }
    },
    yearFrom: function(nvalue, ovalue) {
      if (nvalue !== ovalue) {
        this.getArticleSummary();
      }
    },
    yearTo: function(nvalue, ovalue) {
      if (nvalue !== ovalue) {
        this.getArticleSummary();
      }
    }
  }
};
```

```
</script>

<style>
</style>
```

client\components\ArticleTile.vue

```
<template>
 <v-card min-height="140px" max-width="160px" class="ma-1 elevation-1">
   <v-card-title class="primary white--text font-weight-bold">{{ title
}}</v-card-title>
   <v-card-text>{{ info }}</v-card-text>
 </v-card>
</template>

<script>
export default {
 props: ["title", "info"]
};
</script>

<style>
</style>
```

client\components\BarChart.vue

```
<template>
 <v-card class="elevation-0">
   <v-card-text v-if="type === 'overview'">
     <chartjs-bar :labels="xlabels" :datasets="xdatasets"
:bind="true"></chartjs-bar>
   </v-card-text>
   <v-card-text v-else>
     <chartjs-bar :labels="ylabels" :datasets="ydatasets"
:bind="true"></chartjs-bar>
   </v-card-text>
 </v-card>
</template>

<script>
```

```
export default {
 props: ["type", "title", "yFrom", "yTo", "change"],
 data: () => ({
   xlabels: [],
   xdatasets: [
     {
       label: "Administrator",
       backgroundColor: [],
       borderColor: [],
       borderWidth: 1,
       data: []
     },
     {
       label: "Anonymous",
       backgroundColor: [],
       borderColor: [],
       borderWidth: 1,
       data: []
     },
     {
       label: "Bot",
       backgroundColor: [],
       borderColor: [],
       borderWidth: 1,
       data: []
     },
     {
       label: "Regular User",
       backgroundColor: [],
       borderColor: [],
       borderWidth: 1,
       data: []
     }
   ],
   ylabels: [],
   ydatasets: [
     {
       label: "Administrator",
       backgroundColor: [],
       borderColor: [],
       borderWidth: 1,
```

```
      data: []
    },
    {
      label: "Anonymous",
      backgroundColor: [],
      borderColor: [],
      borderWidth: 1,
      data: []
    },
    {
      label: "Bot",
      backgroundColor: [],
      borderColor: [],
      borderWidth: 1,
      data: []
    },
    {
      label: "Regular User",
      backgroundColor: [],
      borderColor: [],
      borderWidth: 1,
      data: []
    }
  ],
  beginZero: true,
  loaded: false
}),
methods: {
  async getOverallYearRevisionDist() {
    const data = await this.$axios.$get(
      "revisions/getRevisionDistributionByYearUser",
      {
        headers: {
          "x-access-token": this.$store.state.user.authUser.data.token
        }
      }
    );

    for (var i = 0; i < data.data.length; i++) {
      this.xlabels.push(data.data[i]._id);
```

```
      if (data.data[i].admin_revisions) {
        this.xdatasets[0].backgroundColor.push("rgba(255, 99, 132, 0.2)");
        this.xdatasets[0].borderColor.push("rgba(255, 99, 132, 1)");
        this.xdatasets[0].data.push(data.data[i].admin_revisions);
      }
      if (data.data[i].anon_revisions) {
        this.xdatasets[1].backgroundColor.push("rgba(54, 162, 235, 0.2)");
        this.xdatasets[1].borderColor.push("rgba(54, 162, 235, 1)");
        this.xdatasets[1].data.push(data.data[i].anon_revisions);
      }
      if (data.data[i].bot_revisions) {
        this.xdatasets[2].backgroundColor.push("rgba(255, 206, 86, 0.2)");
        this.xdatasets[2].borderColor.push("rgba(255, 206, 86, 1)");
        this.xdatasets[2].data.push(data.data[i].bot_revisions);
      }
      if (data.data[i].reg_revisions) {
        this.xdatasets[3].backgroundColor.push("rgba(75, 192, 192, 0.2)");
        this.xdatasets[3].borderColor.push("rgba(75, 192, 192, 1)");
        this.xdatasets[3].data.push(data.data[i].reg_revisions);
      }
    }

    this.$emit("loaded", true);
  },
  async getIndividualYearRevisionDist() {
      this.ydatasets[0].data = [];
      this.ydatasets[0].backgroundColor = [];
      this.ydatasets[0].borderColor = [];

      this.ydatasets[1].data = [];
      this.ydatasets[1].backgroundColor = [];
      this.ydatasets[1].borderColor = [];

      this.ydatasets[2].data = [];
      this.ydatasets[2].backgroundColor = [];
      this.ydatasets[2].borderColor = [];

      this.ydatasets[3].data = [];
      this.ydatasets[3].backgroundColor = [];
      this.ydatasets[0].borderColor = [];
```

```
    const data = await this.$axios.$get(
      "revisions/getArticleRevsByUserTypeAndYear",
      {
        headers: {
          "x-access-token": this.$store.state.user.authUser.data.token
        },
        params: {
          title: this.title,
          fromyear: this.yFrom ? this.yFrom : "1970-01-01",
          toyear: this.yTo ? this.yTo : new Date().toISOString().substr(0,
10)

        }
      }
    );

    for (var i = 0; i < data.data.length; i++) {
      this.ylabels.push(data.data[i]._id);

      if (data.data[i].admin_revisions) {
        this.ydatasets[0].backgroundColor.push("rgba(255, 99, 132, 0.2)");
        this.ydatasets[0].borderColor.push("rgba(255, 99, 132, 1)");
        this.ydatasets[0].data.push(data.data[i].admin_revisions);
      }
      if (data.data[i].anon_revisions) {
        this.ydatasets[1].backgroundColor.push("rgba(54, 162, 235, 0.2)");
        this.ydatasets[1].borderColor.push("rgba(54, 162, 235, 1)");
        this.ydatasets[1].data.push(data.data[i].anon_revisions);
      }
      if (data.data[i].bot_revisions) {
        this.ydatasets[2].backgroundColor.push("rgba(255, 206, 86, 0.2)");
        this.ydatasets[2].borderColor.push("rgba(255, 206, 86, 1)");
        this.ydatasets[2].data.push(data.data[i].bot_revisions);
      }
      if (data.data[i].reg_revisions) {
        this.ydatasets[3].backgroundColor.push("rgba(75, 192, 192, 0.2)");
        this.ydatasets[3].borderColor.push("rgba(75, 192, 192, 1)");
        this.ydatasets[3].data.push(data.data[i].reg_revisions);
      }
    }

    this.$emit("loaded", true);
```

```
    }
  },
  created() {
    if (this.type === "overview") {
      this.getOverallYearRevisionDist();
    } else {
      this.getIndividualYearRevisionDist();
    }
  },
  watch: {
    title: function(nvalue, ovalue) {
      if (nvalue !== ovalue) {
        this.ylabels = [];
        this.getIndividualYearRevisionDist();
      }
    },
    yFrom: function(nvalue, ovalue) {
      if (nvalue !== ovalue) {
        this.ylabels = [];

        this.getIndividualYearRevisionDist();
      }
    },
    yTo: function(nvalue, ovalue) {
      if (nvalue !== ovalue) {
        this.ylabels = [];

        this.getIndividualYearRevisionDist();
      }
    }
  }
};
</script>

<style>
</style>
```

client\components\ChartContainer.vue

```
<template>
```

```
 <div>
   <v-stepper v-model="e1" v-if="type === 'overview'">
     <v-stepper-header>
       <v-stepper-step step="1">Yearly Revision Distribution by User
Types</v-stepper-step>
       <v-divider></v-divider>
       <v-stepper-step step="2">Revision Number Distribution by User
Type</v-stepper-step>
     </v-stepper-header>
     <v-progress-linear height="2" v-model="value" :active="show"
:indeterminate="query" :query="true"></v-progress-linear>
     <v-stepper-items>
       <v-stepper-content step="1">
         <v-layout fill-height column>
           <BarChart @loaded="progress" :type="type"/>
           <v-btn color="primary" @click="e1 = 2">Change</v-btn>
         </v-layout>
       </v-stepper-content>
       <v-stepper-content step="2">
         <v-layout fill-height column>
           <PieChart @loaded="progress" :type="type"/>
           <v-btn color="primary" @click="e1 = 1">Change</v-btn>
         </v-layout>
       </v-stepper-content>
     </v-stepper-items>
   </v-stepper>
   <v-stepper v-else v-model="e1">
     <v-stepper-header>
       <v-stepper-step step="1">Yearly Revision Distribution by User
Types</v-stepper-step>
       <v-divider></v-divider>
       <v-stepper-step step="2">Revision Number Distribution by User
Type</v-stepper-step>
       <!-- <v-divider></v-divider>
       <v-stepper-step step="3">Revision Number Distribution by User</v-
stepper-step>-->
     </v-stepper-header>
     <v-progress-linear height="2" v-model="value" :active="show"
:indeterminate="query" :query="true"></v-progress-linear>
     <v-stepper-items>
       <v-stepper-content step="1">
```

```
          <v-layout fill-height column>
            <BarChart :title="title" :yFrom="yearFrom" :yTo="yearTo"
@loaded="progress" :change="update"/>
            <v-btn color="primary" @click="e1 = 2">Change</v-btn>
          </v-layout>
        </v-stepper-content>
        <v-stepper-content step="2">
          <v-layout fill-height column>
            <PieChart :title="title" :yFrom="yearFrom" :yTo="yearTo"
@loaded="progress" :change="update"/>
            <v-btn color="primary" @click="e1 = 1">Change</v-btn>
          </v-layout>
        </v-stepper-content>
        <!-- <v-stepper-content step="3">
          <v-layout fill-height column>
            <PieChart :title="title" :yFrom="yearFrom" :yTo="yearTo"
@loaded="progress"/>
            <h1>PLACEHOLDER</h1>
            <v-btn color="primary" @click="e1 = 1">Change</v-btn>
          </v-layout>
        </v-stepper-content>-->
      </v-stepper-items>
    </v-stepper>
  </div>
</template>

<script>
import BarChart from "~/components/BarChart.vue";
import PieChart from "~/components/PieChart.vue";

export default {
  components: {
    BarChart,
    PieChart
  },
  data: () => ({
    e1: 0,
    steps: 2,
    n: 1,
    value: 0,
    query: true,
```

```
    show: true,
    interval: 0
  }),
  props: ["type", "title", "yearFrom", "yearTo", "update"],
  methods: {
    nextStep() {
      if (this.n === this.steps) {
        this.e1 = 1;
      } else {
        this.e1 = this.n + 1;
      }
    },
    progress(status) {
      if (status) {
        this.value += 50;
      }
    }
  },
  watch: {
    steps(val) {
      if (this.e1 > val) {
        this.e1 = val;
      }
    },
    value(val) {
      if (val == 100) {
        this.show = false;
        this.query = false;
      }
    }
  }
};
</script>

<style>
</style>
```

client\components\DatePicker.vue

```
<template>
```

```
 <v-menu ref="menu" v-model="menu" :close-on-content-click="false" :nudge-
right="40" lazy transition="scale-transition" offset-y full-width min-
width="290px">
   <template v-slot:activator="{ on }">
     <v-text-field v-model="date" :label="title" clearable
@click:clear="onClearClicked()" prepend-icon="event" readonly v-on="on"></v-
text-field>
   </template>
   <v-date-picker ref="picker" v-model="date" :max="new
Date().toISOString().substr(0, 10)" min="1950-01-01" @change="save"></v-
date-picker>
 </v-menu>
</template>

<script>
export default {
 data: () => ({
   date: null,
   menu: false
 }),
 props: ["title"],
 watch: {
   menu(val) {
     val && setTimeout(() => (this.$refs.picker.activePicker = "YEAR"));
   }
 },
 methods: {
   save(date) {
     this.$refs.menu.save(date);
     this.$emit("picked", date);
   },
   onClearClicked(date) {
     this.$emit("cleared");
     this.date = null;
     this.$refs.menu.save(date);
   }
 }
};
</script>

<style>
```

```
</style>
```

client\components\HorizontalCards.vue

```
<template>
 <v-card width="100%" class="mb-3">
   <v-card-title class="font-weight-bold">
     <h2>{{ objectHeader }}</h2>
   </v-card-title>
   <v-card-text>
     <v-layout wrap row>
       <div class="loader" v-if="loading">
         <v-progress-circular :width="5" :size="50" color="primary"
indeterminate></v-progress-circular>
       </div>
       <ArticleTile v-for="item in items" :key="item.title" v-
bind:title="item.title" v-bind:info="item.info"/>
       <!-- <v-list row>
         <template v-for="item in items" v-bind:title="item.title">
           <v-list-tile-content :key="item.title">
             <v-list-tile-title v-html="item.title"></v-list-tile-title>
             <v-list-tile-sub-title v-html="item.info"></v-list-tile-sub-
title>
             <v-divider></v-divider>
           </v-list-tile-content>
         </template>
       </v-list>-->
     </v-layout>
   </v-card-text>
 </v-card>
</template>

<script>
import ArticleTile from "~/components/ArticleTile.vue";

export default {
 components: {
   ArticleTile
 },
 props: ["objectType", "objectHeader", "counter"],
```

```
data: () => ({
  //counter: 2,
  loading: false,
  items: []
}),
methods: {
  async getHighestRevisionArticles() {
    //console.log(this.$store.state.user.authUser.data.token);
    this.items = [];
    this.loading = true;
    this.$emit("loading", true);

    const data = await this.$axios.$get(
      "revisions/getHighestRevisionsWithValue",
      {
        headers: {
          "x-access-token": this.$store.state.user.authUser.data.token
        },
        params: { limit: this.counter }
      }
    );

    if (data) {
      this.loading = false;
      for (var i = 0; i < data.data.length; i++) {
        let item = {
          title: data.data[i]._id.trim(),
          info: "Revisions: " + data.data[i].count
        };

        this.items.push(item);
      }
    }
    this.$emit("loading", false);
  },

  async getLowestRevisionArticles() {
    this.items = [];
    this.loading = true;
    this.$emit("loading", true);
```

```
    const data = await this.$axios.$get(
      "revisions/getLowestRevisionsWithValue",
      {
        headers: {
          "x-access-token": this.$store.state.user.authUser.data.token
        },
        params: { limit: this.counter }
      }
    );

    if (data) {
      this.loading = false;
      for (var i = 0; i < data.data.length; i++) {
        let item = {
          title: data.data[i]._id.trim(),
          info: "Revisions: " + data.data[i].count
        };

        this.items.push(item);
      }
    }
    this.$emit("loading", false);
  },

  async getMostRegisteredUserArticle() {
    this.items = [];
    this.loading = true;
    const data = await this.$axios.$get("revisions/getMostRegUsers", {
      headers: {
        "x-access-token": this.$store.state.user.authUser.data.token
      }
    });

    if (data) {
      this.loading = false;
      for (var i = 0; i < data.data.length; i++) {
        let item = {
          title: data.data[i]._id.trim(),
          info: "Registered Users: " + data.data[i].distinctUsers
        };
```

```
        this.items.push(item);
      }
    }
  },
  async getLeastRegisteredUserArticle() {
    this.items = [];
    this.loading = true;
    const data = await this.$axios.$get("revisions/getLeastRegUsers", {
      headers: {
        "x-access-token": this.$store.state.user.authUser.data.token
      }
    });

    if (data) {
      this.loading = false;
      for (var i = 0; i < data.data.length; i++) {
        let item = {
          title: data.data[i]._id.trim(),
          info: "Registered Users: " + data.data[i].distinctUsers
        };

        this.items.push(item);
      }
    }
  },
  async getLongestHistoryArticle() {
    this.items = [];
    this.loading = true;
    const data = await this.$axios.$get("revisions/getOldestArticle", {
      headers: {
        "x-access-token": this.$store.state.user.authUser.data.token
      },
      params: {
        limit: 2
      }
    });

    if (data) {
      this.loading = false;
      for (var i = 0; i < data.data.length; i++) {
        let item = {
```

```
          title: data.data[i]._id.trim(),
          info:
            "Age: " +
            Math.floor(data.data[i].age / (1000 * 60 * 60 * 24)) +
            " days"
        };

        this.items.push(item);
      }
    }
  },
  async getYoungestHistoryArticle() {
    this.items = [];
    this.loading = true;
    const data = await this.$axios.$get("revisions/getYoungestArticle", {
      headers: {
        "x-access-token": this.$store.state.user.authUser.data.token
      },
      params: {
        limit: 1
      }
    });

    if (data) {
      this.loading = false;
      for (var i = 0; i < data.data.length; i++) {
        let item = {
          title: data.data[i]._id.trim(),
          info:
            "Age: " +
            Math.floor(data.data[i].age / (1000 * 60 * 60 * 24)) +
            " days"
        };

        this.items.push(item);
      }
    }
  }
},
created() {
  switch (this.objectType) {
```

```
      case "hirev":
        this.getHighestRevisionArticles();
        break;
      case "lorev":
        this.getLowestRevisionArticles();
        break;
    }
  },
  mounted() {
    switch (this.objectType) {
      case "hiuser":
        this.getMostRegisteredUserArticle();
        break;
      case "louser":
        this.getLeastRegisteredUserArticle();
        break;
      case "oldest":
        this.getLongestHistoryArticle();
        break;
      case "youngest":
        this.getYoungestHistoryArticle();
        break;
    }
  },
  watch: {
    counter: function(value) {
      if (/^\d+$/.test(value) && value > 0) {
        //console.log(value);
        switch (this.objectType) {
          case "hirev":
            this.getHighestRevisionArticles();
            break;
          case "lorev":
            this.getLowestRevisionArticles();
            break;
        }
      }
    }
  }
};
</script>
```

```
<style>
.box {
 width: 100%;
 border-style: solid;
 border-width: 2px;
 border-color: black;
 border-radius: 10px;
 display: flex;
 flex-direction: row;
 flex-wrap: wrap;
 align-content: center;
}

.loader {
 display: flex;
 justify-content: center;
 align-content: center;
 width: 100%;
 margin: 20px;
}
</style>
```

client\components\Login.vue

```
<template>
 <v-container>
   <v-card class="elevation-12">
     <v-toolbar dark color="primary">
       <v-toolbar-title>Login form</v-toolbar-title>
     </v-toolbar>
     <v-card-text>
       <v-form ref="loginForm">
         <v-text-field v-model="email" prepend-icon="person" name="email"
label="Email" type="email" :rules="validationRules.email"></v-text-field>
         <v-text-field v-model="password" id="password" prepend-icon="lock"
name="password" label="Password" type="password"
:rules="validationRules.password"></v-text-field>
       </v-form>
     </v-card-text>
```

```
      <v-alert v-model="alert" dismissible v-if="alert" type="error">{{ error
}}</v-alert>
      <v-card-actions>
        <v-spacer></v-spacer>
        <v-btn color="primary" @click="login">Login</v-btn>
      </v-card-actions>
    </v-card>
  </v-container>
</template>

<script>
export default {
  data: () => ({
    email: "",
    password: "",
    alert: false,
    error: "",
    validationRules: {
      email: [v => !!v || "Required"],
      password: [v => !!v || "Required"]
    }
  }),
  methods: {
    async login() {
      if (this.$refs.loginForm.validate()) {
        try {
          await this.$store.dispatch("user/login", {
            email: this.email,
            password: this.password
          });

          this.$router.push("/overview");
        } catch (e) {
          this.alert = true;
          this.error = e.response.data.message;
        }
      }
    }
  }
};
</script>
```

```
<style>
</style>
```

client\components\Navbar.vue

```
<template>
 <v-toolbar class="primary text--white" fixed app>
   <nuxt-link class="pageTitle" to="/">
     <v-layout row>
       <v-icon large left color="white">trending_up</v-icon>
       <v-toolbar-title class="ml-0 white--text" v-text="title"/>
     </v-layout>
   </nuxt-link>
   <v-spacer/>
   <v-toolbar-items>
     <template v-if="isAuthenticated && loggedInUser">
       <v-btn active-class="btnActive" class="white--text" flat
to="/overview">Overview</v-btn>
       <v-btn active-class="btnActive" class="white--text" flat
to="/article">Individual Article</v-btn>
       <v-btn active-class="btnActive" class="white--text" flat
to="/author">Author</v-btn>
     </template>
   </v-toolbar-items>
   <v-spacer/>
   <v-toolbar-items>
     <v-tooltip bottom v-if="isAuthenticated && loggedInUser">
       <template v-slot:activator="{ on }">
         <v-btn class="white--text" v-on="on" flat @click="logout">
           {{ loggedInUser.data.userName }}
           <v-icon right color="red">power_settings_new</v-icon>
         </v-btn>
       </template>
       <span>Logout</span>
     </v-tooltip>

     <template v-if="!isAuthenticated || !loggedInUser">
       <v-btn active-class="btnActive" class="white--text" flat
to="/register">Register</v-btn>
       <v-btn active-class="btnActive" class="white--text" flat
```

```
to="/login">Login</v-btn>
      </template>
    </v-toolbar-items>
  </v-toolbar>
</template>

<script>
import { mapGetters } from "vuex";

export default {
  data: () => ({
    title: "Article Insight"
  }),
  computed: {
    ...mapGetters({
      isAuthenticated: "user/isAuthenticated",
      loggedInUser: "user/loggedInUser"
    })
  },
  methods: {
    async logout() {
      await this.$store.dispatch("user/logout");

      this.$router.push("/");
    }
  }
};
</script>

<style scoped>
.pageTitle {
  text-decoration: none;
  text-decoration-color: none;
  color: black;
}
.btnActive {
  background-color: white;
  color: black !important;
  box-shadow: none !important;
}
</style>
```

client\components\PieChart.vue

```
<template>
 <v-card>
   <v-card-text v-if="type === 'overview'">
     <chartjs-pie :labels="xlabels" :datasets="xdatasets" :option="option"
:bind="true"></chartjs-pie>
   </v-card-text>
   <v-card-text v-else>
     <chartjs-pie :labels="ylabels" :datasets="ydatasets" :option="option"
:bind="true"></chartjs-pie>
   </v-card-text>
 </v-card>
</template>

<script>
export default {
 props: ["type", "title", "yFrom", "yTo", "change"],
 data: () => ({
   xlabels: [],
   xdatasets: [
     {
       data: [],
       backgroundColor: [
         "rgba(255, 99, 132, 0.5)",
         "rgba(54, 162, 235, 0.5)",
         "rgba(255, 206, 86, 0.5)",
         "rgba(75, 192, 192, 0.5)"
       ],
       hoverBackgroundColor: [
         "rgba(255, 99, 132, 1)",
         "rgba(54, 162, 235, 1)",
         "rgba(255, 206, 86, 1)",
         "rgba(75, 192, 192, 1)"
       ]
     }
   ],
   ylabels: [],
   ydatasets: [
```

```
      {
        data: [],
        backgroundColor: [
          "rgba(255, 99, 132, 0.5)",
          "rgba(54, 162, 235, 0.5)",
          "rgba(255, 206, 86, 0.5)",
          "rgba(75, 192, 192, 0.5)"
        ],
        hoverBackgroundColor: [
          "rgba(255, 99, 132, 1)",
          "rgba(54, 162, 235, 1)",
          "rgba(255, 206, 86, 1)",
          "rgba(75, 192, 192, 1)"
        ]
      }
    ],
    option: {},
    loaded: false
  }),
  methods: {
    async getRevisionsByUserType() {
      const data = await this.$axios.$get("revisions/getRevisionsByUserType",
{
        headers: {
          "x-access-token": this.$store.state.user.authUser.data.token
        }
      });

      for (var i = 0; i < data.data.length; i++) {
        this.xlabels.push(data.data[i]._id);

        data.data[i].admin_revisions
          ?
this.xdatasets[0].data.push(parseInt(data.data[i].admin_revisions))
          : null;
        data.data[i].bot_revisions
          ? this.xdatasets[0].data.push(parseInt(data.data[i].bot_revisions))
          : null;
        data.data[i].anon_revisions
          ?
this.xdatasets[0].data.push(parseInt(data.data[i].anon_revisions))
```

```
          : null;
        data.data[i].reg_revisions
          ? this.xdatasets[0].data.push(parseInt(data.data[i].reg_revisions))
          : null;
      }

      this.$emit("loaded", true);
    },
    async getIndividualRevisionsByUserType() {
      const data = await this.$axios.$get(
        "revisions/getArticleRevsByUserType",
        {
          headers: {
            "x-access-token": this.$store.state.user.authUser.data.token
          },
          params: {
            title: this.title,
            fromyear: this.yFrom ? this.yFrom : "1970-01-01",
            toyear: this.yTo ? this.yTo : new Date().toISOString().substr(0,
10)
          }
        }
      );

      for (var i = 0; i < data.data.length; i++) {
        this.ylabels.push(data.data[i]._id);

        if (data.data[i].admin_revisions) {

this.ydatasets[0].data.push(parseInt(data.data[i].admin_revisions));
        }
        if (data.data[i].bot_revisions) {
          this.ydatasets[0].data.push(parseInt(data.data[i].bot_revisions));
        }
        if (data.data[i].anon_revisions) {
          this.ydatasets[0].data.push(parseInt(data.data[i].anon_revisions));
        }
        if (data.data[i].regular_revisions) {

this.ydatasets[0].data.push(parseInt(data.data[i].regular_revisions));
        }
```

```
      }

      this.$emit("loaded", true);
    }
  },
  created() {
    if (this.type === "overview") {
      this.getRevisionsByUserType();
    } else {
      this.getIndividualRevisionsByUserType();
    }
  },
  watch: {
    title: function(nvalue, ovalue) {
      if (nvalue !== ovalue) {
        this.ylabels = [];
        this.ydatasets[0].data = [];
        this.getIndividualRevisionsByUserType();
      }
    },
    yFrom: function(nvalue, ovalue) {
      if (nvalue !== ovalue) {
        this.ylabels = [];
        this.ydatasets[0].data = [];
        this.getIndividualRevisionsByUserType();
      }
    },
    yTo: function(nvalue, ovalue) {
      if (nvalue !== ovalue) {
        this.ylabels = [];
        this.ydatasets[0].data = [];
        this.getIndividualRevisionsByUserType();
      }
    }
  }
};
</script>

<style>
</style>
```

client\components\Registration.vue

```
<template>
 <v-container>
   <v-card class="elevation-12">
     <v-toolbar dark color="primary">
       <v-toolbar-title>Registration</v-toolbar-title>
     </v-toolbar>
     <v-card-text>
       <v-form lazy-validation ref="regForm">
         <v-text-field name="firstName" label="First Name" type="text" v-
model="firstName" required :rules="validationRules.name"></v-text-field>
         <v-text-field name="lastName" label="Last Name" type="text" v-
model="lastName" required :rules="validationRules.name"></v-text-field>
         <v-text-field name="email" label="Email" type="email" v-
model="email" required :rules="validationRules.email"></v-text-field>
         <v-text-field id="password" name="password" label="Password"
type="password" v-model="password" required
:rules="validationRules.password"></v-text-field>
       </v-form>
     </v-card-text>
     <v-alert dismissible v-if="error" type="error">{{ errorMessage }}</v-
alert>
     <v-card-actions>
       <v-btn flat small color="error" @click="resetForm">Reset</v-btn>
       <v-spacer></v-spacer>
       <v-btn color="primary" @click="register">Register</v-btn>
     </v-card-actions>
   </v-card>
 </v-container>
</template>

<script>
export default {
 data: () => ({
   firstName: "",
   lastName: "",
   email: "",
   password: "",
   error: null,
   errorMessage: "",
```

```
  validationRules: {
    name: [
      v => !!v || "Required",
      v =>
        /^[a-zA-Z. ]+$/.test(v) ||
        "Name cannot contain numbers or special charachters."
    ],
    email: [
      v => !!v || "Required",
      v =>

/^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-
9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-
Z]{2,}))$/.test(
          v
        ) || "Invalid email format"
    ],
    password: [
      v => !!v || "Required",
      v => v.length >= 8 || "Password must be at least 8 characters long",
      v =>
        /[A-Z]/.test(v) ||
        "Password must contain at least one uppercase character",
      v =>
        /[a-z]/.test(v) ||
        "Password must contain at least one lowercase character",
      v =>
        /\d/.test(v) || "Password must contain at least one numeric
character"
    ]
  }
 }),

 methods: {
  resetForm() {
    this.firstName = "";
    this.lastName = "";
    this.email = "";
    this.password = "";

    this.$refs.regForm.resetValidation();
```

```
  },
  async register() {
    if (this.$refs.regForm.validate()) {
      try {
        const response = await this.$axios.post("user/register", {
          firstName: this.firstName,
          lastName: this.lastName,
          email: this.email,
          password: this.password
        });

        if (response.status !== "error") {
          await this.$store.dispatch("user/login", {
            email: this.email,
            password: this.password
          });

          this.$router.push("/overview");
        } else {
          this.error = true;
          this.errorMessage = response.message;
        }
      } catch (error) {
        this.error = true;
        this.errorMessage = error.response.data.message
          ? error.response.data.message
          : error.message;
      }
    }
  }
 }
};
</script>

<style>
</style>
```

client\pages\article.vue

```
<template>
```

```
 <v-layout row>
   <v-flex class="mr-5 mt-2" sm2>
     <v-layout column>
       <h4 class="ml-2">Filter By Date</h4>
       <DatePicker @picked="setFrom" title="From" @cleared="resetFrom"/>
       <DatePicker @picked="setTo" title="To" @cleared="resetTo"/>
       <!-- <v-btn color="primary" flat>Get Revisions</v-btn> -->
     </v-layout>
   </v-flex>
   <v-container>
     <v-toolbar dark color="primary">
       <v-toolbar-title class="mr-4">
         <v-icon>search</v-icon>
       </v-toolbar-title>
       <v-autocomplete class="mt-2" v-model="title" :hint="'Type to search'"
:items="titles" :label="'Search...'" item-text="_id" item-value="_id" flat
solo-inverted>
         <template v-slot:selection="titles">
           <v-list-tile-content>
             <v-list-tile-title v-html="titles.item._id"></v-list-tile-
title>
             <v-list-tile-sub-title v-html="'Revisions: ' +
titles.item.revisions"></v-list-tile-sub-title>
           </v-list-tile-content>
         </template>
         <template v-slot:item="titles">
           <v-list-tile-content>
             <v-list-tile-title v-html="titles.item._id"></v-list-tile-
title>
             <v-list-tile-sub-title v-html="'Revisions: ' +
titles.item.revisions"></v-list-tile-sub-title>
           </v-list-tile-content>
         </template>
       </v-autocomplete>
     </v-toolbar>
     <ArticleContainer ref="arc" :articleTitle="title" :yearFrom="yFrom"
:yearTo="yTo" :update="update"/>
     <ChartContainer v-if="show" type="individual" :title="title"
:yearFrom="yFrom" :yearTo="yTo" :update="update"/>
   </v-container>
 </v-layout>
```

```
</template>

<script>
import ArticleContainer from "~/components/ArticleContainer.vue";
import ChartContainer from "~/components/ChartContainer.vue";
import DatePicker from "~/components/DatePicker.vue";

export default {
 middleware: ["auth"],
 components: {
   ArticleContainer,
   ChartContainer,
   DatePicker
 },
 data: () => ({
   titles: [],
   title: null,
   counter: 0,
   loading: false,
   yFrom: null,
   yTo: null,
   update: false,
   off: false,
   show: false
 }),
 methods: {
   async getAllTitles() {
     const data = await this.$axios.$get("revisions/uniqueTitles", {
       headers: {
         "x-access-token": this.$store.state.user.authUser.data.token
       }
     });

     if (data) {
       this.titles = data.data;
     }
   },
   setFrom(value) {
     this.yFrom = value;
   },
   setTo(value) {
```

```
      this.yTo = value;
    },
    resetFrom() {
      this.yFrom = null;
    },
    resetTo() {
      this.yTo = null;
    }
  },
  beforeCreate() {
    this.$store.dispatch("user/initializeStore");
  },
  created() {
    this.getAllTitles();
  },
  watch: {
    title: function(nvalue, oldvalue) {
      {
        if (nvalue !== oldvalue) {
          this.update = true;
          this.show = true;
        }
      }
    }
  }
};
</script>

<style>
</style>
```

client\pages\author.vue

```
<template>
  <v-container>
    <v-toolbar dark color="primary">
      <v-toolbar-title class="mr-4">
        <v-icon>search</v-icon>
      </v-toolbar-title>
      <v-autocomplete class="mt-2" v-model="author" :hint="'Type to search'"
```

```
:items="authors" :label="'Search...'" item-text="_id" item-value="_id" flat
solo-inverted>
       <template v-slot:selection="authors">
         <v-list-tile-content>
           <v-list-tile-title v-html="authors.item._id"></v-list-tile-title>
           <v-list-tile-sub-title v-html="'Revisions: ' +
authors.item.user"></v-list-tile-sub-title>
         </v-list-tile-content>
       </template>
       <template v-slot:item="authors">
         <v-list-tile-content>
           <v-list-tile-title v-html="authors.item._id"></v-list-tile-title>
           <v-list-tile-sub-title v-html="'Revisions: ' +
authors.item.user"></v-list-tile-sub-title>
         </v-list-tile-content>
       </template>
     </v-autocomplete>
   </v-toolbar>
   <v-card v-if="author">
     <v-card-title>
       <h2>Aricles by Author</h2>
     </v-card-title>
     <v-card-text>
       <v-list>
         <v-list-group v-for="item in authorRecords" :key="item._id" v-
model="item.active" no-action>
           <template v-slot:activator>
             <v-list-tile>
               <v-list-tile-content>
                 <v-list-tile-title>Article: {{ item._id }}, Revisions: {{
item.no_revisions }}</v-list-tile-title>
               </v-list-tile-content>
             </v-list-tile>
           </template>

           <v-list-tile v-for="subItem in item.timestamp_list"
:key="subItem">
             <v-list-tile-content>
               <v-layout row>
                 <v-list-tile-avatar>
                   <v-icon>arrow_right</v-icon>
```

```
                </v-list-tile-avatar>
                <v-list-tile-title>{{ new Date(subItem) }}</v-list-tile-
title>
              </v-layout>
            </v-list-tile-content>
          </v-list-tile>
          <v-divider></v-divider>
        </v-list-group>
      </v-list>
    </v-card-text>
  </v-card>
 </v-container>
</template>

<script>
export default {
 middleware: ["auth"],
 data: () => ({
   authors: [],
   author: null,
   authorRecords: []
 }),
 methods: {
   async getAuthorList() {
     this.authors = [];
     this.author = null;
     const data = await this.$axios.$get("revisions/getAllAuthors", {
       headers: {
         "x-access-token": this.$store.state.user.authUser.data.token
       }
     });

     if (data) {
       this.authors = data.data;
     }
   },
   async getAuhorRecords() {
     this.authorRecords = [];
     if (this.author) {
       console.log(this.author);
       const data = await this.$axios.$get("revisions/getArticlesByAuthor",
```

```
{
          headers: {
            "x-access-token": this.$store.state.user.authUser.data.token
          },
          params: {
            author: this.author
          }
        });

        if (data) {
          this.authorRecords = data.data;
        }
      }
    }
  },
  beforeCreate() {
    this.$store.dispatch("user/initializeStore");
  },
  created() {
    this.getAuthorList();
  },
  watch: {
    author: function(nvalue, ovalue) {
      if (nvalue !== ovalue) {
        this.getAuhorRecords();
      }
    }
  }
};
</script>

<style>
</style>
```

client\pages\index.vue

```
<template>
  <v-layout column justify-center align-center>
    <v-flex xs12 sm8 md6>
      <h1>Welcome to Article Insight</h1>
```

```
      <v-img src="~/assets/IMAGE1.png" aspect-ratio="1"></v-img>
      <v-img src="~/assets/IMAGE2.png" aspect-ratio="1"></v-img>
    </v-flex>
  </v-layout>
</template>

<script>
export default {
  middleware: "guest",
  components: {},
  beforeCreate() {
    this.$store.dispatch("user/initializeStore");
  }
};
</script>
```

client\pages\login.vue

```
<template>
  <v-layout align-center justify-center>
    <v-flex xs12 sm8 md8>
      <Login/>
    </v-flex>
  </v-layout>
</template>

<script>
import Login from "~/components/Login.vue";

export default {
  middleware: "guest",
  components: {
    Login
  }
};
</script>
```

client\pages\overview.vue

```
<template>
```

```
 <v-layout row>
   <v-flex class="mr-5" xs1>
     <h4>No. of Articles</h4>
     <v-text-field v-model="counter" :disabled="loading" label="Number"
outline></v-text-field>
   </v-flex>
   <v-layout column>
     <v-flex>
       <HorizontalCards objectType="hirev" objectHeader="Articles with
Highest Revisions" :counter="counter" @loading="loadingData"/>
       <v-divider></v-divider>
       <HorizontalCards objectType="lorev" objectHeader="Articles with
Lowest Revisions" :counter="counter" @loading="loadingData"/>
     </v-flex>
     <v-divider></v-divider>
     <v-layout row>
       <HorizontalCards objectType="hiuser" objectHeader="Article with Most
Registered Users" :counter="counter"/>
       <HorizontalCards objectType="louser" objectHeader="Article with Least
Registered Users" :counter="counter"/>
     </v-layout>
     <v-divider></v-divider>
     <v-layout row>
       <HorizontalCards objectType="oldest" objectHeader="Articles with
Longest History" :counter="counter"/>
       <HorizontalCards objectType="youngest" objectHeader="Article with
Shortest History" :counter="counter"/>
     </v-layout>
     <keep-alive>
       <ChartContainer type="overview"/>
     </keep-alive>
   </v-layout>
 </v-layout>
</template>

<script>
import HorizontalCards from "~/components/HorizontalCards.vue";
import ChartContainer from "~/components/ChartContainer";

export default {
 middleware: "auth",
```

```
 components: {
   HorizontalCards,
   ChartContainer
 },
 data: () => ({
   counter: 2,
   loading: false
 }),
 beforeCreate() {
   this.$store.dispatch("user/initializeStore");
 },
 methods: {
   loadingData(value) {
     this.loading = value;
   }
 }
};
</script>
```

client\pages\register.vue

```
<template>
 <v-layout align-center justify-center>
   <v-flex xs12 sm8 md8>
     <Registration/>
   </v-flex>
 </v-layout>
</template>

<script>
import Registration from "~/components/Registration.vue";

export default {
 middleware: "guest",
 components: {
   Registration
 }
};
</script>
```

client\layouts\default.vue

```vue
<template>
 <v-app light>
   <Navbar/>
   <v-container>
     <v-content class="px-4 py-5 my-5">
       <transition name="fade" mode="in-out" appear>
         <nuxt ref="page"/>
       </transition>
     </v-content>
   </v-container>
   <v-footer class="customFooter elevation-4" :fixed="fixed" app>
     <v-divider/>
     <span>&copy; 2019</span>
   </v-footer>
 </v-app>
</template>

<script>
import Navbar from "~/components/Navbar.vue";

export default {
 data() {
   return {
     fixed: false
   };
 },
 components: {
   Navbar
 }
};
</script>

<style scoped>
.fade-enter-active,
.fade-leave-active {
 transition: opacity 0.8s;
}
.fade-enter, .fade-leave-to /* .fade-leave-active below version 2.1.8 */ {
 opacity: 0;
}
```

```
.customFooter {
 border-top: 3px;
 border-top-color: black;
}
</style>
```

client\middleware\auth.js

```
export default function({ store, redirect }) {
 if (!store.state.user.loggedIn) {
   return redirect("/");
 }
}
```

client\middleware\guest.js

```
export default function({ store, redirect }) {
 if (store.state.user.loggedIn) {
   return redirect("/overview");
 }
}
```