THE UNIVERSITY OF SYDNEY

School of Information Technologies
Faculty of Engineering & IT

ASSIGNMENT/PROJECT COVERSHEET - GROUP ASSESSMENT

Unit of Study: COMP5347 Web Application Development

Assignment name: Group Assignment-Technical Report

Tutorial time: 8pm - 9pm **Tutor name:** Johan Alibasa

DECLARATION
We the undersigned declare that we have read and understood the *University of Sydney Academic Dishonesty and Plagiarism in Coursework Policy*, an, and except where specifically acknowledged, the work contained in this assignment/project is our own work, and has not been copied from other sources or been previously submitted for award or assessment.

We understand that failure to comply with the *Academic Dishonesty and Plagiarism in Coursework Policy* can lead to severe penalties as outlined under Chapter 8 of the *University of Sydney By-Law 1999* (as amended). These penalties may be imposed in cases where any significant portion of my submitted work has been copied without proper acknowledgement from other sources, including published works, the internet, existing programs, the work of other students, or work previously submitted for other awards or assessments.

We realise that we may be asked to identify those portions of the work contributed by each of us and required to demonstrate our individual knowledge of the relevant material by answering oral questions or by undertaking supplementary work, either written or in the laboratory, in order to arrive at the final assessment mark.

Group-25	**Project team members**			
Student name	**Student ID**	**Participated**	**Agree to share**	**Signature**
1. Madhavan Sundararaj	480490834	Yes / No ✓	Yes/No ✓	Madhavan
2. Shashank Jain	480448754	Yes / No ✓	Yes / No ✓	Shashank
3. Giridhar Aynampudi	480452809	Yes / No ✓	Yes / No ✓	giridhar
4.		Yes / No	Yes / No	
5.		Yes / No	Yes / No	
6.		Yes / No	Yes / No	
7.		Yes / No	Yes / No	
8.		Yes / No	Yes / No	
9.		Yes / No	Yes / No	
10.		Yes / No	Yes / No	

SIT Building, J12
The University of Sydney
NSW 2006 Australia

T +61 2 9351 3423
F +61 2 9351 3838
E sit.info@sydney.edu.au
sydney.edu.au/it

ABN 15 211 513 464
CRICOS 00026A

Table of Contents

1. Introduction

Our project is to build a data analytic web application which is implemented in a typical three-tier web application. The given dataset contains revision histories of Wikipedia articles in json format which has many properties, but only title, timestamp, user and anon are used for this analysis. The dataset also contains extra files such as four administrator type (active admins, semi-active admins, inactive admins, and former admins) files which contains list of all admins of English Wikipedia and one bot file containing list of all bot users.

The project is hosted on localhost:4200, and the main page consists of article insight which shows the basic analytics such as articles with highest number of revisions, articles with lowest number of revisions, etc. Sign up and login are also part of main page where user must sign up to access advanced analytics by giving first name, last name, email (username) and password. The user credentials are stored in database. Once user is registered, he can login by using username and password. After validation, successful users can interact with all analytics. Also, logout option is prompted to the user at the right corner of the web application page.

The stack we used in our Project is:

1. Angular 7 – Frontend / Client
2. Node.Js – Server
3. MongoDB – Backend / Database

Angular 7 is written in Typescript which is a superset of JavaScript. This report describes the design and architecture of the web application of each tier (Application, Presentation and Database) and briefly about the performance and security. Source code is also added in the appendix.

2. Design and Architecture of the Web Application

The application is built based on MVC architecture which separates application into three components - Model, View and Controller.

2.1 Model:

Model represents shape of the data and business logic. It is used to interact with the data in the database of the application. Model objects retrieve data and store in the database. MongoDB is used as back-end database where the Wikipedia dataset is imported, and user credentials are stored. Model is also used to write frequent queries to reduce code redundancy in controller. We used mongoose, which is a mongo dB object modelling for node.js to connect mongo DB.

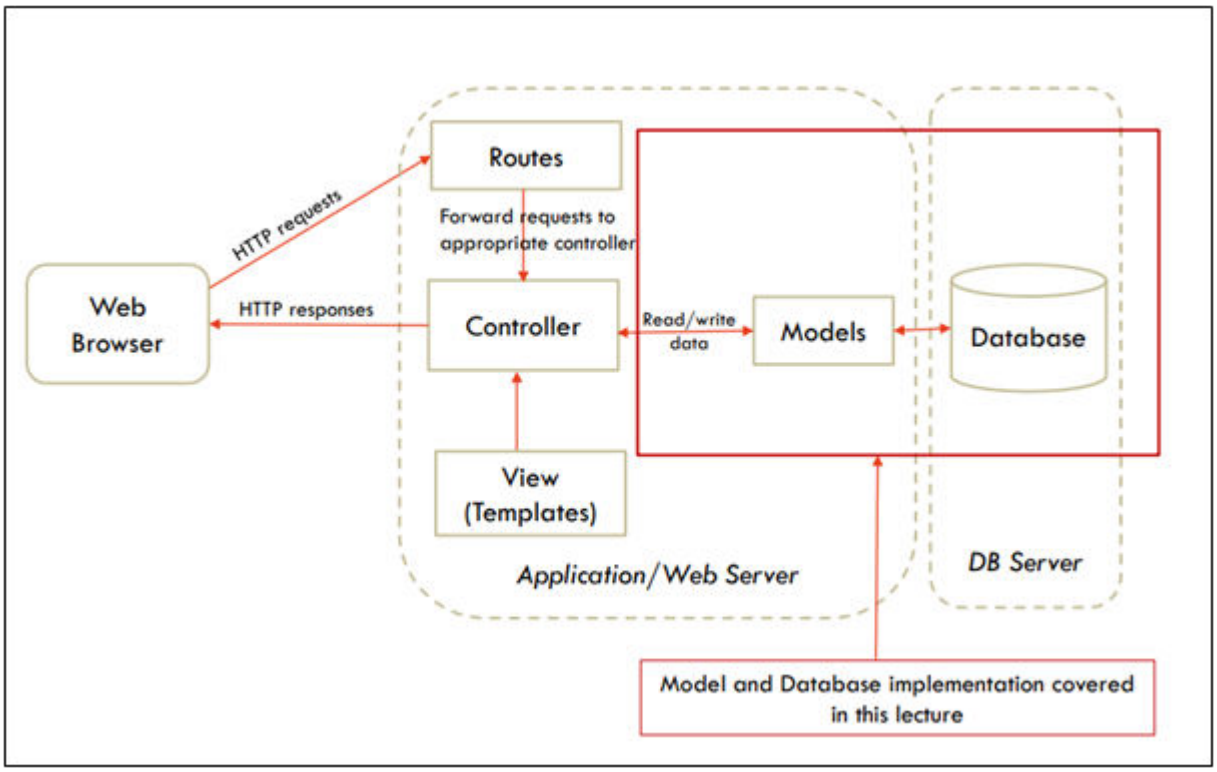


Figure illustrates the MVC architecture

2.2 View:

View is a user interface or presentation tier which allows user to interact with the data and send a request to controller if the data is modified and gets updated data from controller. We used Angular 7 as our view, which is JavaScript framework with HTML as template and CSS as style sheet. The main reason for choosing Angular 7 is because it has good UI components such as mat-card, mat-autocomplete, mat-spinner, etc. It also has good community in the web, and it allowed us to create a beautiful web application.

2.3 Controller:

Controller handles the user request coming from view. Typically, user interacts with View, which in-turn raises appropriate URL request, this request will be handled by a controller. The controller sends the response to the appropriate view with the result from model. Node JS with Express as framework is used for asynchronous communication between the components.

Overall analytics, Individual article analytics and author analytics are performed on single page where user interacts with application and request is sent to the relevant controller with use of routes. Routes are used in both angular as well as node.js to forward user requests from client to appropriate controller. We created a separate code file (below figure) to connect routes from angular to node.js. This code states that, any request (GET or POST) from angular with prefix as "/api/" should be forwarded to localhost: 3000, which is the server port of node.js/express. This code file is called using package.json script.

```
{
  "/api/*": {
    "target": "http://localhost:3000",
    "secure": false,
    "logLevel": "debug"
  }
}
```

The data retrieved from the model is logically sent to controller. For Example, Titles of the two articles with highest number of revisions is displayed on the page by following steps:

1. After the user enters the credentials in the login page, username and password are passed to the controller.
2. Controller verifies the credentials with help of model and sends acknowledgement based on verification to the view.
3. The view routes the user to the overall analytics page and displays the titles of two article with highest number of revisions on successful acknowledgement and throws an error in the login page if credentials are not matched.

2.4 Performance:

- Usually, it takes few seconds for user to sign-up and sign-in and access the main page, hence, we wrote a code in chart.service.ts to fetch the contents of bar chart and pie chart of overall analytics page during the registration of user and store it as cache in browser using local storage service provided by a famous JavaScript library called ngrx-webstorage, so that the contents are shown as soon as user routes to overall analytics page. We implemented this code because, it took 3 and 8 seconds for pie chart and bar chart respectively to receive data from controller.

- In model, we created index for respective schemas (revisions and users) with frequently accessed data as property, to increase the model performance. We also wrote static and instance methods for frequently accessed queries to get reduce code redundancy in controller. Below figure shows the index created and static, instance method used for both revision and user schemas.

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

RevisionSchema.statics.findTopArticles = function(sort, callback) {
    return this.aggregate([
        {$group: {_id: "$title", rev:{$sum: 1}}},
        {$sort: {rev: sort}},
    ]).exec(callback);
}

RevisionSchema.statics.findTitleLatestRev = function(title, callback){

    return this.find({'title':title})
    .sort({'timestamp':-1})
    .limit(1)
    .exec(callback)
}

RevisionSchema.statics.topTwoArticlesLongestHistory = function(callback){
    return this.aggregate([
    {$group:{_id:"$title",timestamp:{"$min":"$timestamp"}}},
    {$sort:{timestamp:1}},
    {$limit:2}
]).exec(callback);
}
```

```
UserSchema.index({email: 1});

UserSchema.methods.createPasswordHash = function(password) {
    this.saltRounds = 12;
    this.passwordHash = bcrypt.hashSync(password, this.saltRounds);
  };

UserSchema.methods.checkPassword = function(password) {
    return bcrypt.compareSync(password, this.passwordHash);
};
```

2.5 Security

We implemented security in all three tiers:

- In Model, we used bcrypt library to convert user's password into hash and store them, instead of directly storing the user's password during registration. And during login, user's entered password will be converted into hash and compared with stored passwordHash by bcrypt. We chose bcrypt because it's the most trusted password hashing library among the developer community. Below figure shows the format in which password is stored in database.

passwordHash	$2b$12$2v.C.hQvw7J5rMLJVK4hrevBSGdeV62QF/WHXszLXnO6oIg.26Pf6

- In View, we created a code file named auth.guard.ts which is an Angular's method of route authentication. Using auth.guard.ts, only logged in users can view the analytics page. If the page is refreshed or user routes to a particular page without logging in, auth.guard.ts file will redirect them to register page. We are using "CanActivate" from angular to code auth.guard.ts file and guard the route. Below figures shows the code implementation and configuration of auth.guard.ts in routes.

```
export class AuthGuard implements CanActivate {
  constructor(private chartService: ChartService, private router: Router) {}

  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot
  ): Observable<boolean> | Promise<boolean> | boolean {
    return this.chartService.userObservable.pipe(
      take(1),
      map(user => {
        const authenticated: boolean = !!user;
        if (!authenticated) {
            this.router.navigate(['/register']);
            return false;
        } else {
          return true;
        }
      })
    );
  }
}
```

```
{
  path: 'overall',
  component: OverallAnalyticsComponent,
  data: { title: 'Overall Analytics' },
  canActivate: [AuthGuard]
},
```

- In Controller, we are using express-session library to check each request coming from view to controller and confirm that the user is logged in. If the session is not created or user is not logged in, it sends unauthorized access message to view. Session will be created when a user successfully logs in through "/login" route. Below figure is one of the examples of session initiation and implementation in controller.

```
app.use(session({
  secret: 'secret',
  resave: false,
  saveUninitialized: true,
  cookie: { secure: true }
}));
```

```
function larGroupOfRegisteredUser(req,res){
  if(req.session) {
    Revision.aggregate([
      {$match:{anon:{$exists:false}}},
            {$group:{_id:"$title",rev:{$sum:1},uniqueValues:{$addToSet:"$user"}}},
            {$project: {title: 1, rev: 1, numberOfUsers: {$cond: {if: {$isArray: "$uniqueValues"}, then: {$size: "$uniqueValues"}, else: "NA"}}}},
            {$sort: {numberOfUsers:-1}},
            {$limit:1}
    ], function(err, result) {
      if(err) {
        console.log(err);
      } else {
        return res.status(200).send(result);
      }
    });
  } else {
    return res.status(500).send({message: 'Unauthorized Access'});
  }
}
```

APPENDIX

Procedure to run our code:

1. Unzip the folder and open in any coding editor (VS code is recommended).
2. Open terminal and cd to folder and if you are using VS code, open integrated terminal.
3. npm install on root folder (Web Dev 25).
4. After successful installation of npm, cd to server,
5. Again, run command npm install.
6. After completion, run command cd .. to go back to previous folder and run command npm run dev.
7. Wait until you get the message “Complied Successfully.” After that open localhost: 4200 in your browser.
8. You will see the home page, register and login to continue browsing the page.

Our code for each component is as follows:

Model:

db.js

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

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

module.exports = mongoose;

```

revision.js

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

var RevisionSchema = new mongoose.Schema(
        {title: {type:String},
        timestamp:{type:Date},
        user:{type:String},
        anon:{type:String}},
        {
            versionKey: false
        });


RevisionSchema.index({title: 1, user: 1});

RevisionSchema.statics.findTopArticles = function(sort, callback) {
    return this.aggregate([
        {$group: {_id: "$title", rev:{$sum: 1}}},
        {$sort: {rev: sort}},
    ]).exec(callback);
}

RevisionSchema.statics.findTitleLatestRev = function(title, callback){

    return this.find({'title':title})
    .sort({'timestamp':-1})
    .limit(1)
    .exec(callback)
}

RevisionSchema.statics.topTwoArticlesLongestHistory = function(callback){
    return this.aggregate([
    {$group:{_id:"$title",timestamp:{"$min":"$timestamp"}}},
    {$sort:{timestamp:1}},
    {$limit:2}
]).exec(callback);
}




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

module.exports = Revision
```

User.js

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

var UserSchema = new mongoose.Schema({
    firstName: { type: String, required: true },
    lastName:{ type: String, required: true },
    email: { type: String, required: true },
    passwordHash:{ type: String, required: true },
},
    {
    versionKey: false
});

UserSchema.index({email: 1});

UserSchema.methods.createPasswordHash = function(password) {
    this.saltRounds = 12;
    this.passwordHash = bcrypt.hashSync(password, this.saltRounds);
  };

UserSchema.methods.checkPassword = function(password) {
    return bcrypt.compareSync(password, this.passwordHash);
};

var User = mongoose.model('User', UserSchema, 'users');

module.exports = User;
```

View:

index.html

```html
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Web Application</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
</head>
<body>
  <app-root></app-root>
</body>
</html>

```

style.css

```
@import '~bootstrap/dist/css/bootstrap.min.css';
@import '@angular/material/prebuilt-themes/deeppurple-amber.css';

.body {
  background-color: #f5f5f5;
}
.mat-toolbar.mat-primary {
    background: #6C809A;
    color: #fff;
  }

.mat-raised-button.mat-primary {
  background: #6C809A;
}

.simple-form-aligner {
  display: flex;
  justify-content: center;
  padding-top: 26px;
}

.simple-form-card {
  width:100%;
}

.simple-form-button {
  align-self: flex-end;
}

.simple-form-field-100 {
  width: 100%;
}

.simple-form-field-50 {
  width: 49%;
}

.simple-form-spinner {
  display:unset !important;
  margin-left:7px;
}

```

app.component.html

```
<div>
  <mat-toolbar color="primary" class="top-navbar" role="header">
    <div>
      <button mat-icon-button (click)="toggleSideNavDrawer()">
        <mat-icon>
          menu
        </mat-icon>
      </button>
    </div>
    <span>
      <!-- Below line was added to prevent the mat-drawer form highlighting the logo with an orange box -->
      <a routerLink="/" (click)="hideSideNavAfterClick()"></a>
      <a routerLink="/" (click)="hideSideNavAfterClick()">
      </a>
      {{ title }}
    </span>
    <span class="navbar-spacer"></span>
    <app-user-dropdown></app-user-dropdown>
  </mat-toolbar>
  <mat-drawer-container>
    <!-- Side nav -->
    <mat-drawer #sideNavDrawer [mode]="drawerMode" [opened]="drawerOpened" class="lwa-dark-theme">
      <mat-nav-list>
        <a *ngFor="let navLink of primaryNavLinks" mat-list-item [hidden]="
            (navLink.isLoggedInRoute && !isLoggedIn)
            " routerLink="{{ navLink.routerLink }}" (click)="hideSideNavAfterClick()">
          <mat-icon>{{ navLink.icon }}</mat-icon>
          {{ navLink.title }}
        </a>
        <a *ngFor="let navLink of secondaryNavLinks" mat-list-item [hidden]="
            (navLink.isLoggedInRoute && !isLoggedIn)
            " routerLink="{{ navLink.routerLink }}" (click)="hideSideNavAfterClick()">
          <mat-icon>{{ navLink.icon }}</mat-icon>
          {{ navLink.title }}
        </a>
      </mat-nav-list>
    </mat-drawer>
    <!-- Top nav -->
    <mat-drawer-content>
      <!-- Route content -->
      <div class="container" #routerOutletParent>
        <router-outlet></router-outlet>
      </div>
    </mat-drawer-content>
  </mat-drawer-container>
</div>
```

app.component.css

```
mat-drawer-container {
  display: flex;
  flex-direction: column;
  position: absolute;
  top: 0;
  bottom: 0;
  left: 0;
  right: 0;
  margin-top: 64px;
}

.mat-drawer {
  width: 266px;
  background-color: lightgray;
  box-shadow: 0 8px 10px -5px rgba(0, 0, 0, .2), 0 16px 24px 2px rgba(0, 0, 0, .14), 0 6px 30px 5px rgba(0, 0, 0, .12);
  background-size: cover;
}
mat-drawer mat-icon {
  margin-right: 16px;
}
.top-navbar {
  position: -webkit-sticky;
  position: sticky;
  top: 0;
  z-index: 1000;
  box-shadow: 0 8px 10px -5px rgba(0, 0, 0, .2), 0 16px 24px 2px rgba(0, 0, 0, .14), 0 6px 30px 5px rgba(0, 0, 0, .12);
}
.top-navbar .logo {
  height: 30px;
  position: relative;
  margin-right: 8px;
}
.navbar-spacer {
  flex: 1 1 auto;
}
.bottom-mobile-nav {
  position: fixed;
  bottom: 0;
  display: flex;
  justify-content: space-around;
}
.container {
  height: calc(100vh - 64px);
  overflow: auto;
  margin-top: 10px;
}
```

app.component.ts

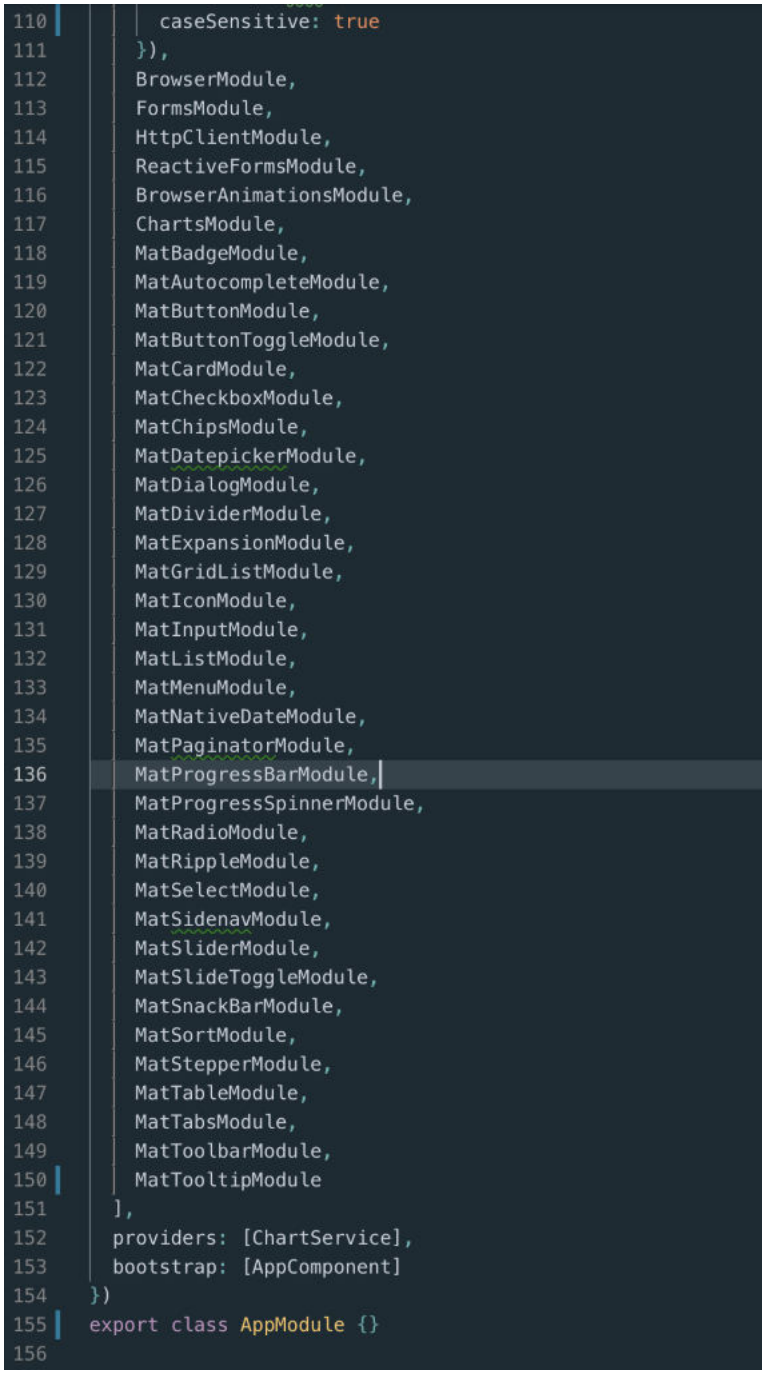
```ts
import { filter } from 'rxjs/operators';
import { Component, ViewChild, OnInit } from '@angular/core';
import {
  ActivatedRoute,
  NavigationStart,
  NavigationEnd,
  Router,
} from '@angular/router';
import { ChartService } from './chart.service';


@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit {
  @ViewChild('sideNavDrawer') sideNavDrawer;
  @ViewChild('routerOutletParent') routerOutletEle;
  screenWidth: number;
  mobileWidth = false;
  isLoggedIn: boolean;
  userIsAdmin: boolean;
  drawerMode: string;
  drawerOpened: boolean;

  siteTheme: string;

  // Scroll position maintainer
  private lastPoppedScrollTop: number;
  private currentRouteId: number;
  private yScrollStack: number[] = [];

  title = 'web-application';
  primaryNavLinks: NavLink[] = [
    {
      routerLink: '/',
      icon: 'home',
      title: 'Home',
    },
  ];

  secondaryNavLinks: NavLink[] = [
    {
      routerLink: '/overall',
      icon: 'create',
      title: 'Overall Analytics',
      isLoggedInRoute: true,
    },
    {
      routerLink: '/individual',
      icon: 'drafts',
      title: 'Individual Analytics',
      isLoggedInRoute: true,
    },
```

```
    {
      routerLink: '/author',
      icon: 'person',
      title: 'Author Analytics',
      isLoggedInRoute: true,
    },
  ];

  toggleSideNavDrawer() {
    this.sideNavDrawer.toggle();
  }

  hideSideNavAfterClick() {
    if (this.mobileWidth) {
      this.toggleSideNavDrawer();
    }
  }

  setSideBar() {
    if (this.screenWidth < 768) {
      this.drawerMode = 'over'; // push or over
      this.drawerOpened = false;
      this.mobileWidth = true;
    } else {
      this.drawerMode = 'side';
      this.drawerOpened = true;
      this.mobileWidth = false;
    }
  }

  constructor(
    private router: Router,
    private route: ActivatedRoute,
    private chartService: ChartService,
  ) {
    // Set side bar mode
    this.screenWidth = window.innerWidth;
    window.onresize = () => {
      this.screenWidth = window.innerWidth;
      this.setSideBar();
    };
    router.events
      .pipe(filter(e => e instanceof NavigationEnd))
      .forEach((e: NavigationEnd) => {
        this.title = route.root.firstChild.snapshot.data['title'];
      });

    this.chartService.userObservable.subscribe(user => {
      this.isLoggedIn = !!user;
    });
    this.setSideBar();
  }

  ngOnInit() {
    this.setSideBar();
```

```
111
112     this.router.events.subscribe((ev: any) => {
113       if (ev instanceof NavigationStart) {
114         if (this.routerOutletEle.nativeElement) {
115           // this has been placed here as a hack since this element is not ready on first load
116           const el = this.routerOutletEle.nativeElement;
117           this.yScrollStack[this.currentRouteId] = el.scrollTop;
118           // Determine if we are going back
119           if (
120             ev.restoredState &&
121             ev.restoredState.navigationId &&
122             this.yScrollStack[ev.restoredState.navigationId]
123           ) {
124             this.lastPoppedScrollTop = this.yScrollStack[
125               ev.restoredState.navigationId
126             ];
127           } else {
128             this.lastPoppedScrollTop = 0;
129           }
130         }
131       } else if (ev instanceof NavigationEnd) {
132         if (this.routerOutletEle.nativeElement) {
133           // this has been placed here as a hack since this element is not ready on first load
134           this.currentRouteId = ev.id;
135           const el = this.routerOutletEle.nativeElement;
136           el.scrollTop = this.lastPoppedScrollTop
137             ? this.lastPoppedScrollTop
138             : 0;
139         }
140       }
141     });
142   }
143 }
144
145 interface NavLink {
146   routerLink: string;
147   icon: string;
148   title: string;
149   isAdminRoute?: boolean;
150   isLoggedInRoute?: boolean;
151 }
152
```

app.module.ts

```
import { AuthorAnalyticsComponent } from "./author-analytics/author-analytics.component";
import { UserDropdownComponent } from "./user-dropdown/user-dropdown.component";

@NgModule({
  declarations: [
    AppComponent,
    LoginComponent,
    MyBarChartComponent,
    HomeComponent,
    OverallAnalyticsComponent,
    RegisterComponent,
    PieChartComponent,
    IndividualAnalyticsComponent,
    AuthorAnalyticsComponent,
    UserDropdownComponent
  ],
  imports: [
    NgbModalModule,
    RouterModule.forRoot([
      {
        path: "",
        component: HomeComponent,
        data: { title: "Home" },
        pathMatch: "full"
      },
      {
        path: "register",
        component: RegisterComponent,
        data: { title: "Register" }
      },
      { path: "login", component: LoginComponent, data: { title: "Login" } },
      {
        path: "overall",
        component: OverallAnalyticsComponent,
        data: { title: "Overall Analytics" },
        canActivate: [AuthGuard]
      },
      {
        path: "individual",
        component: IndividualAnalyticsComponent,
        data: { title: "Individual Analytics" },
        canActivate: [AuthGuard]
      },
      {
        path: "author",
        component: AuthorAnalyticsComponent,
        data: { title: "Author" },
        canActivate: [AuthGuard]
      },
      { path: "**", component: HomeComponent }
    ]),
    NgxWebstorageModule.forRoot({
      prefix: "app",
      separator: ".",
      caseSensitive: true
```



```
      caseSensitive: true
    }),
    BrowserModule,
    FormsModule,
    HttpClientModule,
    ReactiveFormsModule,
    BrowserAnimationsModule,
    ChartsModule,
    MatBadgeModule,
    MatAutocompleteModule,
    MatButtonModule,
    MatButtonToggleModule,
    MatCardModule,
    MatCheckboxModule,
    MatChipsModule,
    MatDatepickerModule,
    MatDialogModule,
    MatDividerModule,
    MatExpansionModule,
    MatGridListModule,
    MatIconModule,
    MatInputModule,
    MatListModule,
    MatMenuModule,
    MatNativeDateModule,
    MatPaginatorModule,
    MatProgressBarModule,
    MatProgressSpinnerModule,
    MatRadioModule,
    MatRippleModule,
    MatSelectModule,
    MatSidenavModule,
    MatSliderModule,
    MatSlideToggleModule,
    MatSnackBarModule,
    MatSortModule,
    MatStepperModule,
    MatTableModule,
    MatTabsModule,
    MatToolbarModule,
    MatTooltipModule
  ],
  providers: [ChartService],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

home.component.html

```
<mat-card>
  <div class="titleDiv">
    Insight on Wikipedia's featured Articles
  </div>
  <hr>
  <div class="overallDiv">
    <div class="barDiv">
      <img mat-card-image src="assets/img/overall_bar.png">
    </div>
    <div class="header">
      <mat-card-header>
        <mat-card-title class="title">Overall Analytics</mat-card-title>
        <mat-card-subtitle class="subtitle">This insight gives following information:</mat-card-subtitle>
      </mat-card-header>
      <ul class="listStyle">
        <li>Articles ranked by number of revisions</li>
        <li>Articles ranked by number of registered users</li>
        <li>Articles with longest and shortest history</li>
        <li>Revision number distribution by year and user type across whole dataset <span class="figureSpan">(Figure on left)</span></li>
        <li>Revision number distribution by user type across whole dataset</li>
      </ul>
    </div>
  </div>
  <div class="overallDiv">
    <div class="barDiv">
      <mat-card-header>
        <mat-card-title class="title">Individual Analytics</mat-card-title>
        <mat-card-subtitle class="subtitle">This insight gives following information:</mat-card-subtitle>
      </mat-card-header>
      <ul class="listStyle">
        <li>This insight consists of individual article information</li>
        <li>Total number of revisions of the article</li>
        <li>Top 5 regular users of article</li>
        <li>Revision number distribution by year and user type of the article</li>
        <li>Revision number distribution by user type of the article <span class="figureSpan">(Figure on right)</span></li>
        <li>Revision number distributed by year and one user</li>
      </ul>
    </div>
    <div class="barDiv">
      <img mat-card-image src="assets/img/individual_pie.png">
    </div>
  </div>
  <div *ngIf="!loggedIn" class="moreDiv">
    For advanced analytics, SignUp or Login
  </div>
  <mat-card-actions *ngIf="!loggedIn" align="end">
    <button mat-raised-button color="primary" (click)="login(modal)">Login</button>
    <button mat-raised-button color="primary" (click)="register()">Register</button>
    <ng-template #modal>
      <app-login></app-login>
    </ng-template>
  </mat-card-actions>
  <hr>
  <div>
    <mat-card-title class="teamHeading">Our Team</mat-card-title>
    <div class="teamDiv">
      <mat-card-header>
        <img mat-card-avatar class="imageDiv" src="assets/img/madh.JPG" />
        <div class="wrapper">
          <mat-card-subtitle class="nameDiv">Madhavan Sundararaj</mat-card-subtitle>
          <mat-card-subtitle>Frontend developer</mat-card-subtitle>
        </div>
      </mat-card-header>
      <mat-card-header>
        <img mat-card-avatar class="imageDiv" src="assets/img/sha.jpg" />
        <div class="wrapper">
          <mat-card-subtitle class="nameDiv">Shashank Jain</mat-card-subtitle>
          <mat-card-subtitle>Node developer</mat-card-subtitle>
        </div>
      </mat-card-header>
      <mat-card-header>
        <img mat-card-avatar class="imageDiv" src="assets/img/giri.jpg" />
        <div class="wrapper">
          <mat-card-subtitle class="nameDiv">Giridhar Aynampudi</mat-card-subtitle>
          <mat-card-subtitle>Backend developer</mat-card-subtitle>
        </div>
      </mat-card-header>
    </div>
  </div>
</mat-card>
```

home.component.css

```css
.titleDiv {
    text-align: center;
    font-weight: bolder;
    color: #6C809A;
    font-size: 30px;
}

img {
    margin-top: 0!important;
}

.overallDiv {
    width:100%;
    display: flex;
}

.barDiv {
    width: 50%;
    margin-right: 17px;
    object-fit: cover;
}

.header {
    width:50%;
}

.title {
    font-size: 26px;
}

.subTitle {
    font-size: 16px;
}

.listStyle {
    font-size: 20px;
}

.moreDiv {
    text-align:center;
    font-weight: bolder;
    font-style: italic;
    margin-bottom: -41px;
}

.teamHeading {
    text-align: center
}

.teamDiv {
    display: flex;
    justify-content: space-around;
}

.imageDiv {
    width: 85px;
    height: 85px;
}

.wrapper {
    margin-left: -23px;
    margin-top: 15px;
}

.nameDiv {
    color:black;
    font-weight: bolder
}

.figureSpan {
    font-style:italic;
    font-size: 16px;
    color: cadetblue;
}
```

home.component.ts

```typescript
import { Component, OnInit, OnDestroy } from '@angular/core';
import { NgbModal } from '@ng-bootstrap/ng-bootstrap';
import { ChartService } from '../chart.service';
import { Router } from '@angular/router';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit, OnDestroy {
  loginComp = false;
  modalReference = null;
  pieData: any;
  loggedIn = false;

  constructor(private modalService: NgbModal, private chartService: ChartService, private router: Router) { }

  ngOnInit() {
    this.chartService.getOverallPieData();
    this.chartService.getOverallBarData();
    this.chartService.userObservable.subscribe(user => {
      this.loggedIn = !!user;
    });
  }

  login(content) {
    this.modalReference = this.modalService.open(content, {backdropClass: 'light-blue-backdrop', centered: true});
  }

  register() {
    this.router.navigateByUrl('/register');
  }

  ngOnDestroy() {
    if (this.modalReference != null) {
      this.modalReference.close();
    }
  }

}

```

register.component.html

```
<mat-card>
  <div class="topDiv">
    <div class="imageDiv">
      <img mat-card-image class="imageStyle" src="assets/img/bg-01.jpg">
    </div>
    <div class="formDiv">
      <form [formGroup]="form" (ngSubmit)="submit(form.value)">
        <mat-card-title> Register </mat-card-title>
        <mat-card-subtitle>
          Already have an account?
          <a routerLink="/login">
            Log in here.
          </a>
        </mat-card-subtitle>
        <mat-card-content>
          <div class="form-group">
            <mat-form-field class="simple-form-field-100">
              <input matInput placeholder="First Name" name="firstName" formControlName="firstName" />
              <mat-error>Required</mat-error>
            </mat-form-field>
            <br>
            <mat-form-field class="simple-form-field-100">
              <input matInput placeholder="Last Name" name="lastName" formControlName="lastName" />
              <mat-error>Required</mat-error>
            </mat-form-field>
            <br>
            <mat-form-field class="simple-form-field-100">
              <input matInput placeholder="Email" name="email" formControlName="email" />
              <mat-error>Not a valid email</mat-error>
            </mat-form-field>
            <br />
            <mat-form-field class="simple-form-field-100">
              <input matInput type="password" placeholder="Password" name="password" formControlName="password" />
              <mat-error>Required</mat-error>
            </mat-form-field>
            <section>
              <button [disabled]="waiting" class="simple-form-button" color="primary" mat-raised-button type="submit"
                value="submit">
                Register
              </button>
            </section>
          </div>
        </mat-card-content>
      </form>
    </div>
  </div>
</mat-card>

```

register.component.css

```
.formDiv {
  width:28%;
  padding-left: 56px;
}

.imageStyle {
  margin-bottom: -16px;
}

.imageDiv {
  width: 62%;
}

.topDiv {
  display:flex;
}
```

register.component.ts

```typescript
import { Component, OnInit } from '@angular/core';
import { FormGroup, FormControl, Validators } from '@angular/forms';
import { HttpClient } from '@angular/common/http';
import { Router } from '@angular/router';
import { ChartService } from '../chart.service';
import { MatSnackBar } from '@angular/material';

@Component({
  selector: 'app-register',
  templateUrl: './register.component.html',
  styleUrls: ['./register.component.css']
})
export class RegisterComponent implements OnInit {
  form: FormGroup;
  constructor(private http: HttpClient, private chartService: ChartService, private router: Router, private snackBar: MatSnackBar) { }

  ngOnInit() {
    this.chartService.getOverallPieData();
    this.chartService.getOverallBarData();
    this.form = new FormGroup({
      firstName: new FormControl('', [Validators.required]),
      lastName: new FormControl('', [Validators.required]),
      email: new FormControl('', [Validators.required, Validators.email]),
      password: new FormControl('')
    });
  }

  submit(formData) {
    if (this.form.invalid) {
      return;
    }

    this.http.post('/api/user/register', {
      firstName: formData.firstName,
      lastName: formData.lastName,
      email: formData.email,
      password: formData.password,
    }).subscribe(success => {
      this.snackBar.open('Registered Successfully, please Login to continue', 'Dismiss', {
        duration: 5000,
        verticalPosition: 'top',
        horizontalPosition: 'center',
      });
        this.router.navigateByUrl('/');
    });
  }

}
```

login.component.html

```
<mat-card class="simple-form-card">
  <form [formGroup]="form" (ngSubmit)="submit(form.value)">
    <mat-card-title> Login </mat-card-title>
    <mat-card-subtitle>
      Don't have an account?
      <a routerLink="/register">
        Register here.
      </a>
    </mat-card-subtitle>
    <mat-card-content>
      <div class="form-group">
        <mat-form-field class="simple-form-field-100">
          <input matInput placeholder="Username" name="username" formControlName="username" />
        </mat-form-field>
        <br />
        <mat-form-field class="simple-form-field-100">
          <input matInput type="password" placeholder="Password" name="password" formControlName="password" />
        </mat-form-field>
        <br />
        <button [disabled]="waiting" class="simple-form-button" color="primary" mat-raised-button type="submit"
          value="submit">
          Log in
        </button>
        <span *ngIf="error" class="error">
          {{ errorMessage }}
        </span>
      </div>
    </mat-card-content>
  </form>
</mat-card>
```

login.component.css

```
.error {
    color: red;
}
```

login.component.ts

```typescript
import { Component, OnInit, ViewChild, ElementRef } from '@angular/core';
import { FormGroup, FormControl, Validators} from '@angular/forms';
import { HttpClient } from '@angular/common/http';
import { Router } from '@angular/router';
import { ChartService } from '../chart.service';
import { MatSnackBar } from '@angular/material';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {
  form: FormGroup;
  error = false;
  errorMessage: string;

  constructor(private http: HttpClient, private router: Router,
      private chartService: ChartService, private snackBar: MatSnackBar) { }

  ngOnInit() {
    this.form = new FormGroup({
      username: new FormControl('', [Validators.required]),
      password: new FormControl('', [Validators.required])
    });

  }

  submit(formData) {
    if (this.form.invalid) {
      return;
    }

    this.http.post('/api/user/login', {
      username: formData.username,
      password: formData.password,
    }).subscribe((data) => {
      this.error = false;
      this.snackBar.open('Logged in Successfully', 'Dismiss', {
        duration: 5000,
        verticalPosition: 'top',
        horizontalPosition: 'center',
      });
      this.chartService.setUser(data);
      this.router.navigateByUrl('overall');
    }, () => {
      this.error = true;
      this.errorMessage = 'Username or password is incorrect';
    });
  }
}
```

overall-analytics.component.html

```
<div class="container">
  <div class="cardStyle">
    <mat-card class="matCard">
      <div class="titleDiv">
        Highest number of revisions
      </div>
      <hr>
      <div>Number of article selected: <span class="userSelected">{{userSelection}}</span>
      </div>
      <div *ngFor="let list of lists">
        <div *ngIf="!changed" class="listStyle">{{list}}</div>
      </div>
      <div *ngFor="let lister of slicedList">
        <div *ngIf="changed" class="listStyle">{{lister}}</div>
      </div>
    </mat-card>
    <div>
      <mat-card class="matCardSlider">
        <div class="titleDiv">
          Choose number of Articles
        </div>
        <hr>
        <mat-slider class="sliderStyle" thumbLabel [displayWith]="formatLabel" tickInterval="1" min="1"
          [max]="listLength" (input)="chosenValue($event)">
        </mat-slider>
      </mat-card>
    </div>
    <mat-card class="matCard">
      <div class="titleDiv">
        Lowest number of revisions
      </div>
      <hr>
      <div>Number of article selected: <span class="userSelected">{{userSelection}}</span>
      </div>
      <div *ngFor="let lowList of lowRevList">
        <div *ngIf="!changed" class="listStyle">{{lowList}}</div>
      </div>
      <div *ngFor="let lowLister of slicedLowRevList">
        <div *ngIf="changed" class="listStyle">{{lowLister}}</div>
      </div>
    </mat-card>
  </div>
  <div class="largeStyle">
    <mat-card class="matWidth">
      <mat-card-header>
        <mat-card-title class="largeTitle">{{largeArticleName}}
          <hr>
        </mat-card-title>
        <mat-card-subtitle>Number of Unique Registered users: <span class="userSelected">{{largeArticleUsers}}</span>
        </mat-card-subtitle>
        <mat-card-subtitle>Number of Revisions: <span class="userSelected">{{largeArticleRevisions}}</span>
        </mat-card-subtitle>
      </mat-card-header>
    </mat-card>
    <mat-card class="matWidth">
```

```
56          <mat-card-header>
57            <mat-card-title class="largeTitle">{{smallArticleName}}
58              <hr>
59            </mat-card-title>
60            <mat-card-subtitle>Number of Unique Registered users: <span class="userSelected">{{smallArticleUsers}}</span>
61            </mat-card-subtitle>
62            <mat-card-subtitle>Number of Revisions: <span class="userSelected">{{smallArticleRevisions}}</span>
63            </mat-card-subtitle>
64          </mat-card-header>
65        </mat-card>
66      </div>
67      <div class="largeStyle">
68        <mat-card class="matWidth">
69          <div class="titleDiv">
70            Longest History
71          </div>
72          <hr>
73          <div class="longArticle">1. {{longArticleName[0]}}:
74            <span class="longDuration">{{longArticleDuration[0]}} days old</span>
75          </div>
76          <div class="longArticle">2. {{longArticleName[1]}}:
77            <span class="longDuration">{{longArticleDuration[1]}} days old</span>
78          </div>
79        </mat-card>
80        <mat-card class="matWidth">
81          <div class="titleDiv">
82            Shortest History
83          </div>
84          <hr>
85          <div class="longArticle">1. {{shortArticleName[0]}}:
86            <span class="longDuration">{{shortArticleDuration[0]}} days old</span>
87          </div>
88          <div class="longArticle">2. {{shortArticleName[1]}}:
89            <span class="longDuration">{{shortArticleDuration[1]}} days old</span>
90          </div>
91        </mat-card>
92      </div>
93      <mat-card class="chartStyle">
94        <div class="chartSelection">
95          <span class="barOrPie">Bar Chart </span>
96          <mat-slide-toggle (change)="barOrPie($event)"></mat-slide-toggle> <span class="barOrPie"> Pie Chart</span>
97        </div>
98        <mat-card-subtitle *ngIf="!chartChoice">Distribution by year and by user type across the whole dataset
99        </mat-card-subtitle>
100       <app-my-bar-chart *ngIf="!chartChoice" [barData]="barData"></app-my-bar-chart>
101       <mat-card-subtitle *ngIf="chartChoice">Distribution by user type across the whole data set </mat-card-subtitle>
102       <app-pie-chart *ngIf="chartChoice" [pieData]="pieData"></app-pie-chart>
103     </mat-card>
104   </div>
105
```

overall-analytics.component.css

```css
.selectionCard {
    display: flex;
    height: 90px;
    margin-top: 10px;
    justify-content: center;
}

.cardStyle {
    display: flex;
    justify-content: space-around;
}
.matWidth {
    width: 49%;
}

.matCard {
    margin-top: 10px;
    width: 346px;
    height: 100%;
}

.matCardSlider {
    margin-top: 10px;
    width: 374px;
}

.titleDiv {
    text-align: left;
    margin-bottom: -10px;
    font-weight: bolder;
    color: #6C809A;
    font-size: 20px;
}

.formDiv {
    display: inline-grid;
}

.userSelected {
    color:black;
    font-weight: bolder;
}

.listStyle {
    font-weight:bolder;
    color: cadetblue;
}

.sliderStyle {
    width: -webkit-fill-available;
}

.largeStyle {
    display:flex;
    justify-content:space-between;
```

```
56      margin-top: 10px;
57  }
58
59  .largeTitle {
60      font-size: 30px;
61      margin-bottom: -10px;
62      color: cadetblue;
63  }
64
65  .longArticle {
66      font-size: 30px;
67      color: cadetblue;
68  }
69
70  .longDuration {
71      font-size: 20px;
72      color: rgba(0,0,0,.54);
73      font-weight: 400;
74  }
75
76  .chartStyle {
77      margin-top: 10px;
78  }
79
80  .chartSelection {
81      float: right;
82  }
83
84  .barOrPie {
85      font-size:14px;
86      font-weight: bolder;
87      color: #6C809A;
88  }
```

overall-analytics.component.ts

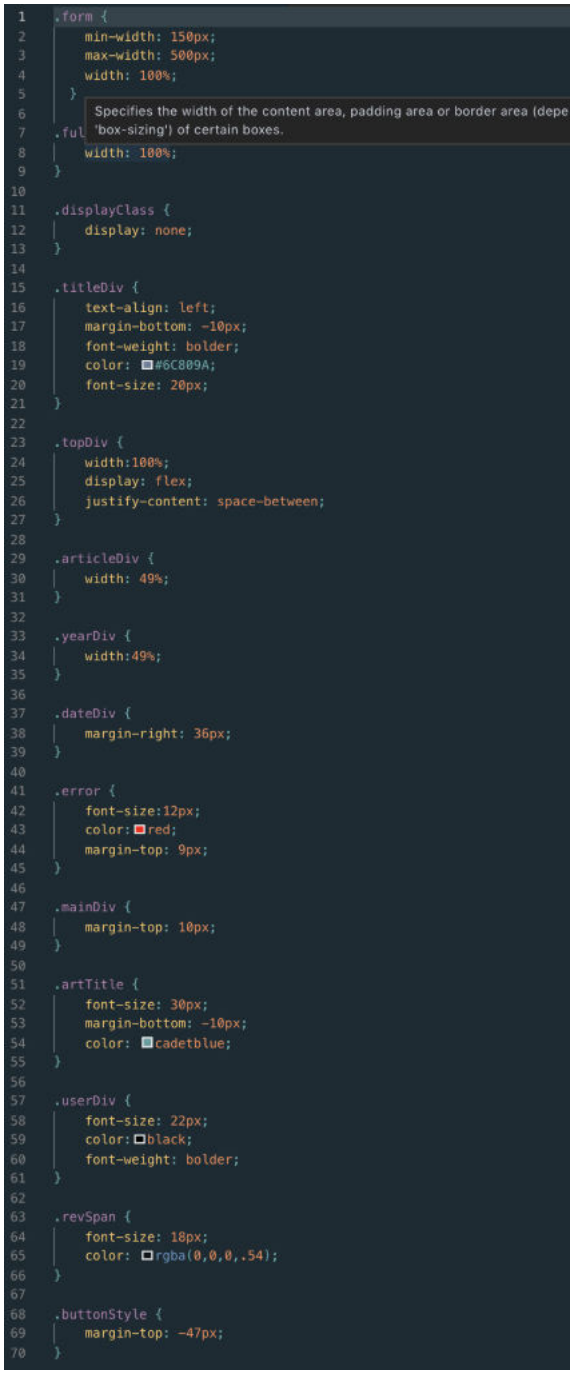
```
import { Component, OnInit } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { ChartService } from '../chart.service';

@Component({
  selector: 'app-overall-analytics',
  templateUrl: './overall-analytics.component.html',
  styleUrls: ['./overall-analytics.component.css']
})
export class OverallAnalyticsComponent implements OnInit {
  lists = [];
  lowRevList = [];
  slicedList = [];
  articleList = [];
  articleLowRev = [];
  slicedLowRevList = [];
  lowListLength: number;
  listLength: number;
  changed = false;
  userSelection = 2;
  error = false;
  largeArticleName: string;
  largeArticleUsers: string;
  largeArticleRevisions: number;
  smallArticleName: string;
  smallArticleUsers: string;
  smallArticleRevisions: number;
  longArticleName: Array<string> = [];
  longArticleDuration: Array<string> = [];
  shortArticleName: Array<string> = [];
  shortArticleDuration: Array<string> = [];
  barData: any;
  pieData: any;
  chartChoice = false;

  constructor(private http: HttpClient, private chartService: ChartService) { }

  ngOnInit() {
    this.http.get('/api/overall/top-most', {
      params: {
        sortNum: '-1',
      }
    }).subscribe(data => {
      Object.values(data).forEach(element => {
        this.articleList.push(element['_id']);
      });
      this.listLength = this.articleList.length;
      this.lists = this.articleList.slice(0, 2);
    });
    this.http.get('/api/overall/top-most', {
      params: {
        sortNum: '1',
      }
    }).subscribe(data2 => {
      Object.values(data2).forEach(element => {
```

```
            this.articleLowRev.push(element['_id']);
          });
          this.lowRevList = this.articleLowRev.slice(0, 2);
          this.lowListLength = this.articleLowRev.length;
        });
        this.http.get('/api/overall/large-registered-users')
          .subscribe(largeUsers => {
            this.largeArticleName = largeUsers[0]._id;
            this.largeArticleUsers = largeUsers[0].numberOfUsers;
            this.largeArticleRevisions = largeUsers[0].rev;
        });
        this.http.get('/api/overall/small-registered-users')
          .subscribe(smallUsers => {
            this.smallArticleName = smallUsers[0]._id;
            this.smallArticleUsers = smallUsers[0].numberOfUsers;
            this.smallArticleRevisions = smallUsers[0].rev;
        });
        this.http.get('/api/overall/longest-history')
          .subscribe(longHistory => {
            this.longArticleName.push(longHistory[0][0]);
            this.longArticleName.push(longHistory[1][0]);
            this.longArticleDuration.push(longHistory[0][1]);
            this.longArticleDuration.push(longHistory[1][1]);
        });
        this.http.get('/api/overall/smallest-history')
          .subscribe(shortHistory => {
            this.shortArticleName.push(shortHistory[0][0]);
            this.shortArticleDuration.push(shortHistory[0][1]);
            this.shortArticleName.push(shortHistory[1][0]);
            this.shortArticleDuration.push(shortHistory[1][1]);
        });
        this.barData = this.chartService.getOverallBarData();
        this.pieData = this.chartService.getOverallPieData();
    }

    wasFormChanged(value) {
      if (value <= this.listLength) {
        this.slicedList = this.articleList.slice(0, value);
        this.userSelection = value;
        this.slicedLowRevList = this.articleLowRev.slice(0, value);
        this.changed = true;
        this.error = false;
      } else {
        this.error = true;
      }
    }

    chosenValue(event) {
      this.wasFormChanged(event.value);
    }

    barOrPie(chartType) {
      this.chartChoice = chartType.checked;
    }
  }
```

individual-analytics.component.html

```
<div class="topDiv">
  <mat-card class="articleDiv">
    <div class="titleDiv">Choose Article</div>
    <hr>
    <form class="form">
      <mat-form-field class="full-width">
        <input matInput placeholder="Articles" [matAutocomplete]="auto" [formControl]="articleCtrl">
        <mat-autocomplete #auto="matAutocomplete" (optionSelected)="selectedArticle($event.option.value)">
          <mat-option *ngFor="let article of filteredArticle | async" [value]="article._id">
            <span>{{article._id}}</span> |
            <small>Revisions: {{article.rev}}</small>
          </mat-option>
        </mat-autocomplete>
      </mat-form-field>
    </form>
  </mat-card>
  <mat-card class="yearDiv">
    <div class="titleDiv">Year Filter</div>
    <hr>
    <mat-form-field class="dateDiv">
      <input matInput placeholder="From:" name="fromDate" [formControl]="date">
    </mat-form-field>
    <mat-form-field>
      <input matInput placeholder="To:" name="toDate" [formControl]="date2">
    </mat-form-field>
    <mat-card-actions align="end">
      <span class="error" *ngIf="error">{{errorMessage}}</span>
      <button [disabled]="filterOn" mat-raised-button color="primary" (click)="filterYears()">Filter</button>
    </mat-card-actions>
  </mat-card>
</div>
<mat-card class="mainDiv" *ngIf="articleReceived">
  <div class="artTitle">{{articleTitle}}
    <mat-card-subtitle>{{totalRevision}} total revisions made</mat-card-subtitle>
  </div>
  <hr>
  <div *ngFor="let users of topFiveUsers">
    <div class="userDiv">
      {{users.user}} : <span class="revSpan">{{users.revisions}} revisions</span>
    </div>
  </div>
</mat-card>
<mat-card class="mainDiv">
  <mat-spinner class="spinner" *ngIf="waiting"></mat-spinner>
  <div *ngIf="articleReceived">
    <mat-card-subtitle *ngIf="overallBar">Distribution by year and by user type across {{articleTitle}}
    </mat-card-subtitle>
    <mat-card-subtitle *ngIf="pieChart">Distribution by year and by user type across {{articleTitle}}
    </mat-card-subtitle>
    <mat-card-subtitle *ngIf="individualBar">Distribution by year and by single user type across {{articleTitle}}
    </mat-card-subtitle>
    <mat-card-actions class="buttonStyle" align="end">
      <button mat-raised-button color="primary" (click)="chartType('overallBar')">Overall Bar Chart</button>
      <button mat-raised-button color="primary" (click)="chartType('pieChart')">Pie chart</button>
      <button mat-raised-button color="primary" (click)="chartType('individualBar')">Individual Bar Chart</button>
    </mat-card-actions>
    <app-my-bar-chart *ngIf="overallBar" [barData]="barData"></app-my-bar-chart>
    <app-pie-chart *ngIf="pieChart" [pieData]="pieData"></app-pie-chart>
    <app-my-bar-chart *ngIf="individualBar" [barData]="individualData" [type]="individual"></app-my-bar-chart>
  </div>
</mat-card>
```

individual-analytics.component.css



```
.form {
    min-width: 150px;
    max-width: 500px;
    width: 100%;
  }

.ful
    width: 100%;
}

.displayClass {
    display: none;
}

.titleDiv {
    text-align: left;
    margin-bottom: -10px;
    font-weight: bolder;
    color: #6C809A;
    font-size: 20px;
}

.topDiv {
    width:100%;
    display: flex;
    justify-content: space-between;
}

.articleDiv {
    width: 49%;
}

.yearDiv {
    width:49%;
}

.dateDiv {
    margin-right: 36px;
}

.error {
    font-size:12px;
    color:red;
    margin-top: 9px;
}

.mainDiv {
    margin-top: 10px;
}

.artTitle {
    font-size: 30px;
    margin-bottom: -10px;
    color: cadetblue;
}

.userDiv {
    font-size: 22px;
    color:black;
    font-weight: bolder;
}

.revSpan {
    font-size: 18px;
    color: rgba(0,0,0,.54);
}

.buttonStyle {
    margin-top: -47px;
}
```

Specifies the width of the content area, padding area or border area (depe
'box-sizing') of certain boxes.

individual-analytics.component.ts

```typescript
import { Component, OnInit, ViewChild } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { FormGroup, FormControl, Form} from '@angular/forms';
import {Observable} from 'rxjs';
import {map, startWith} from 'rxjs/operators';
import { MatSnackBar } from '@angular/material';

@Component({
  selector: 'app-individual-analytics',
  templateUrl: './individual-analytics.component.html',
  styleUrls: ['./individual-analytics.component.css'],
})
export class IndividualAnalyticsComponent implements OnInit {
  date = new FormControl();
  date2 = new FormControl();
  form: FormGroup;
  articleList: Array<string> = [];
  articleCtrl = new FormControl();
  filteredArticle: Observable<any>;
  articleNames: Array<string> = [];
  topFiveUsers: Array<any> = [];
  articleReceived = false;
  articleTitle: string;
  totalRevision: number;
  barData: any;
  pieData: any;
  individualData: any;
  individual: string;
  error = false;
  errorMessage: string;
  filterOn = true;
  yearList: Array<any> = [];
  newList: Array<any> = [];
  overallBar = true;
  pieChart = false;
  individualBar = false;
  waiting = false;
  open = false;

  constructor(private http: HttpClient, private snackBar: MatSnackBar) {
    this.filteredArticle = this.articleCtrl.valueChanges
      .pipe(
        startWith(''),
        map(article => article ? this._filterArticles(article) : this.articleList.slice())
      );
  }

  private _filterArticles(value: string) {
    const filterValue = value.toLowerCase();
    return this.articleList.filter(state => state['_id'].toLowerCase().indexOf(filterValue) === 0);
  }

  ngOnInit() {
    this.http.get('/api/individual/all-articles')
      .subscribe((response) => {
```

```
      Object.values(response).forEach((element) => {
        this.articleList.push(element);
      });
    });
  }

  selectedArticle(value) {
    this.articleReceived = false;
    this.topFiveUsers.length = 0;
    this.open = true;
    this.waiting = true;
    this.http.post('/api/individual/article-details', {
      title: value
    }).subscribe((result) => {
      this.waiting = false;
      this.filterOn = false;
      this.articleTitle = value;
      this.totalRevision = result['rev'][2];
      this.snackBar.open(`Data pulled successfully and ${result['rev'][1]} revisions are downloaded`, 'Dismiss', {
        duration: 5000,
        verticalPosition: 'top',
        horizontalPosition: 'center',
      });
      this.barData = result['rev'][3];
      this.pieData = result['rev'][4];
      this.individualData = result['rev'][5];
      this.individual = result['rev'][0][0]['user'];
      this.yearList = result['rev'][6].reverse();
      Object.values(result['rev'][0]).forEach((element) => {
        this.topFiveUsers.push(element);
      });
        this.articleReceived = true;
    });
  }

  filterYears() {
    if (this.date.value > this.date2.value) {
      this.error = true;
      this.errorMessage = 'Please choose From: year less than To: year';
    } else if (this.yearList.indexOf(parseInt(this.date.value, 10)) === -1 ||
        this.yearList.indexOf(parseInt(this.date2.value, 10)) === -1) {
      this.error = true;
      this.errorMessage = 'Please choose a year from ' + this.yearList[0] + ' to ' + this.yearList[this.yearList.length - 1];
    } else {
      this.articleReceived = false;
      this.topFiveUsers.length = 0;
      this.error = false;
      const first = this.yearList.indexOf(parseInt(this.date.value, 10));
      const second = this.yearList.indexOf(parseInt(this.date2.value, 10));
      this.newList = this.yearList.slice(first, second + 1);
      this.waiting = true;
      this.http.post('/api/individual/article-details', {
          title: this.articleCtrl.value,
          fromDate: this.date.value,
          toDate: this.date2.value,
```

```
111             yearList: this.newList,
112           }).subscribe((result) => {
113             this.waiting = false;
114             this.filterOn = false;
115             this.articleTitle = this.articleCtrl.value;
116             this.totalRevision = result['rev'][7];
117             this.snackBar.open(`Data pulled successfully and ${result['rev'][1]} revisions are downloaded`, 'Dismiss', {
118               duration: 5000,
119               verticalPosition: 'top',
120               horizontalPosition: 'center',
121             });
122             this.barData = result['rev'][3];
123             this.pieData = result['rev'][4];
124             this.individualData = result['rev'][5];
125             this.individual = result['rev'][0][0]['user'];
126             this.yearList = result['rev'][6].reverse();
127             Object.values(result['rev'][0]).forEach((element) => {
128               this.topFiveUsers.push(element);
129             });
130             this.articleReceived = true;
131             this.overallBar = true;
132             this.pieChart = false;
133             this.individualBar = false;
134           });
135         }
136       }
137
138       chartType(type) {
139         if (type === 'pieChart') {
140           this.pieChart = true;
141           this.overallBar = false;
142           this.individualBar = false;
143         } else if (type === 'individualBar') {
144           this.individualBar = true;
145           this.pieChart = false;
146           this.overallBar = false;
147         } else if (type === 'overallBar') {
148           this.overallBar = true;
149           this.pieChart = false;
150           this.individualBar = false;
151         }
152       }
153
154     }
155
```

author-analytics.component.html

```
<div class="topDiv">
  <mat-card>
    <div class="titleDiv">Choose User</div>
    <hr>
    <form>
      <mat-form-field class="firstInput">
        <input matInput placeholder="Enter author name" name="author" [formControl]="authorCtrl">
      </mat-form-field>
    </form>
    <mat-card-actions>
      <button mat-raised-button color="primary" (click)="searchFunction()">Search</button>
      <span *ngIf="error" class="error">{{errorMessage}}</span>
    </mat-card-actions>
  </mat-card>
</div>
<mat-accordion class="headers-align">
  <mat-expansion-panel *ngFor="let article of articleList" hideToggle>
    <mat-expansion-panel-header>
      <mat-panel-title>
        {{article._id}}
      </mat-panel-title>
      <mat-panel-description>{{article.rev}} revisions
      </mat-panel-description>
    </mat-expansion-panel-header>
    <div *ngFor="let time of article.timelist">
      {{time}}
    </div>
  </mat-expansion-panel>
</mat-accordion>
```

author-analytics.component.css

```
.headers-align .mat-expansion-panel-header-title,
.headers-align .mat-expansion-panel-header-description {
  flex-basis: 0;
}

.headers-align .mat-expansion-panel-header-description {
  justify-content: space-between;
  align-items: center;
}

.titleDiv {
    text-align: left;
    margin-bottom: -10px;
    font-weight: bolder;
    color: #6C809A;
    font-size: 20px;
}

.topDiv {
  margin-bottom: 10px;
}

.firstInput {
  width: 50%;
}

.error {
  color: red;
  font-size: 12px;
  margin-left: 10px;
}
```

author-analytics.component.ts

```
import { Component, OnInit } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { FormControl } from '@angular/forms';

@Component({
  selector: 'app-author-analytics',
  templateUrl: './author-analytics.component.html',
  styleUrls: ['./author-analytics.component.css']
})
export class AuthorAnalyticsComponent implements OnInit {
  authorCtrl = new FormControl();
  articleList: Array<any> = [];
  expand = false;
  errorMessage: string;
  error = false;
  constructor(private http: HttpClient) {
  }

  ngOnInit() {

  }

  searchFunction() {
    this.articleList.length = 0;
    this.http.get('/api/author/author-details', {
      params: {
        author: this.authorCtrl.value,
      }
    }).subscribe(result => {
      this.error = false;
      Object.values(result).forEach(element => {
        this.articleList.push(element);
      });
    }, (errResponse) => {
        this.error = true;
        this.errorMessage = 'User not found';
    });
  }
}

```

my-bar-chart.component.html

```
<div>
  <div class="barStyle">
    <canvas baseChart
            [datasets]="barChartData"
            [labels]="barChartLabels"
            [options]="barChartOptions"
            [legend]="barChartLegend"
            [chartType]="barChartType">
    </canvas>
  </div>
</div>
```

my-bar-chart.component.ts

```typescript
import { Component, OnInit, Input } from '@angular/core';

@Component({
  selector: 'app-my-bar-chart',
  templateUrl: './my-bar-chart.component.html',
  styleUrls: ['./my-bar-chart.component.css']
})
export class MyBarChartComponent implements OnInit {
  @Input() barData;
  @Input() type;
  adminData: Array<string> = [];
  anonData: Array<string> = [];
  botData: Array<string> = [];
  regularData: Array<string> = [];
  testList = {};
  individualData: Array<string> = [];
  testLabel: string;
  test3: Array<string> = [];
  constructor() { }
  public barChartOptions = {
    scaleShowVerticalLines: false,
    responsive: true
  };
  barChartLabels: Array<string> = [];
  public barChartType = 'bar';
  public barChartLegend = true;
  public barChartData: any;
  ngOnInit() {
    if (!this.type) {
      setTimeout(() => {
        this.testList = {0: 'Admin', 1: 'Bot', 2: 'Anonymous', 3: 'Regular'};
        for (let i = 0; i < 4; i++) {
          Object.values(this.barData[this.testList[i]]).forEach(element => {
            this.test3.push(element['_id']);
          });
          this.datasetCreation(this.barData, this.testList[i], this.test3);
        }
      }, 0);
      this.barChartData = [
        {data: this.adminData, label: 'Administrator'},
        {data: this.anonData, label: 'Anonymous'},
        {data: this.botData, label: 'Bot'},
        {data: this.regularData, label: 'Regular'}];
    } else {
      setTimeout(() => {
        this.barChartLabels = Object.keys(this.barData);
        let count = 0;
        Object.values(this.barData).forEach(element => {
          count += 1;
          this.individualData.push(element['rev']);
          if (count <= 1) {
            this.testLabel = element['_id'];
          }
        });
      });
```

```
            this.barChartData = [{
              data: this.individualData, label: this.type
            }];
          }
        }

        datasetCreation(data, userType, yearList) {
          Object.values(data[userType]).forEach(ele => {
            if (userType === 'Admin') {
              this.barChartLabels.push(ele['_id']);
              this.adminData.push(ele['rev']);
            } else if (userType === 'Anonymous') {
              this.anonData.push(ele['rev']);
            } else if (userType === 'Bot') {
              this.botData.push(ele['rev']);
            } else if (userType === 'Regular') {
              this.regularData.push(ele['rev']);
            }
          });
          return;
        }
      }

```

pie-chart.component.html

```
<div>
  <div class="chart">
    <canvas baseChart
      [data]="pieChartData"
      [labels]="pieChartLabels"
      [chartType]="pieChartType"
      [options]="pieChartOptions"
      [colors]="pieChartColors"
      [legend]="pieChartLegend">
    </canvas>
  </div>
</div>
```

pie-chart.component.ts

```ts
import { Component, OnInit, Input } from '@angular/core';

@Component({
  selector: 'app-pie-chart',
  templateUrl: './pie-chart.component.html',
  styleUrls: ['./pie-chart.component.css']
})
export class PieChartComponent implements OnInit {

  @Input() pieData;
  public pieChartOptions = {
    responsive: true,
    legend: {
      position: 'top',
    },
    plugins: {
      datalabels: {
        formatter: (value, ctx) => {
          const label = ctx.chart.data.labels[ctx.dataIndex];
          return label;
        },
      },
    }
  };
  public pieChartLabels = [];
  public pieChartData = [];
  public pieChartType = 'pie';
  public pieChartLegend = true;
  public pieChartColors = [
    {
      backgroundColor: ['#80add7', '#ffd740', '#dbb4da', '#d6618f'],
    },
  ];
  constructor() { }

  ngOnInit() {
    setTimeout(() => {
      Object.values(this.pieData).forEach((element) => {
        this.pieChartLabels.push(element[0]._id);
        this.pieChartData.push(element[0].rev);
      });
    });
  }

}
```

user-dropdown.component.html

```
<button *ngIf="!isLoggedIn" mat-button routerLink="/login" class="buttonStyle">
  <mat-icon>person</mat-icon>Login
</button>
<button *ngIf="isLoggedIn" mat-button [matMenuTriggerFor]="userDropdownMenu" class="buttonStyle">
  <label class="iconStyle">
    <mat-icon>person</mat-icon>
  </label>
  {{ user?.firstName }} {{ (user?.lastName)[0] }}
</button>

<mat-menu #userDropdownMenu="matMenu" overlapTrigger="false">
  <button mat-menu-item routerLink="/profile">
    <mat-icon>person</mat-icon>
    <span>Profile</span>
  </button>
  <button mat-menu-item (click)="logout()">
    <mat-icon>exit_to_app</mat-icon>
    <span>Log out</span>
  </button>
</mat-menu>
```

user-dropdown.component.css

```
.buttonStyle {
    min-width:50px
}

.iconStyle {
    margin-top: 16px
}
```

user-dropdown.component.ts

```
import { Component, OnInit, OnDestroy } from '@angular/core';
import { User, ChartService } from '../chart.service';

@Component({
  selector: 'app-user-dropdown',
  templateUrl: './user-dropdown.component.html',
  styleUrls: ['./user-dropdown.component.css']
})
export class UserDropdownComponent implements OnInit {
  user: User;
  isLoggedIn: boolean;

  constructor(private chartService: ChartService) {}

  ngOnInit() {
    this.chartService.userObservable.subscribe(user => {
      this.user = user;
      this.isLoggedIn = !!this.user;
    });
  }

  logout() {
    this.chartService.logout();
  }

}
```

Controller:

server.js

```
var express = require('express');
var path = require('path');
var http = require('http');
var bodyParser = require("body-parser");
var session = require('express-session');


var router = require('./router/router');

var app = express();

app.use(bodyParser.json({}));
app.use(bodyParser.urlencoded({extended: true}));

app.use(session({
  secret: 'secret',
  resave: false,
  saveUninitialized: true,
  cookie: { secure: true }
}));

router(app);

const appPath = path.join(__dirname, "..", "dist");
app.use(express.static(appPath));

app.get('*', function(req, res) {
  res.sendFile(path.resolve(appPath, 'index.html'));
});

app.use(function(error, req, res, next) {
  const statusCode =
    !res.statusCode || res.statusCode === 200 ? 500 : res.statusCode;
  res.status(statusCode).json({ message: error.message });
});

var port = 3000
var server = http.createServer(app);
server.listen(port, function() {
  console.log(`Application listening to port ${port}`);
});

module.exports = app;

```

revision.controller.js

```
var Revision = require('../model/revisions');
var User = require('../model/user');
var fs  = require("fs");
var concat = require('concat');
var sess;

module.exports = function(app) {
  app.post('/api/user/register', register);
  app.post('/api/user/login', login);
  app.get('/api/overall/top-most', getTopMost);
  app.get('/api/overall/large-registered-users',larGroupOfRegisteredUser);
  app.get('/api/overall/small-registered-users', smallGroupOfRegisteredUser);
  app.get('/api/overall/longest-history', ArticlesLongestHistory);
  app.get('/api/overall/smallest-history', ArticlesSmallestHistory);
  app.get('/api/overall/bar-chart', distributionByYearAndUserType);
  app.get('/api/overall/pie-chart', distributionByUserType);
  app.post('/api/user/logout', logout);
}

function register(req, res) {
  const firstName = req.body.firstName;
  const lastName = req.body.lastName;
  const email = req.body.email;
  const password = req.body.password;

  const user = new User();
  user.firstName = firstName;
  user.lastName = lastName;
  user.email = email;
  user.createPasswordHash(password);
  return user.save()
    .then(() => {
      return res.status(200).send({message: 'Success'});
    })
    .catch(() => {
      return res.status(500).send({message: 'Error in registering'});
    });
}

function login(req, res) {
  const email = req.body.username;
  const password = req.body.password;
  return User.findOne({email})
    .then((user) => {
      if(!(user.checkPassword(password))) {
        return res.send({message: 'Incorrect Password'});
      } else {
        return res.status(200).send(user);
      }
    })
    .catch(() => {
      return res.status(500).send({message: 'User not found'});
    })
}

```

```
function getTopMost(req, res) {
  sort = req.query.sortNum;
  if(req.session) {
    Revision.findTopArticles(parseInt(sort), function(err, result) {
      if(err) {
        console.log('Err');
      } else {
        return res.status(200).send(result);
      }
    });
  } else {
    return res.status(500).send({message: 'Unauthorized Access'});
  }
}

function larGroupOfRegisteredUser(req,res){
  if(req.session) {
    Revision.aggregate([
      {$match:{anon:{$exists:false}}},
            {$group:{_id:"$title",rev:{$sum:1},uniqueValues:{$addToSet:"$user"}}},
            {$project: {title: 1, rev: 1, numberOfUsers: {$cond: {if: {$isArray: "$uniqueValues"}, then: {$size: "$uniqueValues"}, else: "NA"}}}},
            {$sort: {numberOfUsers:-1}},
            {$limit:1}
    ], function(err, result) {
      if(err) {
        console.log(err);
      } else {
        return res.status(200).send(result);
      }
    });
  } else {
    return res.status(500).send({message: 'Unauthorized Access'});
  }
}

function smallGroupOfRegisteredUser(req,res) {
  if (req.session) {
    Revision.aggregate([
      {$match:{anon:{$exists:false}}},
            {$group:{_id:"$title",rev:{$sum:1},uniqueValues:{$addToSet:"$user"}}},
            {$project: {title: 1, rev: 1, numberOfUsers: {$cond: {if: {$isArray: "$uniqueValues"}, then: {$size: "$uniqueValues"}, else: "NA"}}}},
            {$sort: {numberOfUsers: 1}},
            {$limit:1}
    ], function(err, result) {
      if(err) {
        console.log(err);
      } else {
        return res.status(200).send(result);
      }
    });
  } else {
    return res.status(500).send({message: 'Unauthorized Access'});
  }
}

```

```
function dateDifference(oldDate) {
  var todaysDate = Date.now();
  var diffDays = parseInt((todaysDate - oldDate) / (1000 * 60 * 60 * 24)); //gives day difference
  return diffDays;
}

function ArticlesLongestHistory(req,res) {
  if (req.session) {
    Revision.aggregate([
      {$group:{_id:"$title",timestamp:{"$min":"$timestamp"}}},
      {$sort:{timestamp:1}},
      {$limit:2}],function(err, result) {
        if(err) {
          console.log(err);
        } else {
          var dateOne = new Date(result[0].timestamp);
          var dateTwo = new Date(result[1].timestamp);
          var newData = [];
          newData.push([result[0]._id, dateDifference(dateOne)]);
          newData.push([result[1]._id, dateDifference(dateTwo)]);
          return res.status(200).send(newData);
        }
      });
  } else {
    return res.status(500).send({message: 'Unauthorized Access'});
  }
}
function ArticlesSmallestHistory(req,res) {
  if (req.session) {
    Revision.aggregate([
      {$group:{_id:"$title",timestamp:{"$min":"$timestamp"}}},
      {$sort:{timestamp: -1}},
      {$limit:2}],function(err, result) {
        if(err) {
          console.log(err);
        } else {
          var dateOne = new Date(result[0].timestamp);
          var dateTwo = new Date(result[1].timestamp);
          var newData = [];
          newData.push([result[0]._id, dateDifference(dateOne)]);
          newData.push([result[1]._id, dateDifference(dateTwo)]);
          return res.status(200).send(newData);
        }
      });
  } else {
    return res.status(500).send({message: 'Unauthorized Access'});
  }
}



function distributionByYearAndUserType(req, res) {
  var merged = [];
  var admin_merge = [];
  concat(['admins/admin_active.txt', 'admins/admin_former.txt', 'admins/admin_inactive.txt', 'admins/admin_semi_active.txt'])
```

```
.then((result) => {
    admin_merge = result.split('\n');
    Revision.aggregate([
      {$match: {user:{"$in": admin_merge}}},
      {$project:{year:{$year:{$dateFromString: {
        dateString: "$timestamp"}}}},
      {$group:{ _id: "$year", rev:{$sum:1}}},
      {$sort: {_id: 1}}
      ], function(err, result) {
        if (err) {
          console.log(err);
        } else {
          var admins = result;
          fs.readFile('admins/bot.txt', function(err, f){
            var bot = f.toString().split('\n');
            var concat1 = admin_merge.concat(bot);
            Revision.aggregate([
              {$match: {user:{"$in": bot}}},
              {$project:{year:{$year:{$dateFromString: {
                dateString: "$timestamp"}}}}},
              {$group:{ _id: "$year", rev:{$sum:1}}},
              {$sort: {_id: 1}}
              ], function(err, result2) {
                if (err) {
                  console.log(err);
                } else {
                  var bot_user = result2;
                  merged = [].concat.apply([], concat1);
                  Revision.aggregate([
                    {$match: {user:{"$nin": merged}, anon:{$exists:false}}},
                    {$project:{year:{$year:{$dateFromString: {
                      dateString: "$timestamp"}}}}},
                    {$group:{ _id: "$year", rev:{$sum:1}}},
                    {$sort: {_id: 1}}
                    ], function(err, result3) {
                      if (err) {
                        console.log(err);
                      } else {
                        var regular = result3;
                        Revision.aggregate([
                          {$match: {anon:{$exists:true}}},
                          {$project:{year:{$year:{$dateFromString: {
                            dateString: "$timestamp"}}}}},
                          {$group:{ _id: "$year", rev:{$sum:1}}},
                          {$sort: {_id: 1}}
                          ], function(err, result4) {
                            if (err) {
                              console.log(err);
                            } else {
                              var anonymous = result4;
                              res.send({Admin: admins, Bot: bot_user, Regular: regular, Anonymous: anonymous});
                            }
                          })
                      }
                    })
                }
              })
          })
        }
      })
```

```
                    }
                  });
                });
              }
            });

        });
}

function distributionByUserType(req, res) {
  var merged = [];
  var admin_merge = [];
  concat(['admins/admin_active.txt', 'admins/admin_former.txt', 'admins/admin_inactive.txt', 'admins/admin_semi_active.txt'])
    .then((result) => {
      admin_merge = result.split('\n');
      Revision.aggregate([
        {$match: {user:{"$in": admin_merge}}},
        {$group:{ _id: "Administrator", rev:{$sum:1}}},
        {$sort: {_id: 1}}
        ], function(err, result) {
          if (err){
            console.log(err);
          } else {
            var adminRev = result;
            fs.readFile('admins/bot.txt', function(err, f){
              var bot = f.toString().split('\n');
              var concat1 = admin_merge.concat(bot);
              Revision.aggregate([
                {$match: {user:{"$in": bot}}},
                {$group:{ _id: "Bot", rev:{$sum:1}}},
                {$sort: {_id: 1}}
                ], function(err, result2) {
                  if (err){
                    console.log(err);
                  } else {
                    var botRev = result2;
                    merged = [].concat.apply([], concat1);
                    Revision.aggregate([
                      {$match: {user:{"$nin": merged}, anon:{$exists:false}}},
                      {$group:{ _id: "Regular User", rev:{$sum:1}}},
                      ], function(err, result3) {
                        if (err){
                          console.log(err);
                        } else {
                          var regularRev = result3;
                          Revision.aggregate([
                            {$match: {anon: true}},
                            {$group: {_id: "Anonymous", rev: {$sum: 1}}}
                          ], function(err, result4) {
                            if (err) {
                              console.log(err);
                            } else {
                              anonRev = result4;
                              return res.send({adminRev, botRev, regularRev, anonRev});
                            }
```

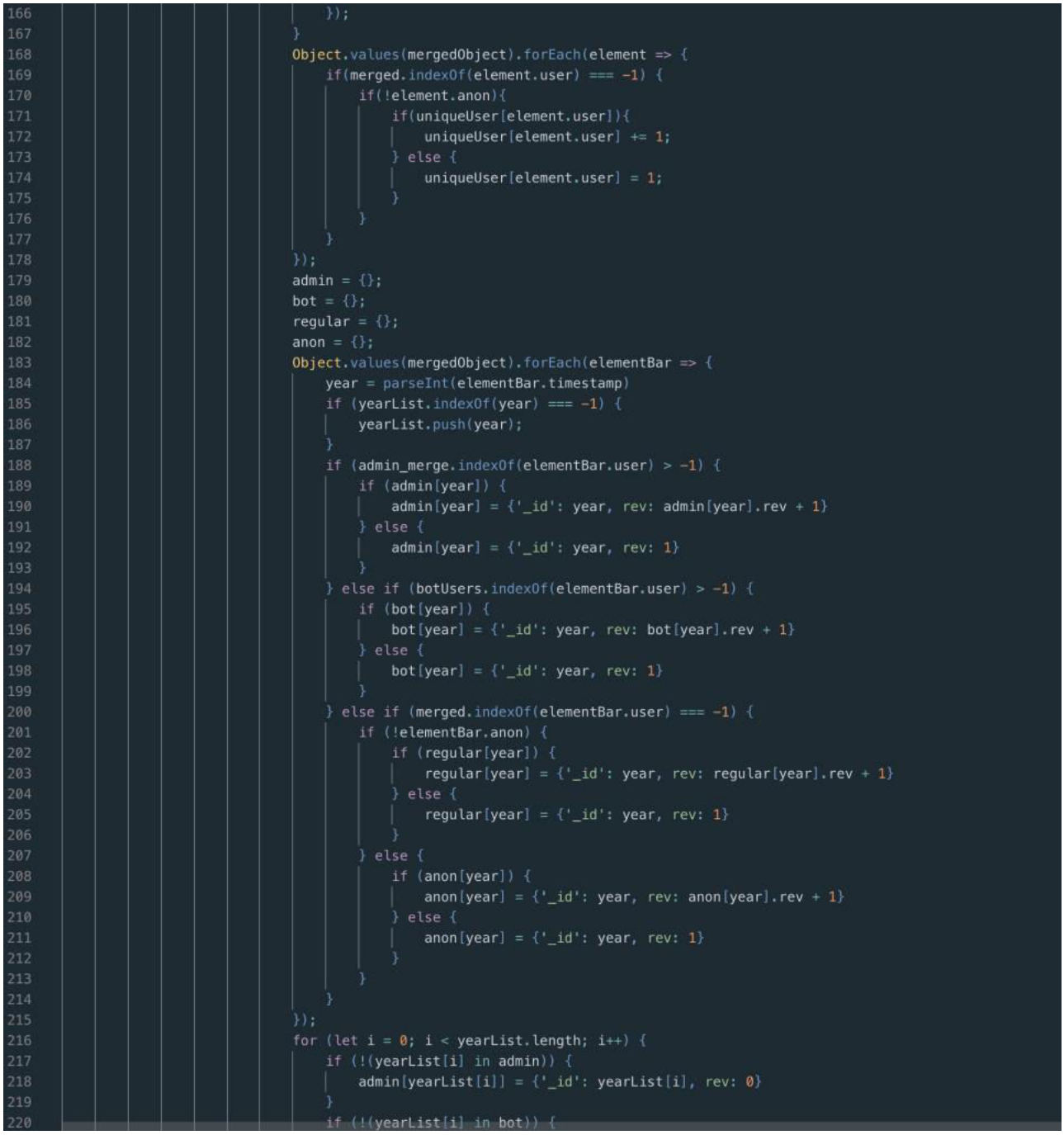
```
276                                    })
277                                  }
278                                })
279                              }
280                            })
281                          });
282                        }
283                      })
284                    });
285  }
286
287  function logout(req, res) {
288    req.session.destroy((err) => {
289      if(err) {
290        return console.log(err);
291      } else {
292        return res.status(200).send({message: 'Successfully logged out'});
293      }
294
295  });
296  }
```

individual.controller.js

```
var Revision = require('../model/revisions');
var request = require('request');
var fs  = require("fs");
var request = require('request');
var concat = require('concat');


module.exports = function(app) {
    app.get('/api/individual/all-articles', allArticles);
    app.post('/api/individual/article-details', restFunction);
}

function allArticles(req, res) {
    if (req.session) {
        Revision.aggregate([
            {$group: {_id: "$title", rev: {$sum:1}}},
            {$sort: {_id: 1}}], function(err, result) {
                if (err) {
                    console.log(err);
                } else {
                    return res.status(200).send(result);
                }
        });
    } else {
        return res.status(500).send({message: 'Unauthorized Access'});
    }
}

function titleDate(title) {
  var sync = true;
  var listData = [];

  Revision.aggregate([
    {$match:{title:title}},
    {$sort:{timestamp:-1}}],function(err, result) {
      if(err) {
        console.log(err);
      } else {
          listData.push(result[0].timestamp);
          listData.push(result);
          sync = false;
      }
    });
  while(sync) {require("deasync").sleep(2000);}
  return listData;
 }

function dateDifference(oldDate) {
  var oldDate = new Date(oldDate);
  var todaysDate = new Date();
  var diffDays = parseInt((todaysDate - oldDate) / (1000 * 60 * 60 * 24)); //gives day difference
  return diffDays;
 }

function restFunction(req,res) {
```

```
    if (req.session) {
        var title = req.body.title;
        var fromDate = req.body.fromDate;
        var toDate = req.body.toDate;
        var newList = req.body.yearList;
        var date = titleDate(title);
        var article = date[1];

        compareDate = dateDifference(date[0])
        stringDate = JSON.stringify(date[0]);
        stringDate = stringDate.replace(/["']/g, "");
        var endDate = new Date();
        endDate.setDate(endDate.getDate() - 1);
        stringEndDate = JSON.stringify(endDate);
        stringEndDate = stringEndDate.replace(/["']/g,"");
        var rev = APIData(title, stringDate, stringEndDate, article, fromDate, toDate, newList);
        return res.status(200).send({rev});
    } else {
        return res.status(500).send({message: 'Unauthorized Access'});
    }
}

function APIData(title, startDate, endDate, article, fromDate, toDate, newList) {
    let individualBarData = {};
    let individualPieData = {};
    let individualUser = {};
    let mergedObject = {};
    let yearList = [];
    let firstUser =[];
    let secondUser = [];
    let thirdUser = [];
    let fourthUser = [];
    let fifthUser = [];
    var sync = true;
    var finalUser = {};
    var wikiEndpoint = "https://en.wikipedia.org/w/api.php";
    var parameters = [
        "rvdir=newer",
        "action=query",
        "prop=revisions",
        "rvlimit=500",
        "rvprop=ids|flags|user|userid|timestamp",
        "formatversion=2",
        "format=json"
    ]
    parameters.unshift("rvend=" + endDate);
    parameters.unshift("rvstart=" + startDate);
    parameters.unshift("titles=" + title);

    var url = wikiEndpoint + "?" + parameters.join("&");
    var options = {
        url: url,
        method: 'GET',
        headers: {
            'Accept': 'application/json',
```

```
            'Accept-Charset': 'utf-8'
        }
    };
    request(options, function (err, res, data){
        if (err) {
            console.log('Error:', err);
        } else if (res.statusCode !== 200) {
            console.log('Error status code:', res.statusCode);
        } else {
            json = JSON.parse(data);
            pages = json.query.pages
            revisions = pages[Object.keys(pages)[0]].revisions;
            if(revisions.length == 500) {
                var last = revisions[revisions.length - 1].timestamp;
                new_data  = APIData(title,last,endDate);
                var data3 = Object.assign({}, revisions, new_data);
                return data3;
            } else
            {
                var users=[];
                var rev = [];
                var timestamp = [];
                for (i in revisions){
                    users.push(revisions[i].user);
                    rev.push(revisions[i].revid);
                    timestamp.push(revisions[i].timestamp);
                }
                uniqueUsers = new Set(users);
                var beforeMerge = {};
                finalLength = article.length + revisions.length;
                var k = 0;
                for(let j = revisions.length; j < finalLength; j++) {
                    beforeMerge[j] = article[k];
                    k += 1;
                }
                mergedObject = Object.assign({}, revisions, beforeMerge);
                var merged = [];
                let admin_merge = [];
                var botUsers = [];
                concat(['admins/admin_active.txt', 'admins/admin_former.txt', 'admins/admin_inactive.txt', 'admins/admin_semi_active.txt'])
                    .then((result) => {
                        admin_merge = result.split('\n');
                        fs.readFile('admins/bot.txt', function(err, f){
                            botUsers = f.toString().split('\n');
                            merged = admin_merge.concat(botUsers);
                            count = 0;
                            uniqueUser = [];
                            if (toDate) {
                                let newObject = mergedObject;
                                mergedObject = {}
                                Object.values(newObject).forEach((element, i) => {
                                    year = parseInt(element.timestamp)
                                    if (newList.indexOf(year) > -1) {
                                        mergedObject[i] = element;
                                    }
```



```
166                 });
167             }
168             Object.values(mergedObject).forEach(element => {
169                 if(merged.indexOf(element.user) === -1) {
170                     if(!element.anon){
171                         if(uniqueUser[element.user]){
172                             uniqueUser[element.user] += 1;
173                         } else {
174                             uniqueUser[element.user] = 1;
175                         }
176                     }
177                 }
178             });
179             admin = {};
180             bot = {};
181             regular = {};
182             anon = {};
183             Object.values(mergedObject).forEach(elementBar => {
184                 year = parseInt(elementBar.timestamp)
185                 if (yearList.indexOf(year) === -1) {
186                     yearList.push(year);
187                 }
188                 if (admin_merge.indexOf(elementBar.user) > -1) {
189                     if (admin[year]) {
190                         admin[year] = {'_id': year, rev: admin[year].rev + 1}
191                     } else {
192                         admin[year] = {'_id': year, rev: 1}
193                     }
194                 } else if (botUsers.indexOf(elementBar.user) > -1) {
195                     if (bot[year]) {
196                         bot[year] = {'_id': year, rev: bot[year].rev + 1}
197                     } else {
198                         bot[year] = {'_id': year, rev: 1}
199                     }
200                 } else if (merged.indexOf(elementBar.user) === -1) {
201                     if (!elementBar.anon) {
202                         if (regular[year]) {
203                             regular[year] = {'_id': year, rev: regular[year].rev + 1}
204                         } else {
205                             regular[year] = {'_id': year, rev: 1}
206                         }
207                     } else {
208                         if (anon[year]) {
209                             anon[year] = {'_id': year, rev: anon[year].rev + 1}
210                         } else {
211                             anon[year] = {'_id': year, rev: 1}
212                         }
213                     }
214                 }
215             });
216             for (let i = 0; i < yearList.length; i++) {
217                 if (!(yearList[i] in admin)) {
218                     admin[yearList[i]] = {'_id': yearList[i], rev: 0}
219                 }
220                 if (!(yearList[i] in bot)) {
```

```
                    bot[yearList[i]] = {'_id': yearList[i], rev: 0}
                }
                if (!(yearList[i] in regular)) {
                    regular[yearList[i]] = {'_id': yearList[i], rev: 0}
                }
                if (!(yearList[i] in anon)) {
                    anon[yearList[i]] = {'_id': yearList[i], rev: 0}
                }
            }
            individualBarData['Admin'] = admin;
            individualBarData['Bot'] = bot;
            individualBarData['Regular'] = regular;
            individualBarData['Anonymous'] = anon;
            adminRev = {};
            botRev = {};
            regularRev = {};
            anonRev = {};
            Object.values(mergedObject).forEach((elementPie) => {
                if (admin_merge.indexOf(elementPie.user) > -1) {
                    if(adminRev[0]) {
                        adminRev[0] = {'_id': 'Administrator', rev: adminRev[0].rev + 1};
                    } else {
                        adminRev[0] = {'_id': 'Administrator', rev: 1};
                    }
                } else if (botUsers.indexOf(elementPie.user) > -1) {
                    if (botRev[0]) {
                        botRev[0] = {'_id': 'Bot', rev: botRev[0].rev + 1};
                    } else {
                        botRev[0] = {'_id': 'Bot', rev: 1};
                    }
                } else if (merged.indexOf(elementPie.user) === -1) {
                    if (!elementPie.anon) {
                        if (regularRev[0]) {
                            regularRev[0] = {'_id': 'Regular User', rev: regularRev[0].rev + 1};
                        } else {
                            regularRev[0] = {'_id': 'Regular User', rev: 1};
                        }
                    } else {
                        if (anonRev[0]) {
                            anonRev[0] = {'_id': 'Anonymous', rev: anonRev[0].rev + 1};
                        } else {
                            anonRev[0] = {'_id': 'Anonymous', rev: 1};
                        }
                    }
                }
            });
            individualPieData = {adminRev, botRev, regularRev, anonRev};
            keysSorted = Object.keys(uniqueUser).sort(function(a,b){return uniqueUser[a]-uniqueUser[b]});
            var sortedUser = keysSorted.reverse();
            var revUser = sortedUser.splice(0,5);
            firstUser.push(revUser[0]);
            secondUser.push(revUser[1]);
            thirdUser.push(revUser[2]);
            fourthUser.push(revUser[3]);
            fifthUser.push(revUser[4]);
```

```
276                         var userList = [firstUser, secondUser, thirdUser, fourthUser, fifthUser];
277                         for (i in revUser) {
278                             finalUser[i] ={user: revUser[i], revisions: uniqueUser[revUser[i]]};
279                         }
280                         Object.values(mergedObject).forEach((elementInd, i) => {
281                             year = parseInt(elementInd.timestamp)
282                             if (firstUser.indexOf(elementInd.user) > -1) {
283                                 if (individualUser[year]) {
284                                     individualUser[year] = {_id: elementInd.user, rev: individualUser[year].rev + 1}
285                                 } else {
286                                     individualUser[year] = {_id: elementInd.user, rev: 1}
287                                 }
288                             }
289                         });
290                     sync = false;
291                 });
292             });
293         }
294     }
295 });
296 while(sync) {require("deasync").sleep(5000);}
297 return [finalUser, revisions.length, finalLength, individualBarData, individualPieData,
298     individualUser, yearList, Object.values(mergedObject).length];
299 }
300
```

author.controller.js

```
var Revision = require('../model/revisions');

module.exports = function(app) {
  app.get('/api/author/author-details', authorsArticles);
}

function authorsArticles(req,res) {
  author_name = req.query.author;
  if (req.session) {
    Revision.aggregate([
      {$match:{user:author_name}},
      {$group:{_id:"$title",rev:{$sum:1}, timelist:{$addToSet: "$timestamp"}}},
      {$project: {_id:1, rev:1, timelist:1}},
      {$sort:{rev:-1}}
      ],function(err, result) {
        if(err) {
          console.log('err');
        } else {
          if (result.length !== 0) {
            return res.status(200).send(result);
          } else {
            return res.status(500).send({message: 'User not found'});
          }
        }
    });
  } else {
    return res.status(500).send({message: 'Unauthorized Access'});
  }

}
```

Additionally, I have included two more files:

1. chart.service.ts - to fetch and store bar chart and pie chart content in local storage (cache) during user registration time, so that user does not need to wait to see the content when he goes to overall-analytics route.

```typescript
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Router } from '@angular/router';
import { LocalStorageService } from 'ngx-webstorage';
import { Subject, ReplaySubject } from 'rxjs';

@Injectable()
export class ChartService {
    overallBarData: Array<string> = [];
    overallPieData: any;
    highestLength: any;
    userLoggedIn = false;
    user: User;
    userObservable: Subject<User> = new ReplaySubject<User>(1);

    constructor(private localStorageService: LocalStorageService, private http: HttpClient, private router: Router) {
    }

    getOverallPieData() {
        if (this.overallPieData) {
            return this.localStorageService.retrieve('overall.pie');
        } else {
            this.http.get('/api/overall/pie-chart')
                .subscribe(pie => {
                    this.overallPieData = pie;
                    this.localStorageService.store('overall.pie', this.overallPieData);
                    return this.overallPieData;
                });
        }
    }

    getOverallBarData() {
        if (Object.keys(this.overallBarData).length !== 0) {
            return this.localStorageService.retrieve('overall.bar');
        } else {
            this.http.get('/api/overall/bar-chart')
                .subscribe(bar => {
                    const convertedAdmin = Object.assign({}, ...bar['Admin'].map(object => ({[object._id]: object})));
                    const convertedBot = Object.assign({}, ...bar['Bot'].map(object => ({[object._id]: object})));
                    const convertedRegular = Object.assign({}, ...bar['Regular'].map(object => ({[object._id]: object})));
                    const convertedAnonymous = Object.assign({}, ...bar['Anonymous'].map(object => ({[object._id]: object})));
                    const lengthList = [bar['Admin'].length, bar['Bot'].length, bar['Regular'].length, bar['Anonymous'].length];
                    let max = 0;
                    for (let i = 0; i < lengthList.length; i++) {
                        const num = lengthList[i];
                        if (num >= max) {
                            max = num;
                        } else {
                            continue;
                        }
                    }
                    if (lengthList.indexOf(max) === 0) {
                        this.highestLength = bar['Admin'];
                    } else if (lengthList.indexOf(max) === 1) {
                        this.highestLength = bar['Bot'];
```

```
                } else if (lengthList.indexOf(max) === 2) {
                    this.highestLength = bar['Regular'];
                } else if (lengthList.indexOf(max) === 3) {
                    this.highestLength = bar['Anonymous'];
                }
                for (let i = 0; i < this.highestLength.length; i++) {
                    if (!(this.highestLength[i]._id in convertedAdmin)) {
                        convertedAdmin[this.highestLength[i]._id] = {_id: this.highestLength[i]._id, rev: 0};
                    }
                    if (!(this.highestLength[i]._id in convertedBot)) {
                        convertedBot[this.highestLength[i]._id] = {_id: this.highestLength[i]._id, rev: 0};
                    }
                    if (!(this.highestLength[i]._id in convertedRegular)) {
                        convertedRegular[this.highestLength[i]._id] = {_id: this.highestLength[i]._id, rev: 0};
                    }
                    if (!(this.highestLength[i]._id in convertedAnonymous)) {
                        convertedAnonymous[this.highestLength[i]._id] = {_id: this.highestLength[i]._id, rev: 0};
                    }
                }
                this.overallBarData['Admin'] = convertedAdmin;
                this.overallBarData['Bot'] =  convertedBot;
                this.overallBarData['Regular'] = convertedRegular;
                this.overallBarData['Anonymous'] = convertedAnonymous;
                this.localStorageService.store('overall.bar', this.overallBarData);
            });
        }
    }

    setUser(user) {
        this.user = user;
        this.localStorageService.store('user-service.user', this.user);
        this.userObservable.next(this.user);
    }

    logout() {
        this.http.post('/api/user/logout', {})
            .subscribe(() => {
                this.localStorageService.clear('user-service.user');
                this.userObservable.next(undefined);
                this.router.navigateByUrl('/');
            });
    }
}

export interface User {
    firstName: string;
    lastName: string;
    email: string;
    isLoggedIn: boolean;
}
```

2. auth.guard.ts - to protect unauthorized access and redirect to register from client side.

```typescript
import { Injectable } from '@angular/core';
import {
  CanActivate,
  ActivatedRouteSnapshot,
  RouterStateSnapshot,
} from '@angular/router';
import { Observable } from 'rxjs';
import { map, take, tap } from 'rxjs/operators';
import { ChartService } from './chart.service';
import { Router } from '@angular/router';

@Injectable({
  providedIn: 'root',
})
export class AuthGuard implements CanActivate {
  constructor(private chartService: ChartService, private router: Router) {}

  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot
  ): Observable<boolean> | Promise<boolean> | boolean {
    return this.chartService.userObservable.pipe(
      take(1),
      map(user => {
        const authenticated: boolean = !!user;
        if (!authenticated) {
          this.router.navigate(['/register']);
          return false;
        } else {
          return true;
        }
      })
    );
  }
}

```