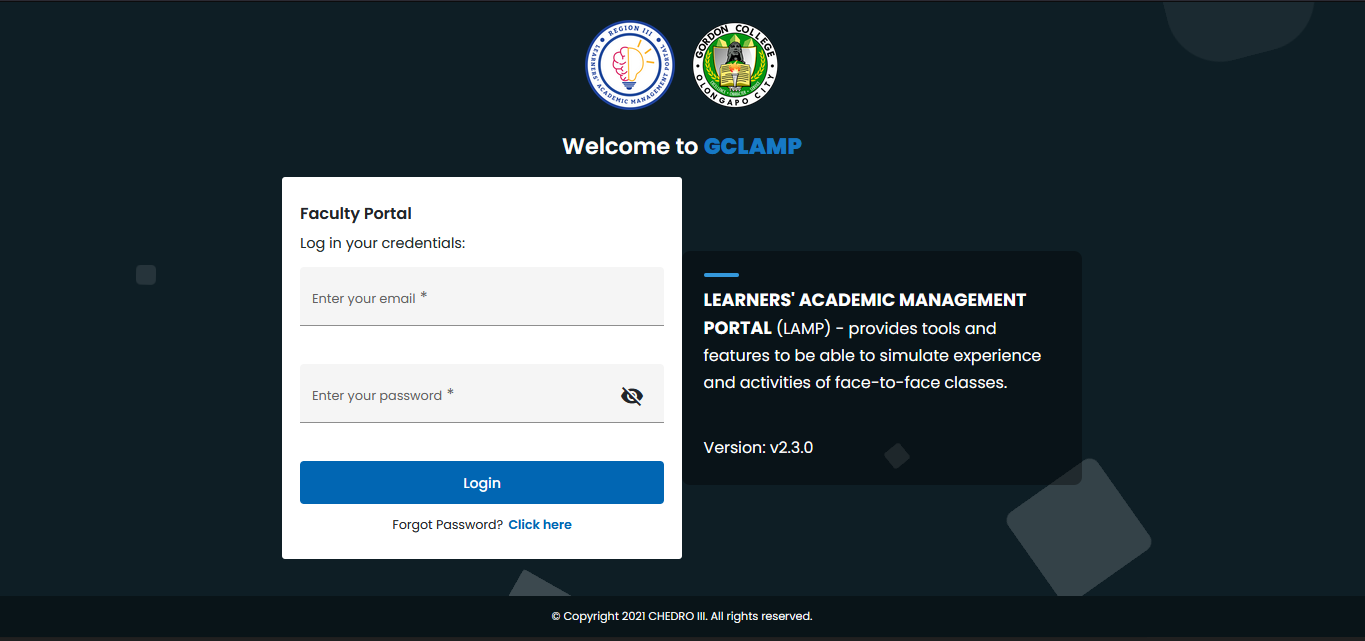
**FACULTY PANEL**

****

**LOGIN HTML**

|  |
| --- |
| **<div** class="background"**>**  **<div** class="wrapper"**>**  **<div** class="header"**>**  **<div** class="header\_\_contain"**>**  **<img** draggable="false" src="assets/images/r3lamp.png" alt="Gordon College"**>**  **<img** draggable="false" src="assets/images/logo.png" alt="lamp-logo"**>**  **</div>**  **<h2>**Welcome to **<strong** style="color: var(--clr-third)"**>** {{ \_ds.prefix + "LAMP"}} **</strong></h2>**  **</div>**  **<section>**  **<div** class="login"**>**  **<p** class="login\_\_headertitle"**>**Faculty Portal**</p>**  **<small** class="login\_\_headersub"**>**Log in your credentials:**</small>**  **<form** class="login\_\_form" [formGroup]="LoginFormGroup"**>**  **<mat**-form-field appearance="fill" class="login\_\_input"**>**  **<mat**-label**>**Enter your email</mat-label>  **<input** formControlName="user\_email" autocomplete="email" type="email" matInput  placeholder="ex. juandelacruz@gordoncollege.edu.ph" required [readonly]="notFound"**>**  **<mat**-error \*ngIf="LoginFormGroup.controls.user\_email.invalid"**>**{{ getErrorMessage() }}</mat-error>  </mat-form-field>  **<mat**-form-field appearance="fill" class="login\_\_input"**>**  **<mat**-label**>**Enter your password</mat-label>  **<input** placeholder="Password" formControlName="user\_pass" autocomplete="current-password" id="password"  name="param2" matInput [type]="hide ? 'password' : 'text'" required [readonly]="notFound"**>**  **<button** type="button" mat-icon-button matSuffix (click)="hide = !hide" [attr.aria-label]="'Hide password'"  [attr.aria-pressed]="hide"**>**  **<mat**-icon**>**{{hide ? 'visibility\_off' : 'visibility'}}</mat-icon>  **</button>**  **<mat**-error \*ngIf="LoginFormGroup.controls.user\_pass.invalid"**>**{{ getPassErrorMessage() }}</mat-error>  </mat-form-field>  **<p** \*ngIf="errMsg != ''" class="text-danger text-center"**>**  **<strong>**{{ errMsg }}**</strong>**  **</p>**  *<!-- <mat-progress-bar \*ngIf="\_user.isLoading()" mode="indeterminate"></mat-progress-bar> -->*  **<button** type="submit" [disabled]="\_user.isLoading()" (click)="onSubmit()" class="btn-default" mat-flat-button \*ngIf="!notFound"**>**  **<span** \*ngIf="!\_user.isLoading()"**>**Login**</span>**  **<span** \*ngIf="\_user.isLoading()"**>**Logging in...**</span>**  **</button>**  **<button** [disabled]="\_user.isLoading()" class="btn-error" mat-flat-button \*ngIf="notFound"**>**  **<small>**Server Under Maintenance, Please Contact Administrator**</small>**  **</button>**  **</form>**  **<div** class="text-center"**>**  **<small>**Forgot Password?**</small>**  **<small** style="color: var(--btn-primary); margin-left: 5px; font-weight: 600; cursor: pointer;"  data-bs-toggle="modal" data-bs-target="#forgotPassModal"**>**Click  here**</small>**  **</div>**  **</div>**  **<article>**  **<div** class="line"**></div>**  **<p><strong>**LEARNERS' ACADEMIC MANAGEMENT PORTAL**</strong>** (LAMP) - provides tools and features to be able  to simulate experience and activities of face-to-face classes.  **</p><br>**  **<small** style="color: white !important"**>**Version: v{{ \_ds.version }}**</small>**  **</article>**  **</section>**  **</div>**  **<footer>**  **<span>**© Copyright 2021 CHEDRO III. All rights reserved.**</span>**  **</footer>**  **<ul** class="circles"**>**  **<li></li>**  **<li></li>**  **<li></li>**  **<li></li>**  **<li></li>**  **<li></li>**  **<li></li>**  **<li></li>**  **<li></li>**  **<li></li>**  **</ul>**  **</div>**  **<div** class="modal fade" id="forgotPassModal" data-backdrop="static" tabindex="-1" aria-labelledby="exampleModalLabel"  aria-hidden="true"**>**  **<div** class="modal-dialog modal-dialog-centered"**>**  **<div** class="modal-content"**>**  **<div** class="modal-body modals"**>**  **<div** class="modals\_\_header"**>**  **<strong** class="modals\_\_title " id="exampleModalLabel"**>**Forgot password?**</strong>**  **<button** type="button" data-bs-dismiss="modal" aria-label="Close" mat-icon-button**>**  **<mat**-icon**>**close</mat-icon>  **</button>**  **</div>**  **<p>** Provide the following information and we'll send you a link to reset your password**</p>**  **<form** class="form " (submit)="forgotPassword($event)" ngNativeValidate**>**  **<mat**-form-field appearance="fill" class="forms\_\_input"**>**  **<mat**-label**>**Faculty ID</mat-label>  **<input** matInput type="text" id="param1" name="param1" matInput required**>**  </mat-form-field>  **<mat**-form-field appearance="fill" class="forms\_\_input"**>**  **<mat**-label**>**Email Address</mat-label>  **<input** matInput id="param3" name="param3" type="email" required**>**  </mat-form-field>  **<mat**-form-field appearance="fill" class="forms\_\_input"**>**  **<mat**-label**>**Birthdate</mat-label>  **<input** matInput id="param2" name="param2" type="date" required**>**  </mat-form-field>  **<button** type="submit" class="btn-default submit" mat-flat-button [disabled]="notFound"**>**Submit**</button>**  **</form>**  **</div>**  **</div>**  **</div>**  **</div>** |

**LOGIN PAGE.ts**

|  |
| --- |
| **export** **class** LoginComponent **implements** OnInit {  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(Breakpoints.XSmall);  hide: **boolean** = **true**;  errMsg: **any**;  showpassword: **number** = **0**;  LoginFormGroup: **FormGroup**;  notFound: **boolean** = **false**;  **constructor**(  **public** route: **Router**,  **public** \_user: **UserService**,  **public** \_ds: **DataService**,  **private** fb: **FormBuilder**,  **private** \_commonsub: **CommonService**,  **private** breakpointObserver: **BreakpointObserver**,  **public** dialog: **MatDialog**,  ) { }  ngOnInit(): **void** {  **this**.initializeComponents();  }  initializeComponents() {  **if** (btoa('facultyState') **in** sessionStorage && **this**.\_user.getLoginState()) {  **this**.route.navigate(['main']);  }  **this**.getSettings();  **this**.LoginFormGroup = **this**.fb.group({  user\_email: ['', [Validators.required, Validators.email]],  user\_pass: ['', Validators.required],  });  }  getSettings() {  **this**.\_user.setLoading(**true**)  **this**.\_commonsub.commonSubscribe('getacadyear', **null**, **2**).then((data: **any**) => {  **this**.\_user.setSettings(data.payload[**0**]);  **this**.\_user.setLoading(**false**);  }).**catch**((error) => {  **this**.notFound = **true**;  **this**.\_user.setLoading(**false**);  })  }  getErrorMessage() {  **if** (**this**.LoginFormGroup.controls.user\_email.hasError('required')) **return** 'You must enter a value';  **if** (**this**.LoginFormGroup.controls.user\_email.hasError('email')) **return** 'Not a valid email';  **if** (**this**.LoginFormGroup.controls.user\_email.hasError('incorrect')) **return** 'EMAIL OR USERNAME - Email or password is invalid.';  }  getPassErrorMessage() {  **if** (**this**.LoginFormGroup.controls.user\_pass.hasError('required')) **return** 'You must enter a value';  **if** (**this**.LoginFormGroup.controls.user\_pass.hasError('incorrect')) **return** 'PASSWORD - Email or password is invalid.';  }  onSubmit() {  **let** load = {  param1: **this.LoginFormGroup.value.user\_email**,  param2: **this.LoginFormGroup.value.user\_pass**,  param3: **5**,  param4: **this.\_user.isMobile**() ? 'Web Access using Mobile phone' : 'Web Access using Desktop/Laptop',  }  **this**.\_user.setLoading(**true**);  **this**.\_commonsub.commonSubscribe('facultylogin', load, **2**).then((data: **any**) => {  **this**.\_user.setData('user', data.payload);  **this**.\_user.setLoading(**false**);  data.payload.reset == **1** ? **this**.Changepassword() : **this**.isAuthorized();  }).**catch**((error: **any**) => {  **this**.LoginFormGroup.controls['user\_email'].setErrors({ 'incorrect': **true** });  **this**.LoginFormGroup.controls['user\_pass'].setErrors({ 'incorrect': **true** });  **this**.\_user.setLoading(**false**);  });  }  Changepassword() {  **const** dialogRef = **this**.dialog.open(ChangepasswordComponent, {  panelClass: '',  data: { type: 'changepassinfirstlogin' },  disableClose: **true**,  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('60%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result) **this**.isAuthorized();  });  }  forgotPassword(e) {  e.preventDefault();  **let** f = e.target.elements;  **let** param1 = f.param1.value;  **let** param2 = f.param2.value;  **let** param3 = f.param3.value;  **let** load = {  param1:**param1**,  param2:**param2**,  param3:**param3**  }  **this**.\_ds.\_httpRequest('forgotpass', load, **2**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  *// console.log(dt, dt.status.message)*  **this**.sendMail(param3,dt.payload.pass);  f.param1.value = **null**;  f.param2.value = **null**;  f.param3.value = **null**;  console.log(dt);  },er=>{  er = **this**.\_user.\_decrypt(er.error.a);  console.log(er);    });  }  sendMail(email,password){  **let** load = {  emailArray:[email],  body: "Hello, Your new password is "+password,  subject:"Reset Password"  }  console.log(load);    **this**.\_ds.\_httpRequest('mailer',load,**5**).subscribe((data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a);  *// console.log("success");*    },er =>{  er = **this**.\_user.\_decrypt(er.error.a);  console.log(er);    })  }  isAuthorized() {  **this**.\_user.setLoginState(**true**);  **this**.route.navigate(['main']);  *// this.\_user.setLoading(false);*  }  } |

**CLASSES**

|  |
| --- |
| **export** **class** ClassesComponent **implements** OnInit {  date = **new** Date();  allClasses: **any**[] = [];  classes: **any**[] = [];  scheduleObject: **any**[] = [];  showLoader: **boolean** = **true**;  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(Breakpoints.XSmall);  **constructor**(**private** breakpointObserver: **BreakpointObserver**, **public** dialog: **MatDialog**, **private** \_ds: **DataService**, **private** \_user: **UserService**) { }  ngOnInit(): **void** {  **this**.initializeComponents();  setInterval(() => {  **this**.date = **new** Date();  }, **1000**);  }  initializeComponents() {  **this**.getFacultyClasses();  }  enterClassroom(item) {  **this**.\_user.setClassroomInfo(item);  }  getFacultyClasses() {  **let** load = {  data: {  email: **this.\_user.getUserEmail**(),  acadyear: **this.\_user.getSettings**().acadyear\_fld,  semester: **this.\_user.getSettings**().sem\_fld  }  }  **this**.\_ds.\_httpRequest('getclasslist', load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.allClasses = **this**.sortClass(dt.payload);  **this**.classes = **this**.allClasses;  **this**.getClassSchedules();  **this**.showLoader = **false**;  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  **this**.classes = er.payload;  });  }  filterClass(ev) {  **let** searchItem: **string** = ev.target.value;  **this**.classes = **this**.allClasses.filter((x)=>{  **return** (  x.classcode\_fld.toUpperCase().includes(searchItem.toUpperCase()) ||  x.subjcode\_fld.toUpperCase().includes(searchItem.toUpperCase()) ||  x.subjdesc\_fld.toUpperCase().includes(searchItem.toUpperCase())  );  });  }  sortClass(payload): any {  **let** arrDays: **string**[] = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'];  **let** tmp: **any** = payload;  **let** items: **any**[]=[];  arrDays.forEach(x => {  tmp.forEach(y => {  **let** found=**0**;  **if**(y.day\_fld.includes(x)) {  items.forEach(z=>{  **if**(y.classcode\_fld==z.classcode\_fld) found = **1**;  });  **if**(found!=**1**) items.push(y);  }  });  });  **return** items  }  getClassSchedules() {  **let** mon = [], tue = [], thu = [], wed = [], fri = [], sat = [], sun = []  **if** (**this**.classes != **null**) {  **this**.classes.forEach((element) => {  **if** (element.day\_fld.match('Mon')) {  mon.push({  day\_fld: 'mon',  starttime\_fld: **element.starttime\_fld**,  endtime\_fld: **element.endtime\_fld**,  subjdesc\_fld: **element.subjdesc\_fld**,  subjcode\_fld: **element.subjcode\_fld**,  block\_fld: **element.block\_fld**  })  }  **if** (element.day\_fld.match('Tue')) {  tue.push({  day\_fld: 'tue',  starttime\_fld: **element.starttime\_fld**,  endtime\_fld: **element.endtime\_fld**,  subjdesc\_fld: **element.subjdesc\_fld**,  subjcode\_fld: **element.subjcode\_fld**,  block\_fld: **element.block\_fld**  })  }  **if** (element.day\_fld.match('Wed')) {  wed.push({  day\_fld: 'wed',  starttime\_fld: **element.starttime\_fld**,  endtime\_fld: **element.endtime\_fld**,  subjdesc\_fld: **element.subjdesc\_fld**,  subjcode\_fld: **element.subjcode\_fld**,  block\_fld: **element.block\_fld**  })  }  **if** (element.day\_fld.match('Thu')) {  thu.push({  day\_fld: 'thu',  starttime\_fld: **element.starttime\_fld**,  endtime\_fld: **element.endtime\_fld**,  subjdesc\_fld: **element.subjdesc\_fld**,  subjcode\_fld: **element.subjcode\_fld**,  block\_fld: **element.block\_fld**  })  }  **if** (element.day\_fld.match('Fri')) {  fri.push({  day\_fld: 'fri',  starttime\_fld: **element.starttime\_fld**,  endtime\_fld: **element.endtime\_fld**,  subjdesc\_fld: **element.subjdesc\_fld**,  subjcode\_fld: **element.subjcode\_fld**,  block\_fld: **element.block\_fld**  })  }  **if** (element.day\_fld.match('Sat')) {  sat.push({  day\_fld: 'sat',  starttime\_fld: **element.starttime\_fld**,  endtime\_fld: **element.endtime\_fld**,  subjdesc\_fld: **element.subjdesc\_fld**,  subjcode\_fld: **element.subjcode\_fld**,  block\_fld: **element.block\_fld**  })  }  **if** (element.day\_fld.match('Sun')) {  sun.push({  day\_fld: 'sun',  starttime\_fld: **element.starttime\_fld**,  endtime\_fld: **element.endtime\_fld**,  subjdesc\_fld: **element.subjdesc\_fld**,  subjcode\_fld: **element.subjcode\_fld**,  block\_fld: **element.block\_fld**  })  }  });  **this**.scheduleObject.push(mon, tue, wed, thu, fri, sat, sun);  }  }  ViewSchedule() {  **let** data = {  "type": 'schedule',  "data": **this**.scheduleObject,  };  **const** dialogRef = **this**.dialog.open(ViewDataOnlyComponent, {  panelClass: 'dialogpadding',  maxWidth: '90vw',  maxHeight: '90vh',  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('100%', 'auto');  } **else** {  dialogRef.updateSize('100%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  });  }  } |

**CLASSFEED**

|  |
| --- |
| **export** **class** ClassfeedComponent **implements** OnInit {  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(Breakpoints.XSmall);  topicObj: **any** = [];  getRecipients = **new** getRecipients();  pollToggle: **boolean** = **false**;  pollOptions: **any** = [];  postobject: **any** = [];  videoobj = [];  activities: **any** = [];  resourcesObj: **any** = [];  filePreviewAndUpload: **any** = [];  postcontent: **string** = '';  acceptedFiles = ".doc,.docx,.html,.htm,.odt,.pdf,.xls,.xlsx,.ods,.ppt,.pptx,.txt";  filetype: **number** = **0**;  loadingclassfeed: **boolean**;  loadmaterials: **boolean**;  **constructor**(  **private** breakpointObserver: **BreakpointObserver**,  **public** dialog: **MatDialog**,  **public** \_ds: **DataService**,  **public** \_user: **UserService**,  **private** \_router: **Router**,  **public** \_uploadservice: **UploadingService**,  **private** sanitizer: **DomSanitizer**,  **private** \_commonservice: **CommonService**) {  }  ngOnInit(): **void** {  **this**.initializeComponents();  }  createQuiz() {  **this**.\_user.setQuiz(**null**);  **this**.\_router.navigate(['/main/creator/quiz']);  }  resetFunction() {  **this**.filePreviewAndUpload = [];  **this**.postcontent = '';  **this**.filetype = **0**;  }  addOption(e) {  e.preventDefault();  **let** f = e.target.elements;  **this**.pollOptions.push(f.option\_name.value);  f.option\_name.value = **null**;  }  removePollOption(i) {  **this**.pollOptions.splice(i, **1**);  }  initializeComponents() {  **this**.getClassPost();  **this**.getClassResources();  **this**.getActivities();  **this**.getClassTopics();  **this**.\_uploadservice.files = [];  }  getFile(event: **any**, type: **number**) {  **this**.filePreviewAndUpload = **this**.\_uploadservice.getFilePreview(event);  **if** (**this**.filePreviewAndUpload.length > **0**) {  **this**.filetype = type;  }  }  removeFilePreviews(i) {  **this**.filePreviewAndUpload.splice(i, **1**);  **if** (**this**.filePreviewAndUpload.length == **0**) {  **this**.filetype = **0**;  }  }  editpoll(item, state, i) {  **let** recipients = **this**.getRecipients.memberMap(**this**.\_user.getClassMembers().student);  **let** load = {  data: {  "withfile\_fld": state,  }, notif: {  id: **this.\_user.getUserID**(),  recipient: **recipients.join**(':'),  message: **this.\_user.getUserFullname**() + (state == **3** ? ' Opened ' : ' Closed ') + "Poll in " + **this**.\_user.getClassroomInfo().subjdesc\_fld,  module: 'Classroom',  }  };  **this**.\_ds.\_httpRequest('editpost/' + item.postcode\_fld + '/' + 'post', load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.addfiles(dt.payload)  .then(data => { **this**.postobject[i] = data[**0**] });  **this**.\_user.openSnackBar((state == **3** ? ' Opened Poll Successfully' : ' Closed Poll Successfully'), **null**, **2000**);  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  });  }  togglePoll() { **this**.pollToggle = !**this**.pollToggle; }  async getClassPost() {  **let** load = {  data: {  classcode: **this.\_user.getClassroomInfo**().classcode\_fld,  type: 'post',  }  }  await **this**.\_ds.\_httpRequest('getcpost', load, **1**).subscribe(async (dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.addfiles(dt.payload).then(data => **this**.postobject = data);  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  **this**.postobject = [];  })  }  async addfiles(array) {  array.forEach(e => {  e.images\_fld = [];  e.video\_fld = [];  e.attachment\_fld = [];  e.pollchoices\_fld = [];  e.embedvideo\_fld = [];  e.pollresponse\_fld = [];  e.authorid\_fld == **this**.\_user.getUserID() ? e.author\_owner = **true** : e.author\_owner = **false**;  **if** (e.dir\_fld) {  **let** files = **this**.\_uploadservice.splitFilestring(e.dir\_fld);  files.map(f => {  **if** (f.path.includes('gif') || f.path.includes('jpeg') || f.path.includes('jpg') || f.path.includes('png') || f.path.includes('bmp')) {  e.images\_fld.push(f);  }  **if** (f.path.includes('mp4') || f.path.includes('m4v') || f.path.includes('avi') || f.path.includes('mpg') || f.path.includes('wmv')) {  e.video\_fld.push(f);  }  **if** (f.path.includes('sql') || f.path.includes('doc') || f.path.includes('docx') || f.path.includes('html') || f.path.includes('htm') || f.path.includes('pdf') || f.path.includes('xls') || f.path.includes('xlsx') || f.path.includes('ods') || f.path.includes('csv') || f.path.includes('ppt') || f.path.includes('pptx') || f.path.includes('txt')) {  e.attachment\_fld.push(f);  }  });  }  **if** (e.content\_fld.includes('youtube.com') || e.content\_fld.includes('google.com/file/d/')) {  e.embedvideo\_fld = **this**.getVideoEmbedURL(e.content\_fld);  }  **if** ((e.withfile\_fld == **3** || e.withfile\_fld == **4**)) {  **let** pollObj = **this**.\_uploadservice.splitPollContent(e);  e.content\_fld = pollObj.content\_fld;  e.pollchoices\_fld = pollObj.pollchoices\_fld;  **let** load = {  data: {  postcode: **e.postcode\_fld**,  filepath: {  options: []  }  }  }  e.pollchoices\_fld.map((chc) => {  load.data.filepath.options.push({ "choice": chc, "votes": **0**, "respondents": [] });  })  **this**.\_commonservice.commonSubscribe('getpoll', load, **1**)  .then(async (dt: **any**) => {  e.pollresponse\_fld = await dt.payload.options;  })  }  });  **return** **new** Promise((resolve) => resolve(array));  }  editPollChoices(item, i) {  item.istoedit\_fld = **1**;  **const** dialogRef = **this**.dialog.open(ViewpollComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100%',  disableClose: **true**,  data: **item**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('50%', 'auto');  }  });  }  getActivities() {  **this**.loadmaterials = **true**;  **this**.\_ds.\_httpRequest('getcpost', { data: { classcode: **this.\_user.getClassroomInfo**().classcode\_fld, type: "" } }, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.activities = dt.payload;  **this**.loadmaterials = **false**;  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  **this**.activities = [];  **this**.loadmaterials = **false**;  })  }  getClassResources() {  **this**.loadmaterials = **true**;  **let** load = {  data: { classcode: **this.\_user.getClassroomInfo**().classcode\_fld, }  }  **this**.\_ds.\_httpRequest('getreslist', load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.resourcesObj = dt.payload;  **this**.loadmaterials = **false**;  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  **this**.loadmaterials = **false**;  **this**.resourcesObj = [];  })  }  *// Add post in Class feed:*  async onSubmit(e) {  **this**.loadingclassfeed = **true**;  e.preventDefault();  **let** recipients = **this**.getRecipients.memberMap(**this**.\_user.getClassMembers().student);  **if** (**this**.pollToggle) {  **let** load = {  data: {  content\_fld: **this.postcontent** + '($)' + **this**.pollOptions.join('(\*)'),  authorid\_fld: **this.\_user.getUserID**(),  classcode\_fld: **this.\_user.getClassroomInfo**().classcode\_fld,  withfile\_fld: **3**,  }, notif: {  id: **this.\_user.getUserID**(),  recipient: **recipients.join**(':'),  message: **this.\_user.getUserFullname**() + " Added Poll in " + **this**.\_user.getClassroomInfo().subjdesc\_fld,  module: 'Classroom',  }  }  **this**.\_commonservice.commonSubscribe('addpost', load, **1**)  .then(async (data: **any**) => {  **this**.addfiles(data.payload).then(dt => **this**.postobject.unshift(dt[**0**]))  **this**.loadingclassfeed = **false**;  **this**.pollToggle = !**this**.pollToggle;  **this**.pollOptions = [];  **this**.resetFunction();  **this**.\_user.openSnackBar('Post has been added.', **null**, **2000**);  });  } **else** {  **this**.\_uploadservice.uploadFiles(**this**.filePreviewAndUpload).then((files => {  **let** load = {  data: {  content\_fld: **this.postcontent**,  authorid\_fld: **this.\_user.getUserID**(),  classcode\_fld: **this.\_user.getClassroomInfo**().classcode\_fld,  withfile\_fld: **this.filetype**,  dir\_fld: **files**,  }, notif: {  id: **this.\_user.getUserID**(),  recipient: **recipients.join**(':'),  message: **this.\_user.getUserFullname**() + " Added Post in " + **this**.\_user.getClassroomInfo().subjdesc\_fld,  module: 'Classroom',  }  }  **this**.\_commonservice.commonSubscribe('addpost', load, **1**)  .then(async (data: **any**) => {  **this**.addfiles(data.payload).then(dt => **this**.postobject.unshift(dt[**0**]))  **this**.resetFunction();  **this**.loadingclassfeed = **false**;  **this**.\_user.openSnackBar('Post has been added.', **null**, **2000**);  });  }));  }  }  deleteRecords(type, item, i) {  **let** data = {  type: **type**,  data: **item**,  };  **const** dialogRef = **this**.dialog.open(DeleteRecordComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('30%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result) {  **this**.postobject.splice(i, **1**);  }  });  }  deleteAct(type, item, i) {  **let** data = {  type: **type**,  data: **item**,  };  **const** dialogRef = **this**.dialog.open(DeleteRecordComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('20%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result) {  **this**.activities.splice(i, **1**);  }  });  }  deleteRes(type, item, i) {  **let** data = {  type: **type**,  data: **item**,  };  **const** dialogRef = **this**.dialog.open(DeleteRecordComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('20%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result) {  **this**.resourcesObj.splice(i, **1**);  }  });  }  editResources(item, i) {  **let** data = {  "type": 'edit\_resources',  "data": item,  "topicdata": **this**.topicObj,  };  **const** dialogRef = **this**.dialog.open(CrudResourcesComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**,  disableClose: **true**,  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('90%', 'auto');  } **else** {  dialogRef.updateSize('50%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result != **undefined**) {  **this**.resourcesObj[i] = result;  }  });  }  getClassTopics() {  **let** load = {  data: { classcode: **this.\_user.getClassroomInfo**().classcode\_fld, }  };  **this**.\_ds.\_httpRequest('gettopic', load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.topicObj = dt.payload;  **this**.\_user.setTopic(dt.payload);  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  **this**.topicObj = [];  });  }  editActivity(item, i) {  **let** data = {  "type": 'edit\_act',  "data": item,  "topicdata": **this**.topicObj,  };  **const** dialogRef = **this**.dialog.open(CrudActivityComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('50%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result != **undefined**) {  **this**.activities[i] = result;  }  });  }  Commentdialog(postcode, i) {  **let** data = {  "type": 'post-commments',  "postcode": postcode,  };  **const** dialogRef = **this**.dialog.open(CrudCommentsComponent, {  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('50%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **this**.postobject[i].commentcount = result  });  }  ViewResource(item) {  **let** data = {  "type": 'Resources',  data: **item**,  };  **const** dialogRef = **this**.dialog.open(ViewDataOnlyComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('50%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  });  }  editPost(item, i) {  **let** data = {  "type": 'edit\_post',  data: **item**,  };  **const** dialogRef = **this**.dialog.open(CrudPostComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**,  disableClose: **true**,  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('45%', 'auto');  }  });  dialogRef.afterClosed().subscribe(async (result) => {  responsiveDialogSubscription.unsubscribe();  **if** (result == **undefined**) **return**  result = await **this**.addfiles(result);  result = await **this**.addVideoUrl(result);  **this**.postobject[i] = result[**0**];  });  }  viewActivity(item) {  window.sessionStorage.removeItem(btoa('activityinfo'));  **this**.\_user.setActivityInfo(item);  }  deletePost(data, i) {  **let** load = {  isdeleted\_fld: **1**  }  *// Bugs Here NO RETURN DATA*  **this**.\_ds.\_httpRequest('editpost/' + data.postcode\_fld + '/' + **null**, load, **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a);  **this**.getClassPost();  }, err => {  err = **this**.\_user.\_decrypt(err.error.a);  **this**.getClassPost();  });  }  addVideoUrl(array) {  **for** (**let** i = **0**; i < array.length; i++) {  **if** (array[i].content\_fld.includes('youtube.com') || array[i].content\_fld.includes('google.com/file/d/')) {  array[i].videolink\_fld = **this**.getVideoEmbedURL(array[i].content\_fld);  }  **else** {  array[i].videolink\_fld = **null**;  }  }  **return** array;  }  *// Addedd by Melner*  getVideoEmbedURL(content: **string**): any {  **let** replacePattern1 = /(\b(https?|ftp):\/\/[-A-Z0-9+&@#\/%?=~\_|!:,.;]\*[-A-Z0-9+&@#\/%=~\_|])/gim;  **let** replacedText = content.replace(replacePattern1, '<\*>$1<\*>');  **let** content\_arr: **string**[] = replacedText.split('<\*>');  **let** a: **any** = content\_arr.filter((x) => {  **if** (x.includes('http')) {  **return** x  };  });  **for** (**let** i = **0**; i < a.length; i++) {  **if** (a[i].includes('youtube.com')) {  a[i] = a[i].replace('watch?v=', 'embed/');;  }  **else** {  a[i] = a[i].replace('view?usp=sharing', 'preview');  }  a[i] = **this**.sanitizer.bypassSecurityTrustResourceUrl(a[i]);  }  **return** a;  }  editQuiz(item, i) {  item.istoedit\_fld = **1**;  **this**.\_user.setQuiz(item);  **this**.\_router.navigate(['main/creator/quiz']);  }  createAssignmentDialog() {  **let** data = {  "type": 'assignment',  "topicdata": **this**.topicObj,  };  **const** dialogRef = **this**.dialog.open(CrudActivityComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  disableClose: **true**,  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('50%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result != **undefined**) {  **this**.activities.unshift(result[**0**]);  }  });  }  createResourcesDialog() {  **let** data = {  "type": 'resources',  "topicdata": **this**.topicObj,  };  **const** dialogRef = **this**.dialog.open(CrudResourcesComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100%',  disableClose: **true**,  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('90%', 'auto');  } **else** {  dialogRef.updateSize('50%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result != **undefined**) {  **this**.resourcesObj.unshift(result);  }  });  }  viewPoll(pollresponse) {  **const** dialogRef = **this**.dialog.open(ViewpollComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100%',  disableClose: **false**,  data: **pollresponse**,  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('50%', 'auto');  }  });  }  imgpreview(img) {  **let** data = {  type: 'Previewimage',  img: **img**,  };  **this**.dialog.open(ViewDataOnlyComponent, {  panelClass: 'dialogpadding',  data: **data**,  });  }  } |

**ACTIVITIES**

|  |
| --- |
| **export** **class** ActivitiesComponent **implements** OnInit {  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(Breakpoints.XSmall);  activities = [];  resourcesObj = [];  topicObj = [];  getRecipients = **new** getRecipients();  **constructor**(  **private** breakpointObserver: **BreakpointObserver**,  **public** dialog: **MatDialog**, **private** route: **Router**,  **public** \_ds: **DataService**,  **public** \_user: **UserService**,  **public** \_router: **Router**,  ) { }  ngOnInit(): **void** {  **this**.initializeComponents();  }  *// ########## DISPLAY activities ################*  initializeComponents() {  **this**.getClassActivities();  **this**.getClassResources();  **this**.getClassTopics();  }  createQuiz() {  **this**.\_user.setQuiz(**null**);  **this**.\_router.navigate(['/main/creator/quiz']);  }  ViewResource(item) {  **let** data = {  "type": 'Resources',  data: **item**,  };  **const** dialogRef = **this**.dialog.open(ViewDataOnlyComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('45%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  });  }  viewActivity(item) {  window.sessionStorage.removeItem(btoa('activityinfo'));  **this**.\_user.setActivityInfo(item);  }  getClassActivities() {  **this**.\_ds.\_httpRequest('getcpost', { data: { classcode: **this.\_user.getClassroomInfo**().classcode\_fld, type: **null** } }, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.activities = dt.payload;  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  });  }  getClassResources() {  **let** load = {  data: { classcode: **this.\_user.getClassroomInfo**().classcode\_fld, }  };  **this**.\_ds.\_httpRequest('getreslist', load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.resourcesObj = dt.payload;  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  });  }  editQuiz(item, i) {  item.istoedit\_fld = **1**;  **this**.\_user.setQuiz(item);  **this**.\_router.navigate(['/main/creator/quiz']);  }  getClassTopics() {  **let** load = {  data: { classcode: **this.\_user.getClassroomInfo**().classcode\_fld, }  };  **this**.\_ds.\_httpRequest('gettopic', load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.topicObj = dt.payload;  **this**.\_user.setTopic(dt.payload);  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  **this**.topicObj = [];  });  }  deleteAct(type, item, i) {  **let** data = {  type: **type**,  data: **item**,  };  **const** dialogRef = **this**.dialog.open(DeleteRecordComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('20%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result) {  **this**.activities.splice(i, **1**);  }  });  }  deleteRes(type, item, i) {  **let** data = {  type: **type**,  data: **item**,  };  **const** dialogRef = **this**.dialog.open(DeleteRecordComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('20%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result) {  **this**.resourcesObj.splice(i, **1**);  }  });  }  *// ########## CREATE activities ################*  createAssignmentDialog() {  **let** data = {  "type": 'assignment',  "topicdata": **this**.topicObj,  };  **const** dialogRef = **this**.dialog.open(CrudActivityComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  disableClose: **true**,  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('95%', 'auto');  } **else** {  dialogRef.updateSize('50%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result != **undefined**) {  **this**.activities.unshift(result[**0**]);  }  });  }  createResourcesDialog() {  **let** data = {  "type": 'resources',  "topicdata": **this**.topicObj,  };  **const** dialogRef = **this**.dialog.open(CrudResourcesComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100%',  disableClose: **true**,  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('95%', 'auto');  } **else** {  dialogRef.updateSize('50%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result != **undefined**) {  **this**.resourcesObj.unshift(result);  }  });  }  createTopicDialog() {  **let** data = {  "type": 'topic',  };  **const** dialogRef = **this**.dialog.open(CrudTopicComponent, {  panelClass: 'dialogpadding',  width: '100%',  height: '100%',  disableClose: **true**,  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('100vw', 'auto');  } **else** {  dialogRef.updateSize('30%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result != **undefined**) {  **this**.topicObj.unshift(result[**0**]);  }  });  }  *// ########## EDIT activities ################*  editActivity(item, i) {  **let** data = {  "type": 'edit\_act',  "data": item,  "topicdata": **this**.topicObj,  };  **const** dialogRef = **this**.dialog.open(CrudActivityComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  disableClose: **true**,  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('50%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result != **undefined**) {  **this**.activities[i] = result;  }  });  }  editResources(item, i: **number**) {  **let** data = {  "type": 'edit\_resources',  "data": item,  "topicdata": **this**.topicObj,  };  **const** dialogRef = **this**.dialog.open(CrudResourcesComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100%',  disableClose: **true**,  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('90%', 'auto');  } **else** {  dialogRef.updateSize('50%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result != **undefined**) {  **this**.resourcesObj[i] = result;  }  });  }  editTopic(item, i) {  **let** data = {  "type": 'edit\_topic',  "data": item,  };  **const** dialogRef = **this**.dialog.open(CrudTopicComponent, {  panelClass: 'dialogpadding',  width: '100%',  height: '100%',  disableClose: **true**,  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('100vw', 'auto');  } **else** {  dialogRef.updateSize('30%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result != **undefined**) {  **this**.getObjects();  }  });  }  deleteTopic(data, i) {  **let** load = {  data: {  isdeleted\_fld: **1**  }, notif: {  id: **this.\_user.getUserID**(),  recipient: "",  message: **this.\_user.getUserFullname**() + " Deleted Topic in " + **this**.\_user.getClassroomInfo().subjdesc\_fld,  module: 'Activities',  }  };  **this**.\_ds.\_httpRequest('edittopic/' + data.topiccode\_fld + '/' + 'del', load, **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a);  **this**.getObjects();  }, (err) => {  err = **this**.\_user.\_decrypt(err.error.a);  **this**.getObjects();  });  }  getObjects() {  **this**.getClassActivities();  **this**.getClassResources();  **this**.getClassTopics();  }  *// ########## OTHER STUFF IN UI ################*  panelOpenState = **false**;  moveToTopic(topic, type, item) {  **let** recipients = **this**.getRecipients.memberMap(**this**.\_user.getClassMembers().student);  **let** load: **any**;  **switch** (type) {  **case** 'act':  load = {  data: {  "topiccode\_fld": topic == **0** ? 0 : **topic.topiccode\_fld**,  }, notif: {  id: **this.\_user.getUserID**(),  recipient: **recipients.join**(':'),  message: **this.\_user.getUserFullname**() + " Activity moved in Label" + **this**.\_user.getClassroomInfo().subjdesc\_fld,  module: 'Classroom',  }  };  **this**.\_ds.\_httpRequest('editpost/' + item + '/' + 'act', load, **1**).subscribe((res: **any**) => {  res = **this**.\_user.\_decrypt(res.a);  **this**.getObjects();  **this**.\_user.openSnackBar('The activities were successfully saved.', **null**, **3000**);  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  });  **break**;  **case** 'res':  load = {  data: {  "topiccode\_fld": topic == **0** ? 0 : **topic.topiccode\_fld**,  }, notif: {  id: **this.\_user.getUserID**(),  recipient: **recipients.join**(':'),  message: **this.\_user.getUserFullname**() + " Resources moved in Label" + **this**.\_user.getClassroomInfo().subjdesc\_fld,  module: 'Classroom',  }  };  **this**.\_ds.\_httpRequest('editres/' + item, load, **1**).subscribe((res: **any**) => {  res = **this**.\_user.\_decrypt(res.a);  **this**.getObjects();  **this**.\_user.openSnackBar('The activities were successfully saved.', **null**, **3000**);  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  });  **break**;  **default**:  **break**;  }  }  } |

**CLASSLIST**

|  |
| --- |
| **export** **class** MembersComponent **implements** OnInit {  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(  Breakpoints.XSmall  );  allStudents: **any**[] = [];  students: **any**[] = [];  studentact: **any**[] = [];  pageslice;  classmembersinfo: **any**[] = [];  isClassListLoaded = **false**;  personelData: **any**;  currentPersonel: **any**;  **constructor**(  **private** breakpointObserver: **BreakpointObserver**,  **public** dialog: **MatDialog**,  **public** \_ds: **DataService**,  **public** \_user: **UserService**,  **private** excelExport: **ExportExcelService**,  **private** \_common: **CommonService**,  ) { }  ngOnInit(): **void** {  **this**.initializeComponents();  **this**.currentPersonel = **this**.\_user.getClassroomInfo().email\_fld.split(',');  }  async initializeComponents() {  **this**.allStudents = **this**.\_user.getClassMembers().student;  **this**.students = await **this**.allStudents;  **this**.isClassListLoaded = **true**;  **this**.getPersonel();  }  ViewStudentInfo(info) {  **let** data = {  type: 'academic',  data: **info**,  };  **const** dialogRef = **this**.dialog.open(ViewDataOnlyComponent, {  panelClass: 'dialogpadding',  *// backdropClass: 'g-transparent-backdrop',*  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**,  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(  (result) => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('80%', 'auto');  }  }  );  dialogRef.afterClosed().subscribe((result) => {  responsiveDialogSubscription.unsubscribe();  });  }  filterStudents(ev) {  **let** searchItem: **string** = ev.target.value;  **this**.students = **this**.allStudents.filter((x) => {  **return** (  x.lname\_fld.toUpperCase().includes(searchItem.toUpperCase()) ||  x.fname\_fld.toUpperCase().includes(searchItem.toUpperCase())  );  });  }  ViewStudentActivities(info) {  **let** data = {  type: 'academic',  data: **info**,  };  **const** dialogRef = **this**.dialog.open(StudentworksComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**,  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(  (result) => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('80%', 'auto');  }  }  );  dialogRef.afterClosed().subscribe((result) => {  responsiveDialogSubscription.unsubscribe();  });  }  exportToExcel() {  **let** dataForExcel: **any** = [];  **let** tmp: **any** = [];  **this**.students.forEach((row: **any**, i) => {  tmp.push({  No: **i** + **1**,  StudentID: **row.studnum\_fld**,  StudentName:  **row.lname\_fld** + ', ' + row.fname\_fld + ', ' + row.mname\_fld,  Program: **row.program\_fld**,  Block: **row.block\_fld**,  ContactNumber: **row.contactnum\_fld**,  EmailAdd: **row.email\_fld**,  LearningMode: **row.learningtype\_fld**,  });  });  tmp.forEach((element) => {  dataForExcel.push(Object.values(element));  });  **let** activeClass: **any** = **this**.\_user.getClassroomInfo();  **let** tmpSubjCode: **string** = activeClass.subjcode\_fld;  tmpSubjCode = tmpSubjCode.replace(/[^a-zA-Z0-9]/g, '');  **let** tmpSubjDesc: **string** = activeClass.subjdesc\_fld;  tmpSubjDesc = tmpSubjDesc.replace(/[^a-zA-Z0-9]/g, '');  **let** tmpSchedule: **string** =  activeClass.day\_fld +  ' ' +  activeClass.starttime\_fld +  '-' +  activeClass.endtime\_fld;  **let** reportData = {  title: 'Student List\_' + activeClass.classcode\_fld + '-' + tmpSubjCode,  title1: **activeClass.classcode\_fld** + ' - ' + tmpSubjCode,  title2: **tmpSubjDesc**,  title3: **tmpSchedule**,  data: **dataForExcel**,  headers: **Object.keys**(tmp[**0**]),  };  **this**.excelExport.exportExcel(reportData);  }  getPersonel() {  **this**.\_user.setLoading(**true**);  **this**.\_ds.\_httpRequest('getallpersonel', **null**, **1**).subscribe(async (dt: **any**) => {  dt = await **this**.\_user.\_decrypt(dt.a);  **this**.personelData = dt.payload;  **this**.\_user.setLoading(**false**);  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  **this**.\_user.setLoading(**false**);  });  }  addTeacher() {  **let** data = {  type: 'addteacher',  data: **this.personelData**,  };  **const** dialogRef = **this**.dialog.open(StudentworksComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**,  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(  (result) => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('80%', 'auto');  }  }  );  dialogRef.afterClosed().subscribe((result) => {  **if** (!result) **return**;  **this**.currentPersonel = result.split(',');  responsiveDialogSubscription.unsubscribe();  });  }  removeTeacher(i) {  **let** data = {  type: 'confirm-delete',  data: **this.personelData**,  };  **const** dialogRef = **this**.dialog.open(StudentworksComponent, {  panelClass: 'dialogpadding',  maxWidth: '30%',  maxHeight: '100vh',  data: **data**,  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(  (result) => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('80%', 'auto');  }  }  );  dialogRef.afterClosed().subscribe((result) => {  **if** (!result) **return**;  **this**.\_user.setLoading(**true**);  **this**.currentPersonel.splice(i, **1**);  **let** load = {  data: {  email\_fld: **this.currentPersonel.toString**(),  }  }  **this**.\_common.commonSubscribe('updateteacher/' + **this**.\_user.getClassroomInfo().classcode\_fld, load, **1**)  .then(() => {  **this**.\_user.setLoading(**false**);  responsiveDialogSubscription.unsubscribe();  **this**.\_user.updateClassroomInformation({  email\_fld: **load.data.email\_fld**  });  **this**.\_user.openSnackBar('Removed Successfully', 'Close', **2000**);  })  .**catch**(() => {  **this**.\_user.setLoading(**false**);  })  });  }  } |

**SUMMARY REPORT**

|  |
| --- |
| **export** **class** SummaryComponent **implements** OnInit {  summaryObj: **any** = [];  studentsObj: **any** = [];  submissionObj: **any** = []  actObj: **any** = []  **constructor**(  **private** breakpointObserver: **BreakpointObserver**,  **public** dialog: **MatDialog**, **private** route: **Router**,  **public** \_ds: **DataService**,  **public** \_user: **UserService**,  **private** excelExport: **ExportExcelService**  ) { }  ngOnInit(): **void** {  **this**.getClassList();  }  getClassList() {  **let** load = {  data: {  classcode: **this.\_user.getClassroomInfo**().classcode\_fld,  acadyear: **this.\_user.getSettings**().acadyear\_fld,  semester: **this.\_user.getSettings**().sem\_fld  }  }  **this**.\_ds.\_httpRequest('getmembers', load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.\_user.setClassMembers(dt.payload);  **this**.studentsObj = dt.payload.student;  **this**.getSummaryScores();  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  });  }  getSummaryScores() {  **let** load = {  data: { classcode: **this.\_user.getClassroomInfo**().classcode\_fld, }  }  **this**.\_ds.\_httpRequest('getsummary', load, **1**).subscribe( async (dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  *// console.log(dt)*  **this**.submissionObj = await dt.payload.sub  **this**.summaryObj = await **this**.getSummaryObject(dt.payload.act, dt.payload.sub)  console.log(**this**.summaryObj)  *// this.studentsObj.map((stud) => {*  *// console.log(stud)*  *// console.log(stud.studnum\_fld);*  *// console.log(stud.fname\_fld);*  *// console.log(stud.lname\_fld);*  *// });*  *// this.summaryObj.map(s => {*  *// console.log(s);*  *// console.log(s.title);*  *// // console.log(s.studnum\_fld);*  *// // console.log(s.fname\_fld);*  *// // console.log(s.lname\_fld);*  *// })*  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  });  }  getScore(s: **string**, list: **any**): any {  **let** a: **any**;  *// console.log(list.students[s].score)*  **try** {  a = list.students[s].isscored\_fld != **0** ? list.students[s].score\_fld:'';  } **catch** {  **return** 'N/A';  }  **return** a;  }  exportScores() {  **let** dataForExcel: **any** = [];  **let** tmp: **any** = [];  **this**.summaryObj.forEach(element => {  });  }  exportToExcel() {  **let** dataForExcel: **any** = [];  **let** tmp: **any** = [];  **this**.studentsObj.forEach((row: **any**, i) => {  tmp[i] = { No: **i** + **1**, StudentID: **row.studnum\_fld**, StudentName: **row.lname\_fld** + ', ' + row.fname\_fld }  **this**.summaryObj.map((element, j) => {  tmp[i][element.title] = **this**.getScore(row.studnum\_fld, element)  })  })  tmp.forEach(element => {  dataForExcel.push(Object.values(element));  });  **let** activeClass: **any** = **this**.\_user.getClassroomInfo();  **let** tmpSubjCode: **string** = activeClass.subjcode\_fld;  tmpSubjCode = tmpSubjCode.replace(/[^a-zA-Z0-9]/g, '');  **let** tmpSubjDesc: **string** = activeClass.subjdesc\_fld;  tmpSubjDesc = tmpSubjDesc.replace(/[^a-zA-Z0-9]/g, '');  **let** tmpSchedule: **string** = activeClass.day\_fld + ' ' + activeClass.starttime\_fld + '-' + activeClass.endtime\_fld;  **let** reportData = {  title: 'Summary of Scores\_' + activeClass.classcode\_fld + '-' + tmpSubjCode,  title1: **activeClass.classcode\_fld** + ' - ' + tmpSubjCode,  title2: **tmpSubjDesc**,  title3: **tmpSchedule**,  data: **dataForExcel**,  headers: **Object.keys**(tmp[**0**])  }  **this**.excelExport.exportScores(reportData);  }  getSummaryObject(act, sub){  **let** sumObj: **any** = []  act.forEach(element => {  sumObj.push({  actcode: **element.actcode\_fld**,  title: **element.title\_fld**,  totalscore: **element.totalscore\_fld**,  students: **this.getStudentScore**(element.recipient\_fld.split('.'), element.actcode\_fld)  })  })  **return** sumObj  }  getStudentScore(recipients, actcode){  **let** studobj: **any** = [];  **let** nosub = {  score\_fld: **0**,  issubmitted\_fld: **0**,  isscored\_fld: **0**  }  recipients.forEach(element => {  **if**(**this**.submissionObj == **null**){  studobj[element] = nosub  }  **else**{  studobj[element] = **this**.submissionObj  .find( (item: **any** ) => {  **return** item.actcode\_fld == actcode && item.studnum\_fld == element  }) || nosub  }  });  **return** studobj  }  } |

**VIEW ACTIVITIES AND QUIZ**

|  |
| --- |
| **export** **class** ViewActivitiesComponent **implements** OnInit {  **constructor**(  **public** \_user: **UserService**,  **public** \_ds: **DataService**,  **public** uploadservice: **UploadingService**,  **private** \_bottomSheetRef: **MatBottomSheet**,  **private** \_router: **Router**,  *// private dialogReg: MatDialogRef<ViewActivitiesComponent>,*  **private** dialog: **MatDialog**,  **private** breakpointObserver: **BreakpointObserver**,  **private** exportQuizExcel: **ExportExcelService**  ) { }  isOpen = **this**.\_user.getActivityInfo().withfile\_fld == **3** ? **true** : **false**;  isDisplayResult = **this**.\_user.getActivityInfo().quizoptions\_fld == **1** ? **true** : **false**;  comments: **string** = ''  commentObject = [];  displayedColumns: **string**[] = ['studnum\_fld', 'lname\_fld', 'submittedfiles', 'datetime\_fld', 'score\_fld', 'action'];  dataSource: **any**;  classList: **any** = [];  getRecipients = **new** getRecipients();  studworks = []  TabIndex = **0**  quizObject: **any** = []  selectedIndex: **any**;  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(Breakpoints.XSmall);  quizdisplayedColumns = ['number', 'question', 'type', 'answerkey', 'score'];  quizdataSource: **any**;  @ViewChild(MatSort) sort: **MatSort**;  @ViewChild(MatPaginator) paginator: **MatPaginator**;  ngOnInit(): **void** {  **this**.getActComments()  **this**.getClassList();  **this**.getStudentWorks();  **if** (**this**.\_user.getActivityInfo().type\_fld == **1**)  **this**.showQuiz();  }  setTabIndex(e) {  localStorage.setItem('index', e.index)  }  getIndex() {  **this**.TabIndex = +localStorage.getItem('index')  }  getFileExtension(filename) {  **if** (filename == 'pdf') {  **return** '#EA462E';  } **else** **if** (filename == 'docx') {  **return** '#2D5292';  }  **else** **if** (filename == 'ppt') {  **return** '#CA4223';  }  **else** **if** (filename == 'zip') {  **return** '#B23333';  }  **else** **if** (filename == 'txt') {  **return** '#546A7A';  } **else** {  **return** '#222';  }  }  async getClassList() {  **let** load = {  data: {  classcode: **this.\_user.getClassroomInfo**().classcode\_fld,  acadyear: **this.\_user.getSettings**().acadyear\_fld,  semester: **this.\_user.getSettings**().sem\_fld  }  }  **return** **new** Promise((resolve, rejects) => {  **this**.\_ds.\_httpRequest('getmembers', load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  resolve(dt.payload.student);  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  **return** **new** Promise(() => {  rejects(er);  })  });  })  }  getActComments() {  **this**.\_ds.\_httpRequest('getccomment', { data: { acode: **this.\_user.getActivityInfo**().actcode\_fld } }, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a)  **this**.commentObject = dt.payload  }, er => {  er = **this**.\_user.\_decrypt(er.error.a)  })  }  addComments() {  **if** (**this**.comments == '') **return**;  **this**.\_user.setLoading(**true**);  **let** load = {  data: {  content\_fld: **this.comments**,  authorid\_fld: **this.\_user.getUserID**(),  classcode\_fld: **this.\_user.getClassroomInfo**().classcode\_fld,  actioncode\_fld: **this.\_user.getActivityInfo**().actcode\_fld,  }, notif: {  id: **this.\_user.getUserID**(),  recipient: **this.\_user.getUserID**(),  message: **this.\_user.getUserFullname**() + " Added Comment Activity in " + **this**.\_user.getClassroomInfo().subjdesc\_fld,  module: 'Classroom',  }  }  **this**.\_ds.\_httpRequest('addcomment', load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a)  **this**.comments = ''  **this**.commentObject.unshift(dt.payload[**0**]);  **this**.\_user.setLoading(**false**);  }, er => {  er = **this**.\_user.\_decrypt(er.error.a)  **this**.\_user.setLoading(**false**);  })  }  splitLink(data) {  }  async getStudentWorks() {  **this**.\_user.setLoading(**true**);  **let** load = {  data: {  classcode: **this.\_user.getClassroomInfo**().classcode\_fld,  actcode: **this.\_user.getActivityInfo**().actcode\_fld,  type: **null**,  acadyear: **this.\_user.getSettings**().acadyear\_fld,  semester: **this.\_user.getSettings**().sem\_fld  }  }  **this**.\_ds.\_httpRequest('getstudworks', load, **1**).subscribe(async (dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a)  **this**.studworks = await **this**.dataTableHelper(dt.payload);  **this**.dataSource = **new** MatTableDataSource(**this**.studworks);  **this**.dataSource.sort = **this**.sort;  **this**.dataSource.paginator = **this**.paginator;  **this**.\_user.setLoading(**false**);  **this**.getIndex()  }, async er => {  er = **this**.\_user.\_decrypt(er.error.a);  **this**.dataSource = er.payload;  **this**.\_user.setLoading(**false**);  });  }  async dataTableHelper(data: **any**) {  data.forEach(element => {  **if** (element.type\_fld == **0**) {  element.dir\_fld = **this**.uploadservice.splitFilestring(element.dir\_fld);  }  **if** (element.type\_fld == **2**) {  element.dir\_fld = element.dir\_fld.split(',');  }  });  **return** data;  }  applyFilter(event: **Event**) {  **const** filterValue = (event.target **as** HTMLInputElement).value;  **this**.dataSource.filter = filterValue.trim().toLowerCase();  }  viewSubmission(data) {  *// console.log(!(data['dir\_fld'].includes("https")));*  *// if((data['dir\_fld'].includes("https"))){*  *// return*  *// }*  **if** (data.issubmitted\_fld == **1** || data.issubmitted\_fld == **2**) {  **this**.\_user.setViewSubmission(data);  **this**.\_router.navigate([`main/classes/${data.classcode\_fld}/${data.actcode\_fld}/${data.studnum\_fld}`]);  }  }  *// async setStudentWorks(submissions: any) {*  *// let assigned: any = this.\_user.getActivityInfo().recipient\_fld.split('.');*  *// assigned.splice(assigned.indexOf('0'), 1);*  *// let students: any = await this.getClassList();*  *// console.log(assigned);*  *// console.log(students);*  *// console.log(submissions);*  *// assigned.map((id) => {*  *// students.map((s) => {*  *// if (id == s.studnum\_fld) {*  *// submissions.map((s) => {*  *// console.log(s);*  *// })*  *// }*  *// // if (id != s.studnum\_fld)*  *// })*  *// })*  *// // submissions.map((data) => {*  *// // console.log(data.studnum\_fld);*  *// // })*  *// // assigned.map(a => {*  *// // console.log(a);*  *// // })*  *// // let submissions: any = data;*  *// // let students: any = await this.getClassList();*  *// // let n = assigned.indexOf('0');*  *// // if (n > -1) {*  *// // assigned.splice(n, 1);*  *// // }*  *// // if (submissions != null && assigned.length != submissions.length) {*  *// // assigned.forEach(rec => {*  *// // const found = submissions.some(el => el.studnum\_fld === rec)*  *// // if (!found) {*  *// // students.forEach(element => {*  *// // if (element.studnum\_fld == rec) {*  *// // submissions.push({ fname\_fld: element.fname\_fld, lname\_fld: element.lname\_fld, studnum\_fld: rec, dir\_fld: null, issubmitted\_fld: 0 })*  *// // }*  *// // });*  *// // }*  *// // });*  *// // } else {*  *// // submissions = []*  *// // students.forEach(element => {*  *// // const found = assigned.some(el => el == element.studnum\_fld);*  *// // if (found) {*  *// // submissions.push({ fname\_fld: element.fname\_fld, lname\_fld: element.lname\_fld, studnum\_fld: element.studnum\_fld, dir\_fld: null, issubmitted\_fld: 0 })*  *// // }*  *// // });*  *// // return new Promise((res) => {*  *// // submissions = submissions*  *// // res(submissions);*  *// // })*  *// // }*  *// // return new Promise((res) => {*  *// // res(data);*  *// // });*  *// }*  editGrade(i) {  **this**.dataSource.\_renderData.\_value[i].isscored\_fld = !**this**.dataSource.\_renderData.\_value[i].isscored\_fld;  }  saveGrade(i) {  **let** load = {  data: {  score\_fld: **this.dataSource.\_renderData.\_value**[i].score\_fld,  isscored\_fld: **1**  }, notif: **null**  }  **let** api = `editsubmit/${**this**.dataSource.\_renderData.\_value[i].submitcode\_fld}/${**this**.dataSource.\_renderData.\_value[i].studnum\_fld}/${**this**.dataSource.\_renderData.\_value[i].actcode\_fld}`  **this**.\_ds.\_httpRequest(api, load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.dataSource.\_renderData.\_value[i].isscored\_fld = !**this**.dataSource.\_renderData.\_value[i].isscored\_fld;  *// this.setStudentWorks(dt.payload);*  }, err => {  err = **this**.\_user.\_decrypt(err.error.a);  });  }  *// openViewFiles(item) {*  *// this.\_bottomSheetRef.open(FilesbottomsheetComponent, {*  *// data: item,*  *// });*  *// }*  editActivity(state) {  **let** load = {  data: { withfile\_fld: **state** },  notif: **null**  }  **this**.\_ds.\_httpRequest('editpost/' + **this**.\_user.getActivityInfo().actcode\_fld + '/' + 'act', load, **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a);  **this**.\_user.setActivityInfo(res.payload[**0**]);  **this**.\_user.openSnackBar(state == **2** ? 'The quiz is now over.' : 'The quiz is about to begin.', 'Okay', **5000**);  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  });  }  showQuiz() {  **this**.\_ds.\_httpRequest('getquiz', { filepath: **this.\_user.getActivityInfo**().filedir\_fld }, **1**).subscribe(async (res: **any**) => {  res = **this**.\_user.\_decrypt(res.a)  **let** answerkey = await res.payload.answerkey.content;  **let** quizcontent = await res.payload.quiz.content;  **this**.quizObject = await res.payload;  **let** initial\_array = [];  quizcontent.map((q) => {  answerkey.map((a) => {  **if** (q.questionid == a.questionid) {  **let** m1 = { ...q, ...a }  initial\_array.push(m1);  }  });  });  **this**.quizdataSource = **new** MatTableDataSource(initial\_array);  })  }  checkIfQuestionIsString(question, isImage) {  **if** (**typeof** question === 'string' && isImage === **0**) **return** question  **if** (**typeof** question !== 'object' && isImage === **1**) **return** **null**  **return** isImage === **1** ? question[**1**] : question[**0**]  }  onSaveSettings() {  **let** load = {  data: {  withfile\_fld: **this.isOpen** ? '3' : '2',  quizoptions\_fld: **this.isDisplayResult** ? '1' : **0**  },  notif: **null**  }  **this**.\_ds.\_httpRequest('editpost/' + **this**.\_user.getActivityInfo().actcode\_fld + '/' + 'act', load, **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a);  **this**.\_user.setActivityInfo(res.payload[**0**]);  **this**.\_user.openSnackBar('The quiz settings were successfully saved.', **null**, **2000**);  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  });  }  deleteRecords(type, item, i) {  **let** data = {  type: **type**,  data: **item**,  };  **const** dialogRef = **this**.dialog.open(DeleteRecordComponent, {  panelClass: '',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('20%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result) {  **this**.commentObject.splice(i, **1**);  }  });  }  splitChoice(data) {  **return** data.split('//\*//');  }  downloadQuiz() {  **const** blob = **new** Blob([JSON.stringify(**this**.quizObject)], { type: 'text/json' })  **const** url = URL.createObjectURL(blob);  **const** a = document.createElement("a");  a.href = url;  a.download = `${**this**.\_user.getActivityInfo().title\_fld}.json`;  a.click();  }  deleteSubmit(item, i) {  **let** load = {  data: {  isdeleted\_fld: **1**  },  notif: **null**  }  **let** api = `editsubmit/${item.submitcode\_fld}/${item.studnum\_fld}/${item.actcode\_fld}`  **this**.\_ds.\_httpRequest(api, load, **1**).subscribe(async (res: **any**) => {  res = **this**.\_user.\_decrypt(res.a)  await **this**.studworks.splice(i, **1**)  **this**.dataSource = await **new** MatTableDataSource(**this**.studworks)  *// this.getStudentWorks();*  }, er => {  er = **this**.\_user.\_decrypt(er.error.a)  })  }  sortData(sort: **any**) {  **const** data = **this**.dataSource.data  **if** (!sort.active || sort.direction === '') {  **this**.dataSource.data = data;  **return**;  }  **this**.dataSource.data = data.sort((a, b) => {  **const** isAsc = sort.direction === 'asc';  **switch** (sort.active) {  **case** 'lname\_fld': **return** compare(a.lname\_fld, b.lname\_fld, isAsc);  **case** 'studnum\_fld': **return** compare(a.studnum\_fld, b.studnum\_fld, isAsc);  **case** 'datetime\_fld': **return** compare(a.datetime\_fld, b.datetime\_fld, isAsc);  **case** 'score\_fld': **return** compare(a.score\_fld, b.score\_fld, isAsc);  *// case 'carbs': return compare(a.carbs, b.carbs, isAsc);*  *// case 'protein': return compare(a.protein, b.protein, isAsc);*  **default**: **return** **0**;  }  });  **function** compare(a: **number** | **string**, b: **number** | **string**, isAsc: **boolean**) {  **return** (a < b ? -1 : **1**) \* (isAsc ? **1** : -**1**);  }  }  exportQuiz() {  **let** excelQuizData = []  **for** (**let** i = **0**; i < **this**.quizdataSource.data.length; i++) {  excelQuizData[i] = [  **this**.quizdataSource.data[i].questionid,  **this**.quizdataSource.data[i].question,  **this**.quizdataSource.data[i].score,  **this**.quizdataSource.data[i].questiontype,  **this**.quizdataSource.data[i].choices,  **this**.quizdataSource.data[i].answer  ]  }  **this**.exportQuizExcel.exportQuiz(**this**.\_user.getActivityInfo().title\_fld, excelQuizData)  }  } |

**VIEW STUDENT SUBMISSIONS**

|  |
| --- |
| **export** **class** ViewStudentSumbmissionComponent **implements** OnInit, OnDestroy {  dataPreview: **any** = **null**;  downloadData: **any**;  fileType: **number**;  documentValidator = ['pptx', 'docx', 'xlsx', 'doc', 'csv']; *// Document Validators*  imageValidator = ['jpg', 'png', 'gif', 'jpeg']; *// Image Validators*  videoValidator = ['mp4']; *// Image Validators*  SaveGradesForm: **FormGroup**;  submissionData: **any** = [];  quizQuestion: **any**;  quizKeyAnswer: **any**;  quizStudentAnswer: **any**;  dataSource: **any**;  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(Breakpoints.XSmall);  displayedColumns = ['number', 'question', 'type', 'answerkey', 'studentanswer', 'mark', 'score', 'action'];  **constructor**(  **public** \_user: **UserService**,  **public** \_uploadservice: **UploadingService**,  **private** sanitizer: **DomSanitizer**,  **public** dialog: **MatDialog**,  **private** breakpointObserver: **BreakpointObserver**,  **private** fb: **FormBuilder**,  **public** \_ds: **DataService**,  **private** http: **HttpClient**,  **private** \_common: **CommonService**,  ) { }  ngOnInit(): **void** {  **this**.initializeFunctions();  }  ngOnDestroy(): **void** {  window.sessionStorage.removeItem(btoa('submission'));  }  initializeFunctions() {  **this**.submissionData = **this**.\_user.getViewSubmission();  **this**.SaveGradesForm = **this**.fb.group({  grades\_input: [**this**.submissionData.score\_fld, [Validators.required, Validators.max(+**this**.\_user.getActivityInfo().totalscore\_fld), Validators.min(**0**)]],  });  **switch** (**this**.submissionData.type\_fld.toString()) {  **case** '0':  **this**.viewAttachments(**this**.submissionData.dir\_fld[**0**]); *// Viewing for Attachments*  **break**;  **case** '1':  **this**.getQuizResult(); *// Viewing for Quiz Results*  **break**;  **case** '2':  **this**.viewLink(**this**.submissionData.dir\_fld[**0**]); *// Viewing for Links*  **break**;  **default**:  **break**;  }  }  viewAttachments(item: **any**) {  *// Validator for DOCUMENTS*  **if** (**this**.documentValidator.includes(item.extension[**0**])) {  **this**.fileType = **0**;  **this**.dataPreview = `https:*//view.officeapps.live.com/op/embed.aspx?src=${item.link}&embedded=true`;*  **this**.dataPreview = **this**.sanitizer.bypassSecurityTrustResourceUrl(**this**.dataPreview);  }  *// Validator for IMAGES*  **if** (**this**.imageValidator.includes(item.extension[**0**])) {  **this**.fileType = **1**;  **this**.dataPreview = item.link;  }  *// Validator for VIDEO*  **if** (**this**.videoValidator.includes(item.extension[**0**])) {  **this**.fileType = **2**  **this**.dataPreview = item.link;  }  *// Validator for NOT SUPPORTED FILES*  **if** (!**this**.imageValidator.includes(item.extension[**0**]) && !**this**.documentValidator.includes(item.extension[**0**])) {  **this**.dataPreview = **undefined**;  **this**.downloadData = {  link: **item.path**,  name: **item.name**  };  }  }  viewLink(item: **any**) {  **try** {  **this**.fileType = **3**;  **this**.dataPreview = **this**.sanitizer.bypassSecurityTrustResourceUrl(item);  } **catch** (error) {  error  }  }  getErrorMessage() {  **if** (**this**.SaveGradesForm.controls.grades\_input.hasError('required')) **return** 'You must enter a value';  **if** (**this**.SaveGradesForm.controls.grades\_input.hasError('max')) **return** `Score should be less than or equal to ${**this**.SaveGradesForm.controls.grades\_input.errors.max.max}`  }  saveGrade() {  **if** (**this**.SaveGradesForm.invalid) **return**;  **this**.\_user.setLoading(**true**);  **let** load = {  data: {  score\_fld: **this.SaveGradesForm.value.grades\_input**,  isscored\_fld: **1**  }, notif: **null**,  }  **this**.\_ds.\_httpRequest(`editsubmit/${**this**.submissionData.submitcode\_fld}/${**this**.submissionData.studnum\_fld}/${**this**.submissionData.actcode\_fld}`, load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.\_user.setLoading(**false**);  **this**.submissionData.score\_fld = dt.payload[**0**].score\_fld;  **this**.submissionData.isscored\_fld = dt.payload[**0**].isscored\_fld;  **this**.\_user.setViewSubmission(**this**.submissionData);  **this**.\_user.openSnackBar('Grades were successfully saved.', **null**, **3000**);  }, err => {  err = **this**.\_user.\_decrypt(err.error.a);  **this**.\_user.setLoading(**false**);  });  }  editGrade() {  **this**.submissionData.isscored\_fld = **0**;  }  getQuizResult() {  **this**.\_common.commonSubscribe('getquizresult', { filepath: **this.\_user.getViewSubmission**()['dir\_fld'], quizpath: **this.\_user.getActivityInfo**().filedir\_fld }, **1**)  .then((dt: **any**) => {  **this**.quizQuestion = dt.payload.quiz.content;  **this**.quizKeyAnswer = dt.payload.key.content;  **this**.quizStudentAnswer = dt.payload.studentanswer;  })  .**finally**(() => {  **this**.quizSegregationObject();  })  .**catch**((error) => error);  }  quizSegregationObject() {  **let** initial\_array = [];  **let** final\_array = [];  **this**.quizQuestion.map((q) => {  **this**.quizKeyAnswer.map((a) => {  **if** (q.questionid === a.questionid) {  **let** m1 = { ...q, ...a }  initial\_array.push(m1);  }  });  });  initial\_array.map((k) => {  **this**.quizStudentAnswer.map((sa) => {  **if** (k.questionid === sa.questionid) {  k.answerkey = k.answer;  **let** m2 = { ...k, ...sa }  final\_array.push(m2)  }  })  });  **this**.dataSource = **new** MatTableDataSource(final\_array);  }  checkIfQuestionIfString(question, isImage) {  **if** (**typeof** question === 'string' && isImage === **0**) **return** question  **if** (**typeof** question !== 'object' && isImage === **1**) **return** **null**  **return** isImage === **1** ? question[**1**] : question[**0**]  }  splitChoice(data) {  **return** data.split('//\*//');  }  changeMarkStatus(i: **number**) {  **this**.dataSource.\_renderData.\_value[i].mark = !**this**.dataSource.\_renderData.\_value[i].mark;  **this**.quizStudentAnswer.map((data: **any**) => {  **if** (**this**.dataSource.\_renderData.\_value[i].questionid == data.questionid) {  **this**.dataSource.\_renderData.\_value[i].mark == **true** ? data.mark = parseInt('1') : data.mark = parseInt('0');  **this**.dataSource.\_renderData.\_value[i].mark == **true** ? data.points = parseInt(**this**.dataSource.\_renderData.\_value[i].score) : data.points = parseInt('0');  **this**.dataSource.\_renderData.\_value[i].mark == **true** ? **this**.dataSource.\_renderData.\_value[i].points = parseInt(**this**.dataSource.\_renderData.\_value[i].score) : **this**.dataSource.\_renderData.\_value[i].points = parseInt('0');  **this**.dataSource.\_renderData.\_value[i].mark == **true** ? **this**.dataSource.\_renderData.\_value[i].mark = parseInt('1') : **this**.dataSource.\_renderData.\_value[i].mark = parseInt('0');  }  });  **this**.getNewScoreValue();  }  inputEssayScore(i: **number**) {  **let** data = {  "type": 'score\_essay',  "data": **this**.dataSource.\_renderData.\_value[i],  };  **const** dialogRef = **this**.dialog.open(ViewDataOnlyComponent, {  width: '100%',  height: '100%',  disableClose: **true**,  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('100vw', 'auto');  } **else** {  dialogRef.updateSize('100vw', 'auto');  }  });  dialogRef.afterClosed().subscribe((result) => {  responsiveDialogSubscription.unsubscribe();  **if** (result == **undefined**) **return**;  **this**.dataSource.\_renderData.\_value[i].points = result;  **this**.quizStudentAnswer.map((data: **any**) => {  **if** (**this**.dataSource.\_renderData.\_value[i].questionid == data.questionid) {  result > **0** ? data.mark = parseInt('1') : data.mark = parseInt('0');  result > **0** ? **this**.dataSource.\_renderData.\_value[i].mark = parseInt('1') : **this**.dataSource.\_renderData.\_value[i].mark = parseInt('0');  data.points = parseInt(result);  }  });  **this**.getNewScoreValue();  });  }  getNewScoreValue() {  **const** arr = [];  **this**.quizStudentAnswer.map(data => {  arr.push(parseInt(data.points));  });  **const** reducer = (accumulator, curr) => accumulator + curr;  **this**.submissionData.score\_fld = arr.reduce(reducer);  **this**.saveStudentQuiz();  }  saveStudentQuiz() {  *// this.\_user.setLoading(true);*  **let** load = {  filepath: **this.submissionData.dir\_fld**,  answer: {  result: **this.dataSource.data**,  totalscore: **this.submissionData.score\_fld**,  quiztotal: **this.\_user.getActivityInfo**().totalscore\_fld,  }  }  **this**.\_common.commonSubscribe('editanswer', load, **1**)  .then(() => {  **let** load = {  data: {  score\_fld: **this.submissionData.score\_fld**,  }, notif: **null**  }  **this**.\_common.commonSubscribe(`editsubmit/${**this**.submissionData.submitcode\_fld}/${**this**.submissionData.studnum\_fld}/${**this**.submissionData.actcode\_fld}`, load, **1**)  .then((dt: **any**) => {  **this**.\_user.setLoading(**false**);  **this**.submissionData.score\_fld = dt.payload[**0**].score\_fld;  **this**.\_user.setViewSubmission(**this**.submissionData);  })  })  }  } |

**FORUMS**

|  |
| --- |
| **export** **class** ForumsComponent **implements** OnInit {  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(Breakpoints.XSmall);  ForumsObject: **any** = [];  allforum: **any** = [];  filter: **string** = '';  showLoader: **boolean**;  data: **number** = **0**;  **constructor**(  **public** dialog: **MatDialog**,  **private** \_router: **Router**,  **private** breakpointObserver: **BreakpointObserver**,  **public** \_user: **UserService**,  **public** \_ds: **DataService**) { }  ngOnInit(): **void** {  **this**.intializeComponents();  }  intializeComponents() {  **this**.getForumTopic();  }  getForumTopic() {  **this**.showLoader = **true**;  **this**.\_ds.\_httpRequest('getforumtopic', { dept: **this.\_user.getUserDept**() }, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.ForumsObject = dt.payload;  **this**.allforum = dt.payload;  **this**.showLoader = **false**;  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  **this**.allforum = er.payload;  **this**.showLoader = **false**;  });  }  addForum() {  **let** data = {  "type": 'mainforum',  };  **const** dialogRef = **this**.dialog.open(CRUDForumsComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('40%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result != **undefined**) {  **this**.ForumsObject.push(result[**0**]);  }  });  }  gotoSelectedForum(item) {  window.sessionStorage.removeItem(btoa('subforumtopic'));  **this**.\_user.setSelectedForum(([item]));  **this**.\_router.navigate(['main/forums', item.forumcode\_fld]);  }  *// filterlist() {*  *// let results: any = []*  *// if (this.filter != '') {*  *// this.ForumsObject.forEach(index => {*  *// if (index.forumtitle\_fld.toLowerCase().includes(this.filter.toLowerCase()) || index.forumdesc\_fld.toLowerCase().includes(this.filter.toLowerCase())) {*  *// results.push(index);*  *// }*  *// })*  *// this.ForumsObject = results;*  *// } else {*  *// this.getForumTopic();*  *// }*  *// }*  deleteRecords(type, item, i) {  **let** data = {  type: **type**,  data: **item**,  };  **const** dialogRef = **this**.dialog.open(DeleteRecordComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('20%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result) {  **this**.ForumsObject.splice(i, **1**);  }  });  }  editTopic(e, i) {  **let** data = {  type: 'editmainforum',  data: **e**,  };  **const** dialogRef = **this**.dialog.open(CRUDForumsComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('50%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  **if** (result != **undefined**) {  **this**.ForumsObject[i] = result[**0**];  }  responsiveDialogSubscription.unsubscribe();  });  }  filterlist() {  **let** results: **any** = []  **if** (**this**.filter != '') {  **this**.allforum.forEach(index => {  **if** (index.forumtitle\_fld.toLowerCase().includes(**this**.filter.toLowerCase()) || index.forumdesc\_fld.toLowerCase().includes(**this**.filter.toLowerCase())) {  results.push(index);  }  })  **this**.ForumsObject = results;  } **else** {  **this**.ForumsObject = **this**.allforum;  }  }  } |

**SUBFORUMS**

|  |
| --- |
| **export** **class** SubforumsComponent **implements** OnInit {  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(Breakpoints.XSmall);  SubForumsObject = [];  selectedForums: **any** = (**this**.\_user.getSelectedForum());  **constructor**(**private** breakpointObserver: **BreakpointObserver**,  **public** dialog: **MatDialog**,  **public** \_user: **UserService**,  **public** \_ds: **DataService**,  overlay: **Overlay**,  **public** \_router: **Router**) { }  ngOnInit(): **void** {  **this**.intializeComponents();  *// this.\_user.\_convert(this.selectedForums);*  }  *// ngOnDestroy(): void {*  *// window.sessionStorage.removeItem(btoa('forumtopic'))*  *// }*  intializeComponents() {  **this**.getSubForumTopic();  }  getSubForumTopic() {  **this**.\_ds.\_httpRequest('getsubforum', { data: { forumcode: **this.selectedForums**[**0**].forumcode\_fld } }, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.SubForumsObject = dt.payload;  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  });  }  showforumscomments(item) {  **this**.\_user.setSubSelectedForum([item]);  }  editSubforum(d, i) {  **let** data = {  type: 'editsubtopic',  data: **d**  };  **const** dialogRef = **this**.dialog.open(CrudSubforumsComponent, {  panelClass: 'dialogpadding',  width: '100%',  height: '100%',  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('40%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  **if** (result != **undefined**) {  **this**.SubForumsObject[i] = (result[**0**]);  }  responsiveDialogSubscription.unsubscribe();  });  }  AddSubforums() {  **let** data = {  type: 'Subtopic',  data: **this.\_user.getSelectedForum**()  };  **const** dialogRef = **this**.dialog.open(CrudSubforumsComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('40%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result != **undefined**) {  **this**.SubForumsObject.unshift(result[**0**]);  }    });  }  deleteRecords(type, item, i) {  **let** data = {  type: **type**,  data: **item**,  };  **const** dialogRef = **this**.dialog.open(DeleteRecordComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('20%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result) {  **this**.SubForumsObject.splice(i, **1**);  }  });  }  } |

**FORUMS COMMENTS**

|  |
| --- |
| **export** **class** ForumscontentComponent **implements** OnInit {  subforumContent = **this**.\_user.getSubSelectedForum();  content: **string** = '';  forumContent = [];  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(Breakpoints.XSmall);  SubForumsObject = [];  selectedForums: **any** = (**this**.\_user.getSelectedForum());  **constructor**(**private** breakpointObserver: **BreakpointObserver**,  **public** dialog: **MatDialog**,  **public** \_user: **UserService**,  **public** \_ds: **DataService**,  overlay: **Overlay**,  **public** \_router: **Router**,  **private** ar: **ActivatedRoute**, **private** \_location: **Location**,) {  *// this.\_user.\_convert(this.subforumContent);*  }  ngOnInit(): **void** {  **this**.getComments();  }  getComments() {  **this**.\_ds.\_httpRequest('getsubcontent', { data: { subcode: **this.subforumContent**[**0**].subcode\_fld } }, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.forumContent = dt.payload;  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  });  }  addReply() {  **let** subcode;  **this**.ar.paramMap.subscribe((p) => {  subcode = p.get('sid');  });  **let** load = {  data: {  subcode\_fld: **subcode**,  authorid\_fld: **this.\_user.getUserID**(),  content\_fld: **this.content**,  }, notif: **null**  }  **this**.\_ds.\_httpRequest('addreply', load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.forumContent.unshift(dt.payload[**0**]);  **this**.content = '';  *// this.closeDialog(dt.payload);*  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  });  }  deleteRecords(type, item, i) {  **let** data = {  type: **type**,  data: **item**,  };  **const** dialogRef = **this**.dialog.open(DeleteRecordComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('20%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result) {  **this**.forumContent.splice(i, **1**);  }  });  }  backToSubforums(): **void** {  **this**.\_location.back();  }  } |

**NEWS AND ANNOUNCEMENTS**

|  |
| --- |
| **export** **class** NewsComponent **implements** OnInit {  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(Breakpoints.XSmall);  showLoader: **boolean** = **true**;  announcementObj: **any** = [];  latestObj: **any** = [];  allAnnounce: **any** = [];  filter: **string** = '';  imageRandom = "?v=" + Math.random();  **constructor**(**private** \_router: **Router**,  **public** \_user: **UserService**, **public** \_ds: **DataService**,  **private** breakpointObserver: **BreakpointObserver**,  **public** dialog: **MatDialog**) { }  ngOnInit(): **void** {  **this**.initializeComponents();  **this**.imageRandom = **this**.imageRandom.toString();  }  initializeComponents() {  **this**.getAnnouncements();  }  getAnnouncements() {  **this**.\_ds.\_httpRequest('getannounce', { dept: **this.\_user.getUserDept**() }, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.announcementObj = dt.payload;  **this**.allAnnounce = dt.payload;  **this**.latestObj.push(dt.payload[**0**]);  **this**.announcementObj = **this**.announcementObj.splice(**0**, **1**);  **this**.showLoader = **false**;  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  **this**.allAnnounce = er.payload;  **this**.showLoader = **false**  });  }  gotoInnerContent(data) {  **this**.\_router.navigate(['./main/news', data.announcecode\_fld]);  **this**.\_user.setNews(([data]));  }  filterlist() {  **let** results: **any** = []  **if** (**this**.filter != '') {  **this**.allAnnounce.forEach(index => {  **if** (index.title\_fld.toLowerCase().includes(**this**.filter.toLowerCase()) || index.content\_fld.toLowerCase().includes(**this**.filter.toLowerCase())) {  results.push(index);  }  })  **this**.announcementObj = results;  } **else** {  **this**.announcementObj = **this**.allAnnounce;  }  }  } |

**NEWS CONTENTS**

|  |
| --- |
| **export** **class** NewscontentComponent **implements** OnInit, OnDestroy {  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(Breakpoints.XSmall);  showModal: **boolean** = **true**;  newsContent: **any** = **this**.\_user.getNews();  announcementObj = [];  imageRandom = "?v=" + Math.random();  **constructor**(  **public** dialog: **MatDialog**,  **private** breakpointObserver: **BreakpointObserver**,  **private** \_router: **Router**,  **public** \_user: **UserService**,  **public** \_ds: **DataService**  ) { }  ngOnInit(): **void** {  **this**.imageRandom = **this**.imageRandom.toString();  **this**.getAnnouncements();  **this**.newsContent = **this**.newsContent;  }  ngOnDestroy(): **void** {  window.sessionStorage.removeItem(btoa('news'));  }  Imagepreview(item) {  **let** data = {  "type": 'Previewimage',  "img": item.imgdir\_fld  };  **const** dialogRef = **this**.dialog.open(ViewDataOnlyComponent, {  panelClass: 'csss-modalbox',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('auto', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  });  }  opencomment() {  **let** data = {  type: 'newscomment',  data: **this.newsContent**[**0**]  };  **const** dialogRef = **this**.dialog.open(CrudNewsCommentsComponent, {  width: '100%',  height: '100%',  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('100%', 'auto');  } **else** {  dialogRef.updateSize('50%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **this**.newsContent[**0**].commentcount = result;  });  }  getAnnouncements() {  **this**.\_ds.\_httpRequest('getannounce', { dept: **this.\_user.getUserDept**() }, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.announcementObj = dt.payload;  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  });  }  show() {  **this**.showModal = **true**; *// Show-Hide Modal Check*  }  *//Bootstrap Modal Close event*  hide() {  **this**.showModal = **false**;  }  viewAnn(a) {  **this**.newsContent = [{  announcecode\_fld: **a.announcecode\_fld**,  commentcount: **a.commentcount**,  content\_fld: **a.content\_fld**,  datetime\_fld: **a.datetime\_fld**,  dtedit\_fld: **a.dtedit\_fld**,  imgdir\_fld: **a.imgdir\_fld**,  recipientcode\_fld: **a.recipientcode\_fld**,  title\_fld: **a.title\_fld**,  withimg\_fld: **a.withimg\_fld**  }];  }  } |

**PROFILE**

|  |
| --- |
| **export** **class** ProfileComponent **implements** OnInit {  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(Breakpoints.XSmall);  profiledata: **any**;  random: **number** = Math.random();  **constructor**(  **private** breakpointObserver: **BreakpointObserver**,  **public** dialog: **MatDialog**,  **public** \_ds: **DataService**,  **public** \_user: **UserService**) { }  ngOnInit(): **void** {  **this**.getProfile();  }  getProfile() {  **this**.\_ds.\_httpRequest('facultyprofile/' + **this**.\_user.getUserID(), **null**, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.profiledata = dt.payload;  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  });  }  Changepassword(data) {  **const** dialogRef = **this**.dialog.open(ChangepasswordComponent, {  panelClass: 'dialogpadding',  maxHeight: '100vh',  maxWidth: '100vw',  data: { data , type: 'changepassword'},  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('100%', 'auto');  } **else** {  dialogRef.updateSize('40%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  });  }  updatePicture() {  **const** dialogConfig = **new** MatDialogConfig();  dialogConfig.autoFocus = **true**;  dialogConfig.disableClose = **true**;  dialogConfig.panelClass = 'dialogpadding';  dialogConfig.width = "40vw";  dialogConfig.data = {  load: **this.profiledata**  }  **const** updatePicture = **this**.dialog.open(UpdateProfilePictureComponent, dialogConfig);  updatePicture.afterClosed().subscribe((result: **any**) => {  **if** (result == **undefined**) **return**;  **this**.profiledata[**0**].profilepic\_fld = result;  })  }  UpdateProfile(data) {  **const** dialogRef = **this**.dialog.open(UpdateProfileComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**,  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('100%', 'auto');  } **else** {  dialogRef.updateSize('50%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result != **undefined**) {  **this**.profiledata = result;  }  });  }  } |

**ELIBRARY**

|  |
| --- |
| **export** **class** LibraryComponent **implements** OnInit {  isactive: **boolean** = **false**;  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(  Breakpoints.XSmall  );  isTiled: **boolean** = **true**;  bookTitle: **string**;  bookObject: **any** = [];  bookObjectIndividual;  searchList;  **constructor**(  **private** breakpointObserver: **BreakpointObserver**,  **public** dialog: **MatDialog**,  **public** \_ds: **DataService**,  **public** \_user: **UserService**  ) {}  ngOnInit(): **void** {  **this**.getBooks();  }  toggleview() {  **this**.isTiled = !**this**.isTiled;  }  Search() {  **let** book: **any**;  **let** author: **any**;  book = **this**.searchList.filter((res: **any**) =>  res.title\_fld  .toLocaleLowerCase()  .match(**this**.bookTitle.toLocaleLowerCase())  );  author = **this**.searchList.filter((res: **any**) =>  res.author\_fld  .toLocaleLowerCase()  .match(**this**.bookTitle.toLocaleLowerCase())  );  **if** (**this**.bookTitle == '') {  **this**.getBooks();  } **else** **if** (**this**.bookTitle != '') {  **if** (book == '') {  **this**.bookObject = author;  } **else** {  **this**.bookObject = book;  }  }  }  getBooks() {  **this**.\_ds.\_httpRequest('getbooks', '', **1**).subscribe(  (dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.bookObject = dt.payload;  *// this.\_user.setBook(this.bookObject);*  **this**.searchList = **this**.bookObject;  },  (er) => {  *// let err = this.\_user.\_decrypt(er.error.a);*  console.clear();  **this**.bookObject = **null**;  }  );  }  viewBook(link) {  **let** data = {  resourceString: `${**this**.\_ds.booksURL}${link}#toolbar=**1**&navpanes=**1**`,  reourcename: **name**,  };  **const** dialogRef = **this**.dialog.open(ResourcePreviewComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  disableClose: **true**,  data: **data**,  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(  (result) => {  **if** (result.matches) {  dialogRef.updateSize('100%', '100%');  } **else** {  dialogRef.updateSize('100%', '100%');  }  }  );  dialogRef.afterClosed().subscribe((result) => {  responsiveDialogSubscription.unsubscribe();  });  }  openbook() {  *// let data = {*  *// type: 'viewbooks',*  *// book: this.bookObjectIndividual*  *// };*  *// const dialogRef = this.dialog.open(ViewbookComponent, {*  *// width: '80%',*  *// height: '100%',*  *// maxWidth: '80%',*  *// maxHeight: '100vh',*  *// data: data*  *// });*  *// const responsiveDialogSubscription = this.isExtraSmall.subscribe(result => {*  *// if (result.matches) {*  *// dialogRef.updateSize('100%', '100vh');*  *// } else {*  *// dialogRef.updateSize('90%', '100vh');*  *// }*  *// });*  *// dialogRef.afterClosed().subscribe(result => {*  *// responsiveDialogSubscription.unsubscribe();*  *// // this.ngOnInit();*  *// // if (result != undefined) { this.resourceObject = result }*  *// });*  }  addBook() {  *// let data = {*  *// type: 'addbooks',*  *// book: ''*  *// };*  *// const dialogRef = this.dialog.open(AddbookComponent, {*  *// width: '10%',*  *// height: '100%',*  *// maxWidth: '100vw',*  *// maxHeight: '100vh',*  *// data: data,*  *// autoFocus: false*  *// });*  *// const responsiveDialogSubscription = this.isExtraSmall.subscribe(result => {*  *// if (result.matches) {*  *// dialogRef.updateSize('100%', '100vh');*  *// } else {*  *// dialogRef.updateSize('90%', '100vh');*  *// }*  *// });*  *// dialogRef.afterClosed().subscribe(result => {*  *// responsiveDialogSubscription.unsubscribe();*  *// this.ngOnInit();*  *// // if (result != undefined) { this.resourceObject = result }*  *// });*  }  } |

**MAINLAYOUT (sidebar menu)**

|  |
| --- |
| **export** **class** MainlayoutComponent **implements** OnInit {  isExtraSmall: **Observable**<BreakpointState> = **this**.\_breakpointObserver.observe(Breakpoints.XSmall);  random: **string** = '?' + Math.random();  notif: **any** = [];  @ViewChild('drawer') drawer: **any**;  isHandset$: **Observable**<**boolean**> = **this**.\_breakpointObserver.observe(Breakpoints.Handset)  .pipe(  map(result => result.matches),  shareReplay()  );  **constructor**(  *// private chat: ChatService,*  **private** \_breakpointObserver: **BreakpointObserver**,  **private** \_router: **Router**,  **public** \_user: **UserService**,  **public** route: **Router**,  **public** \_ds: **DataService**,  **public** dialog: **MatDialog**,  **private** themeService: **ThemeService**  ) { }  ngOnInit(): **void** {  **if** (!(btoa('facultyState') **in** sessionStorage && **this**.\_user.getLoginState())) {  **this**.route.navigate(['login']);  }  *// this.getNotif();*  *// this.setLightbulb();*  *// this.chat.openWebSocket();*  }  closeSideNav() {  **if** (**this**.drawer.\_mode == 'over') {  **this**.drawer.close();  }  }  *// getNotif() {*  *// this.\_ds.\_httpRequest('getnotif', null, 1).subscribe((dt: any) => {*  *// dt = this.\_user.\_decrypt(dt.a);*  *// this.notif = dt.payload;*  *// }, er => {*  *// er = this.\_user.\_decrypt(er.error.a);*  *// });*  *// }*  userLogout() {  **let** data = {  "type": 'logout',  };  **const** dialogRef = **this**.dialog.open(ViewDataOnlyComponent, {  maxWidth: '90vw',  maxHeight: '90vh',  disableClose: **true**,  data: **data**,  panelClass: 'dialogpadding'  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('100%', 'auto');  } **else** {  dialogRef.updateSize('20%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result = **undefined**) **return**  });  }  setLightbulb() {  **let** data = JSON.parse(localStorage.getItem("theme"));  **if** (data.name == 'dark') {  **this**.themeService.setDarkTheme();  } **else** {  **this**.themeService.setLightTheme();  }  }  toggleTheme() {  **if** (**this**.themeService.isDarkTheme()) {  **this**.themeService.setLightTheme();  } **else** {  **this**.themeService.setDarkTheme();  }  **this**.setLightbulb();  }  } |

**DIALOGS**

**ADD STUDENTS**

|  |
| --- |
| **export** **class** AddstudentComponent **implements** OnInit {  checked = **false**;  studentlist = [];  studentinfo = [];  **constructor**(**private** dialogReg: **MatDialogRef**<AddstudentComponent>,  **private** breakpointObserver: **BreakpointObserver**,  **public** dialog: **MatDialog**,  **public** \_ds: **DataService**,  **public** \_user: **UserService**,  @Inject(MAT\_DIALOG\_DATA) **public** data,  ) { }  ngOnInit(): **void** { }  search(e) {  e.preventDefault();  **let** f = e.target.elements;  **this**.\_ds.\_httpRequest('studentprofile/' + f.id.value, **null**, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.studentinfo = dt.payload;  **this**.\_user.openSnackBar(dt.status.message, **null**, **2000**);  f.id.value = **null**;  }, err => {  err = **this**.\_user.\_decrypt(err.error.a);  **this**.\_user.openSnackBar(err.status.message, **null**, **2000**);  });  }  addtolist(data, i) {  **if** (**this**.duplicateChecker(data)) {  **this**.\_user.openSnackBar("This student is already enrolled in this course.", **null**, **4000**);  } **else** **if** (**this**.duplicateChecker(data) == **false**) {  **this**.studentlist.push(data);  **this**.studentinfo.splice(i, **1**);  }  }  duplicateChecker(data) {  **var** matched = [];  **this**.data.map(e => {  **if** (e.studnum\_fld == data.studnum\_fld) {  matched.push(**true**);  } **else** {  matched.push(**false**);  }  });  **return** matched.includes(**true**) ? **true** : **false**;  }  remove(i) {  **this**.studentlist.splice(i, **1**);  }  removequery(i) {  **this**.studentinfo.splice(i, **1**);  }  addStudents() {  **let** load = {};  **this**.studentlist.map((e) => {  load = {  data: {  studnum\_fld: **e.studnum\_fld**,  classcode\_fld: **this.\_user.getClassroomInfo**().classcode\_fld,  subjcode\_fld: **this.\_user.getClassroomInfo**().subjcode\_fld,  block\_fld: **this.\_user.getClassroomInfo**().block\_fld,  ay\_fld: **this.\_user.getSettings**().acadyear\_fld,  sem\_fld: **this.\_user.getSettings**().sem\_fld,  }  }  **this**.\_ds.\_httpRequest('addstudent', load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.closeDialog(dt.payload.student)  });  })  }  closeDialog(data) {  **this**.dialogReg.close(data);  }  } |

**CHANGEPASSWORD**

|  |
| --- |
| **export** **class** ChangepasswordComponent **implements** OnInit {  errMsg: **any**;  hide1: **boolean**;  hide2: **boolean**;  hide3: **boolean**;  npass: **any** = '';  passwordStatus: **string**;  firstFormGroup: **FormGroup**;  secondFormGroup: **FormGroup**;  strongRegex = **new** RegExp("^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[0-9])(?=.\*[!@#\$%\^&\\*])(?=.{8,})");  mediumRegex = **new** RegExp("^(((?=.\*[a-z])(?=.\*[A-Z]))|((?=.\*[a-z])(?=.\*[0-9]))|((?=.\*[A-Z])(?=.\*[0-9])))(?=.{6,})");  **constructor**(  **private** dialogReg: **MatDialogRef**<ChangepasswordComponent>,  @Inject(MAT\_DIALOG\_DATA) **public** data: **any**,  **private** \_dialog: **MatDialog**,  **public** \_user: **UserService**, **private** \_ds: **DataService**,  **private** fb: **FormBuilder**,  **private** route: **Router**  ) { }  ngOnInit(): **void** {  **this**.intializeFunctions();  }  intializeFunctions() {  **this**.firstFormGroup = **this**.fb.group({  current: ['', Validators.required],  newpass: ['', [Validators.required, Validators.minLength(**8**)]],  confirmpass: ['', Validators.required],  });  }  focusFunction() {  **this**.errMsg = **null**;  }  getErrorMessage() {  **if** (**this**.firstFormGroup.controls.current.hasError('required')) **return** 'You must enter a value';  **if** (**this**.firstFormGroup.controls.newpass.hasError('required')) **return** 'You must enter a value';  **if** (**this**.firstFormGroup.controls.confirmpass.hasError('required')) **return** 'You must enter a value';  }  passwordTest(value) {  **let** passDisplay = document.getElementById('displaystyle');  **if** (passDisplay == **null**) **return**;  **if** (**this**.strongRegex.test(value)) {  **this**.passwordStatus = 'Strong';  passDisplay.style.color = 'green';  } **else** **if** (**this**.mediumRegex.test(value)) {  **this**.passwordStatus = 'Average';  passDisplay.style.color = 'orange';  } **else** {  **this**.passwordStatus = 'Weak';  passDisplay.style.color = 'red';  }  }  Changepassword() {  **if** (**this**.firstFormGroup.invalid) **return**;  **if** (**this**.firstFormGroup.value.newpass != **this**.firstFormGroup.value.confirmpass) {  **this**.\_user.setLoading(**false**);  **this**.firstFormGroup.controls['confirmpass'].setErrors({ 'incorrect': **true** });  **this**.\_user.openSnackBar('PASSWORD - Password did not match.', **null**, **3000**);  **return**;  }  **this**.\_user.setLoading(**true**);  **let** load = {  param1: **this.firstFormGroup.value.current**,  param2: **this.firstFormGroup.value.newpass**,  param3: **this.firstFormGroup.value.confirmpass**,  }  **this**.\_ds.\_httpRequest('changepass/' + **this**.\_user.getUserID(), load, **1**).subscribe(async (dt: **any**) => {  dt = await **this**.\_user.\_decrypt(dt.a);  **this**.\_ds.\_httpRequest('updateprofile/' + **this**.\_user.getUserID(), { data: { ispwordtochange: **0** } }, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.\_user.updateUserData({ reset: **0** });  **this**.closeDialog(**true**);  **this**.\_user.setLoading(**false**);  }, err => {  err = **this**.\_user.\_decrypt(err.error.a);  **this**.\_user.setLoading(**false**)  });  }, (error) => {  error = **this**.\_user.\_decrypt(error.error.a);  **this**.firstFormGroup.controls['current'].setErrors({ 'incorrect': **true** });  **this**.\_user.openSnackBar('PASSWORD - Incorrect current password.', **null**, **3000**);  })  }  closeDialog(data) {  **this**.dialogReg.close(data);  }  } |

**CREATE AND EDIT ACTIVITIY**

|  |
| --- |
| **export** **class** CrudActivityComponent **implements** OnInit, OnDestroy {  firstFormGroup: **FormGroup**;  secondFormGroup: **FormGroup**;  getRecipients = **new** getRecipients();  filePreviewAndUpload: **any** = [];  students: **any** = [];  editFiles = [];  **public** filepath;  filesToBeUpdated = [];  editedFile\_dir: **any** = [];  @ViewChild('allSelected') **private** allSelected: **MatOption**;  file = '';  minDate = **new** Date();  act\_due: **any**;  dueDatetime: **any**;  **constructor**(  **private** dialogReg: **MatDialogRef**<CrudActivityComponent>,  @Inject(MAT\_DIALOG\_DATA) **public** data,  **private** dialog: **MatDialog**,  **private** fb: **FormBuilder**,  **private** datepipe: **DatePipe**,  **public** \_user: **UserService**,  **private** \_ds: **DataService**,  **public** \_uploadservice: **UploadingService**,  **public** \_common: **CommonService**,  ) { }  ngOnInit(): **void** {  **this**.intializeFunctions();  **this**.\_uploadservice.files = [];  }  ngOnDestroy(): **void** { }  intializeFunctions() {  **this**.getClassList();  **this**.firstFormGroup = **this**.fb.group({  act\_title: ['', Validators.required],  act\_desc: [''],  act\_pts: ['', Validators.required],  act\_topic: [],  selected: ['', Validators.required],  });  **this**.secondFormGroup = **this**.fb.group({});  **if** (**this**.data.type == 'edit\_act') {  **this**.editActivity();  } **else** {  **this**.getDate();  }  }  getErrorMessage() {  **if** (**this**.firstFormGroup.controls.act\_title.hasError('required')) **return** 'You must enter a value';  **if** (**this**.firstFormGroup.controls.act\_pts.hasError('required')) **return** 'You must enter a value';  **if** (**this**.firstFormGroup.controls.selected.hasError('required')) **return** 'You must enter a value';  }  getClassList() {  **let** load = {  data: {  classcode: **this.\_user.getClassroomInfo**().classcode\_fld,  acadyear: **this.\_user.getSettings**().acadyear\_fld,  semester: **this.\_user.getSettings**().sem\_fld  }  }  **this**.\_ds.\_httpRequest('getmembers', load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.\_user.setClassMembers(dt.payload);  **this**.students = dt.payload.student;  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  });  }  editActivity() {  **this**.editedFile\_dir = (**this**.data.data.filedir\_fld);  **if** (**this**.editedFile\_dir !== '') {  **this**.editFiles = **this**.\_uploadservice.splitFilestring(**this**.editedFile\_dir);  }  **try** {  **let** datetime: **any** = **this**.datepipe.transform(**new** Date(**this**.data.data.deadline\_fld), 'yyyy-MM-dd;HH:mm:ss');  datetime = datetime.split(';');  **this**.act\_due = datetime[**0**];  **this**.dueDatetime = datetime[**1**];  } **catch** (error) {  **this**.act\_due = **new** Date();  **this**.dueDatetime = **this**.datepipe.transform(**new** Date(), '23:59:00');  }  **this**.firstFormGroup = **this**.fb.group({  act\_title: [**this**.data.data.title\_fld, Validators.required],  act\_desc: [**this**.data.data.desc\_fld],  act\_pts: [**this**.data.data.totalscore\_fld, Validators.required],  act\_topic: [**this**.data.data.topiccode\_fld, Validators.required],  selected: ['', Validators.required],  });  **this**.firstFormGroup.controls.selected.patchValue([...**this**.data.data.recipient\_fld.split('.')].map((item: **any**) => item));  }  getDate() {  **this**.act\_due = **new** Date();  **this**.act\_due = **this**.datepipe.transform(**new** Date().setDate(**new** Date().getDate() + **1**), 'yyyy-MM-dd');  **this**.dueDatetime = **new** Date();  **this**.dueDatetime = **this**.datepipe.transform(**new** Date(), '23:59:00');  }  tosslePerOne() {  **if** (**this**.allSelected.selected) { **this**.allSelected.deselect(); **return** **false** };  **if** (**this**.firstFormGroup.controls.selected.value.length == **this**.students.length)  **this**.allSelected.select();  }  toggleAllSelection() {  **if** (**this**.allSelected.selected)  **this**.firstFormGroup.controls.selected.patchValue([...**this**.students.map((item: **any**) => item.studnum\_fld), '0']);  **if** (!**this**.allSelected.selected)  **this**.firstFormGroup.controls.selected.patchValue([]);  }  async addActivity() {  **if** (**this**.firstFormGroup.invalid) **return**;  **this**.\_user.setLoading(**true**);  **this**.act\_due = **this**.datepipe.transform(**this**.act\_due, 'yyyy-MM-dd');  await **this**.\_uploadservice.uploadFiles(**this**.filePreviewAndUpload).then((files: **any**) => {  **let** recipients = **this**.getRecipients.memberMap(**this**.\_user.getClassMembers().student);  **let** load = {  data: {  "authorid\_fld": **this**.\_user.getUserID(),  "type\_fld": **0**,  "recipient\_fld": **this**.firstFormGroup.value.selected.join('.'),  "topiccode\_fld": !**this**.firstFormGroup.value.act\_topic ? 0 : **this.firstFormGroup.value.act\_topic**,  "title\_fld": **this**.firstFormGroup.value.act\_title,  "desc\_fld": **this**.firstFormGroup.value.act\_desc,  "totalscore\_fld": **this**.firstFormGroup.value.act\_pts,  "classcode\_fld": **this**.\_user.getClassroomInfo().classcode\_fld,  "withfile\_fld": files == **null** ? 0 : **1**,  "filedir\_fld": files == **null** ? '' : files,  "deadline\_fld": `${**this**.act\_due} ${**this**.dueDatetime}`,  }, notif: {  id: **this.\_user.getUserID**(),  recipient: **recipients.join**(':'),  message: **this.\_user.getUserFullname**() + " Added Activity in " + **this**.\_user.getClassroomInfo().subjdesc\_fld,  module: 'Classroom',  }  };  **this**.\_common.commonSubscribe('addactivity', load, **1**).then((data: **any**) => {  **this**.\_user.setLoading(**false**);  **this**.filePreviewAndUpload = [];  **this**.closeDialog(data.payload);  **this**.\_user.openSnackBar('The activity has been successfully added. 🤩🐱‍🏍', 'Close', **3000**);  }).**catch**(() => {  **this**.\_user.setLoading(**false**);  **this**.\_user.openSnackBar('An error occurred; please try again! 😭💔', 'Close', **3000**);  })  });  }  async saveActivity() {  **if** (**this**.firstFormGroup.invalid) **return**;  **this**.\_user.setLoading(**true**);  await **this**.\_uploadservice.uploadFiles(**this**.filePreviewAndUpload).then((files: **any**) => {  **let** arr: **any**;  **let** recipients = **this**.getRecipients.memberMap(**this**.\_user.getClassMembers().student);  **if** (files != **null**) {  arr = [...**this**.editedFile\_dir.split(':'), ...files.split(':')];  arr = arr.filter((f) => {  **return** f != "";  });  arr = arr.join(':');  } **else** **if** (files == **null**) {  arr = **this**.editedFile\_dir;  }  **let** load = {  data: {  "actcode\_fld": **this**.data.data.actcode\_fld,  "authorid\_fld": **this**.\_user.getUserID(),  "type\_fld": **0**,  "recipient\_fld": **this**.firstFormGroup.value.selected.join('.'),  "topiccode\_fld": **this**.firstFormGroup.value.act\_topic,  "title\_fld": **this**.firstFormGroup.value.act\_title,  "desc\_fld": **this**.firstFormGroup.value.act\_desc,  "totalscore\_fld": **this**.firstFormGroup.value.act\_pts,  "classcode\_fld": **this**.\_user.getClassroomInfo().classcode\_fld,  "withfile\_fld": arr == '' ? 0 : **1**,  "filedir\_fld": arr,  "deadline\_fld": `${**this**.datepipe.transform(**this**.act\_due, 'yyyy-MM-dd')} ${**this**.dueDatetime}`,  "datetime\_fld": **this**.data.data.datetime\_fld,  }, notif: {  id: **this.\_user.getUserID**(),  recipient: **recipients.join**(':'),  message: **this.\_user.getUserFullname**() + " Edited Activity in " + **this**.\_user.getClassroomInfo().subjdesc\_fld,  module: 'Classroom',  }  };  **this**.\_common.commonSubscribe('editpost/' + **this**.data.data.actcode\_fld + '/' + 'act', load, **1**)  .then(() => {  **this**.closeDialog(load.data);  **this**.\_user.setLoading(**false**);  });  });  }  async deleteFiles(i, path) {  **let** newFiles: **any** = [];  **this**.\_common.commonSubscribe('deletefile', { dir\_fld: **path** }, **1**)  .then(() => {  **this**.editFiles.splice(i, **1**);  **this**.editFiles.map((f) => {  newFiles.push(f.name + '?' + f.path);  })  })  .then(() => {  newFiles = newFiles.join(':');  **let** load = {  data: {  "withfile\_fld": newFiles == '' ? 0 : **1**,  "filedir\_fld": newFiles  }, notif: {  id: **this.\_user.getUserID**(),  recipient: **this.\_user.getUserID**(),  message: **this.\_user.getUserFullname**() + " Edited Activity in " + **this**.\_user.getClassroomInfo().subjdesc\_fld,  module: 'Classroom',  }  }  **this**.\_common.commonSubscribe('editpost/' + **this**.data.data.actcode\_fld + '/' + 'act', load, **1**)  .then((data: **any**) => {  **this**.editedFile\_dir = data.payload[**0**].filedir\_fld;  });  }).**catch**((error) => {  error  });  }  getFile(event: **any**) {  **this**.filePreviewAndUpload = **this**.\_uploadservice.getFilePreview(event);  }  removeFilePreviews(i) { **this**.filePreviewAndUpload.splice(i, **1**); }  closeDialog(data) {  **this**.dialogReg.close(data);  }  getFileExtension(filename) {  **if** (filename == 'pdf') {  **return** '#EA462E';  } **else** **if** (filename == 'docx') {  **return** '#2D5292';  }  **else** **if** (filename == 'ppt') {  **return** '#CA4223';  }  **else** **if** (filename == 'zip') {  **return** '#B23333';  }  **else** **if** (filename == 'txt') {  **return** '#546A7A';  } **else** {  **return** '#222';  }  }  } |

**CREATE QUIZ QUESTIONS**

|  |
| --- |
| **export** **class** CRUDClassAndActivitiesComponent **implements** OnInit, OnDestroy {  selecteQuizType: **number** = **0**;  questioniaresForms: **FormGroup**;  choicesObject: **any** = [];  selectedAnswer: **any**;  quizContent: **any** = [];  quizAnwerKey: **any** = [];  filePreviewAndUpload: **any** = [];  previewQuizImage: **String**;  **constructor**(  **public** \_ds: **DataService**,  **public** \_user: **UserService**,  **private** dialogReg: **MatDialogRef**<CRUDClassAndActivitiesComponent>, @Inject(MAT\_DIALOG\_DATA)  **public** data: **any**,  **private** fb: **FormBuilder**,  **private** \_uploadservice: **UploadingService**,  **private** \_common: **CommonService**  ) { }  ngOnInit(): **void** {  **this**.intializeFunctions();  }  ngOnDestroy(): **void** {  **this**.previewQuizImage = '';  }  intializeFunctions() {  **this**.questioniaresForms = **this**.fb.group({  question\_points: [**1**, Validators.required],  question\_description: ['', Validators.required],  question\_answerkey: ['', Validators.required],  question\_choice: [[]],  });  **if** (**this**.data.type) {  **this**.selecteQuizType = **this**.data.data.questions.questiontype  **this**.editQuestion();  }  }  editQuestion() {  *// Check if the question has images attached:*  **if** (**this**.data.data.questions.question[**1**] != **null**) {  **this**.previewQuizImage = **this**.data.data.questions.question[**1**];  }  *// Check if the choices is present in the question:*  **if** (!**this**.data.data.questions.choices) {  **delete** **this**.data.data.questions.choices;  } **else** {  **this**.choicesObject = **this**.data.data.questions.choices.split('//\*//');  }  *// Initialize Data binding for the FormBuilder Group:*  **this**.questioniaresForms = **this**.fb.group({  question\_points: [**this**.data.data.answer.score, Validators.required],  question\_description: [**this**.data.data.questions.question[**0**], Validators.required],  question\_answerkey: [**this**.data.data.answer.answer, Validators.required],  question\_choice: [],  });  **this**.questioniaresForms.controls.question\_answerkey.setValue(**this**.data.data.answer.answer);  }  addChoices() {  **if** (**this**.questioniaresForms.controls.question\_choice.invalid) **return**;  **if** (**this**.choicesObject.length > **4**) {  **this**.\_user.openSnackBar('The maximum number of choices has been reached.', 'Close', **3000**);  **return**  }  **this**.choicesObject.push(**this**.questioniaresForms.value.question\_choice);  **this**.questioniaresForms.controls.question\_choice.reset()  **if** (**this**.hasDuplicates(**this**.choicesObject)) {  **this**.choicesObject.splice(-**1**, **1**);  **this**.\_user.openSnackBar('Invalid! duplicate choices.', 'Close', **3000**);  }  }  removeChoices(i) {  **this**.choicesObject.splice(i, **1**);  **this**.\_user.openSnackBar('Choice removed.', 'Close', **3000**);  }  hasDuplicates(array) {  **return** (**new** Set(array)).size !== array.length;  }  addQuizQuestion() {  **this**.questioniaresForms.controls.question\_choice.setValue(**this**.choicesObject.join('//\*//'));  **this**.quizContent.push({  questionid: **this.data.questionid**,  questiontype: **this.selecteQuizType**,  choices: **this.questioniaresForms.value.question\_choice**,  question: **this.questioniaresForms.value.question\_description** + "[!m4g3$]" + **this**.previewQuizImage,  });  **this**.quizAnwerKey.push({  questionid: **this.data.questionid**,  answer: **this.questioniaresForms.value.question\_answerkey**,  score: **this.questioniaresForms.value.question\_points**,  });  **this**.questioniaresForms.reset({ question\_points: **1** });  **this**.choicesObject = [];  **let** load = {  question: **this.quizContent**,  answer: **this.quizAnwerKey**,  type: **this.data.type**,  index: **null**  }  **if** (**this**.data.type == **1**) load.index = **this**.data.data.index;  **if** (**this**.selecteQuizType == **3**) **this**.questioniaresForms.controls.question\_answerkey.setValue('');  **this**.dialogReg.close(load);  }  getFile(event: **any**) {  **this**.\_uploadservice.files = [];  **this**.filePreviewAndUpload = **this**.\_uploadservice.getFilePreview(event);  **this**.\_uploadservice.uploadFiles(**this**.filePreviewAndUpload)  .then((files: **any**) => {  **this**.previewQuizImage = files.split('?')[**1**];  });  }  removeQuizImage() {  **this**.data.data.questions.question[**1**] = **null**;  **this**.\_common.commonSubscribe('deletefile', { dir\_fld: **this.previewQuizImage** }, **1**)  .then((files: **any**) => {  **this**.previewQuizImage = ''  **this**.\_user.openSnackBar('Quiz image removed.', **null**, **2000**);  });  }  } |

**ADD AND EDIT COMMENTS IN CLASSFEED**

|  |
| --- |
| **export** **class** CrudCommentsComponent **implements** OnInit, OnDestroy {  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(Breakpoints.XSmall);  Content: **any** = [];  commentObject = [];  replyObj: **any** = [];  getRecipients = **new** getRecipients();  **constructor**(**private** dialogReg: **MatDialogRef**<CrudCommentsComponent>, @Inject(MAT\_DIALOG\_DATA) **public** data: **object**,  **private** dialog: **MatDialog**, **private** fb: **FormBuilder**,  **public** \_user: **UserService**,  **public** \_ds: **DataService**, **private** breakpointObserver: **BreakpointObserver**,  ) {  **this**.Content = data;  }  ngOnInit(): **void** {  **this**.intializeComponents();  }  ngOnDestroy(): **void** {  *// + this.commentObject[0].reply.length*  **this**.closeDialog(**this**.commentObject.length);  }  intializeComponents() {  **this**.getPostComment();  setTimeout(() => {  (<HTMLInputElement>document.getElementById('user-input')).focus();  });  }  state(i, item) {  (<HTMLInputElement>document.getElementById('content\_fld' + i)).value = item.content\_fld;  **let** comments = document.getElementById('comments' + i);  **let** img = document.getElementById('img-container' + i);  **let** editbox = document.getElementById('edit-box' + i);  comments.style.display === "none" ? comments.style.display = "block" : comments.style.display = "none";  img.style.display === "none" ? img.style.display = "block" : img.style.display = "none";  editbox.style.display === "none" ? editbox.style.display = "flex" : editbox.style.display = "none";  (<HTMLInputElement>document.getElementById('content\_fld' + i)).focus();  }  reply\_state(i, j, item) {  (<HTMLInputElement>document.getElementById(i + 'reply\_content\_fld' + j)).value = item.content\_fld;  **let** reditbox = document.getElementById(i + 'redit-box' + j);  *// let img = document.getElementById(i + 'img-container' + j);*  **let** editbox = document.getElementById(i + 'replybox' + j);  editbox.style.display === "none" ? editbox.style.display = "block" : editbox.style.display = "none";  *// img.style.display === "none" ? img.style.display = "block" : img.style.display = "none";*  reditbox.style.display === "none" ? reditbox.style.display = "flex" : reditbox.style.display = "none";  (<HTMLInputElement>document.getElementById(i + 'reply\_content\_fld' + j)).focus();  }  closeEdit(i) {  **let** editbox = document.getElementById('edit-box' + i);  **let** comments = document.getElementById('comments' + i);  **let** img = document.getElementById('img-container' + i);  editbox.style.display === "flex" ? editbox.style.display = "none" : editbox.style.display = "none";  comments.style.display = 'block';  img.style.display = 'block';  }  closeReplyEdit(i, j) {  **let** replybox = document.getElementById(i + 'replybox' + j);  **let** redit = document.getElementById(i + 'redit-box' + j);  redit.style.display === "flex" ? redit.style.display = "none" : redit.style.display = "none";  replybox.style.display = 'flex';  }  saveComment(e, i, item) {  e.preventDefault();  **let** f = e.target.elements;  **this**.\_ds.\_httpRequest('editcomm/' + item.commentcode\_fld, { data: { content\_fld: **f.content\_fld.value** } }, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.commentObject[i].content\_fld = dt.payload[**0**].content\_fld;  **this**.state(i, dt.payload[**0**].content\_fld);  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  })  }  saveReply(e, i, j, item) {  e.preventDefault();  **let** f = e.target.elements;  **this**.\_ds.\_httpRequest('editcomm/' + item.commentcode\_fld, { data: { content\_fld: **f.content\_fld.value** } }, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.commentObject[i].reply[j] = dt.payload[**0**];  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  })  }  getPostComment() {  **let** load = {  data: { acode: **this.Content.postcode**, }  }  **this**.\_ds.\_httpRequest('getccomment', load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.commentObject = dt.payload;  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  })  }  deleteRecords(type, item, i) {  **let** data = {  type: **type**,  data: **item**,  };  **const** dialogRef = **this**.dialog.open(DeleteRecordComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('20%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result) {  **this**.commentObject.splice(i, **1**);  }  });  }  deleteReply(type, item, i, j) {  **let** data = {  type: **type**,  data: **item**,  };  **const** dialogRef = **this**.dialog.open(DeleteRecordComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('20%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result) {  **this**.commentObject[i].reply.splice(j, **1**);  }  });  }  addComments(e) {  **let** recipients = **this**.getRecipients.memberMap(**this**.\_user.getClassMembers().student);  e.preventDefault();  **let** f = e.target.elements;  **let** load = {  data: {  content\_fld: **f.content\_fld.value**,  authorid\_fld: **this.\_user.getUserID**(),  classcode\_fld: **this.\_user.getClassroomInfo**().classcode\_fld,  actioncode\_fld: **this.Content.postcode**,  }, notif: {  id: **this.\_user.getUserID**(),  recipient: **recipients.join**(':'),  message: **this.\_user.getUserFullname**() + " Commented in " + **this**.\_user.getClassroomInfo().subjdesc\_fld,  module: 'Classroom',  }  }  **this**.\_ds.\_httpRequest('addcomment', load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  f.content\_fld.value = **null**;  **this**.commentObject.unshift(dt.payload[**0**]);  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  })  }  addReply(item, i) {  **let** x = document.getElementById('replydiv' + i);  x.style.display === "none" ? x.style.display = "block" : x.style.display = "none";  **let** j = document.getElementById('replybtn' + i);  j.style.display === "none" ? j.style.display = "block" : j.style.display = "none";  (<HTMLInputElement>document.getElementById("input-content" + i)).focus();  }  cancelReply(i) {  **let** y = document.getElementById('replydiv' + i);  **let** x = document.getElementById('replybtn' + i);  y.style.display = 'none';  x.style.display = "flex";  }  sendReply(e, item, i) {  **let** recipients = **this**.getRecipients.memberMap(**this**.\_user.getClassMembers().student);  e.preventDefault();  **let** f = e.target.elements;  **let** load = {  data: {  content\_fld: **f.content\_fld.value**,  authorid\_fld: **this.\_user.getUserID**(),  classcode\_fld: **this.\_user.getClassroomInfo**().classcode\_fld,  actioncode\_fld: **item.commentcode\_fld**,  }, notif: {  id: **this.\_user.getUserID**(),  recipient: **recipients.join**(':'),  message: **this.\_user.getUserFullname**() + " Added reply in " + **this**.\_user.getClassroomInfo().subjdesc\_fld,  module: 'Classroom',  }  }  **this**.\_ds.\_httpRequest('addcomment', load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **if** (**this**.commentObject[i].reply == **null**) {  **this**.commentObject[i].reply = [];  **this**.commentObject[i].reply.push(dt.payload[**0**]);  } **else** {  **this**.commentObject[i].reply.push(dt.payload[**0**]);  }  f.content\_fld.value = **null**;  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  });  }  closeDialog(data) {  **this**.dialogReg.close(data);  }  } |

**ADD FORUMS DISCUSSIONS**

|  |
| --- |
| **export** **class** CRUDForumsComponent **implements** OnInit {  Content: **any** = [];  ForumsFormGroup: **FormGroup**;  **constructor**(  **private** dialogReg: **MatDialogRef**<CRUDForumsComponent>,  @Inject(MAT\_DIALOG\_DATA) **public** data,  **private** fb: **FormBuilder**,  **public** \_user: **UserService**,  **public** \_ds: **DataService**,  **private** \_common: **CommonService**) {  }  ngOnInit(): **void** {  **this**.ForumsFormGroup = **this**.fb.group(  {  topic\_title: ['', Validators.required],  topic\_desc: [''],  }  );  **if** (**this**.data.type == 'editmainforum') {  **this**.ForumsFormGroup = **this**.fb.group(  {  topic\_title: [**this**.data.data.forumtitle\_fld, Validators.required],  topic\_desc: [**this**.data.data.forumdesc\_fld],  }  );  }  }  addForumTopic() {  **let** load = {  data: {  authorid\_fld: **this.\_user.getUserID**(),  forumtitle\_fld: **this.ForumsFormGroup.value.topic\_title**,  forumdesc\_fld: **this.ForumsFormGroup.value.topic\_desc**,  isapproved\_fld: **1**,  }, notif: **null**  }  **if** (**this**.ForumsFormGroup.invalid) **return**;  **this**.\_common.commonSubscribe('addforumtopic', load, **1**)  .then((dt: **any**) => {  **this**.closeDialog(dt.payload);  });  }  saveForumTopic() {  **let** load = {  data: {  forumtitle\_fld: **this.ForumsFormGroup.value.topic\_title**,  forumdesc\_fld: **this.ForumsFormGroup.value.topic\_desc**,  },  notif: **null**,  }  **if** (**this**.ForumsFormGroup.valid) {  **this**.\_ds.\_httpRequest('editforum/' + **this**.data.data.forumcode\_fld + '/', load, **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a);  **this**.\_user.openSnackBar(res.status.message, **null**, **2000**);  **this**.closeDialog(res.payload);  }, err => {  err = **this**.\_user.\_decrypt(err.a);  });  }  }  closeDialog(data) {  **this**.dialogReg.close(data);  }  } |

**CREATE AND EDIT SUBFORUMS**

|  |
| --- |
| **export** **class** CrudSubforumsComponent **implements** OnInit {  ForumsFormGroup: **FormGroup**;  **constructor**(  **private** dialogReg: **MatDialogRef**<CrudSubforumsComponent>,  @Inject(MAT\_DIALOG\_DATA) **public** data,  **private** fb: **FormBuilder**,  **public** \_user: **UserService**,  **public** \_ds: **DataService**) {  }  ngOnInit(): **void** {  **this**.ForumsFormGroup = **this**.fb.group({  topic\_title: ['', Validators.required],  topic\_desc: [''],  });  **if** (**this**.data.type == "editsubtopic") {  **this**.ForumsFormGroup = **this**.fb.group({  topic\_title: [**this**.data.data.subtitle\_fld, Validators.required],  topic\_desc: [**this**.data.data.subdesc\_fld],  });  }  }  addSubForum() {  **let** load = {  data: {  forumcode\_fld: **this.data.data**[**0**].forumcode\_fld,  authorid\_fld: **this.\_user.getUserID**(),  subtitle\_fld: **this.ForumsFormGroup.value.topic\_title**,  subdesc\_fld: **this.ForumsFormGroup.value.topic\_desc**,  isapproved\_fld: **1**,  },  notif: **null**  }  **if** (**this**.ForumsFormGroup.valid) {  **this**.\_ds.\_httpRequest('addsubforum', load, **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a);  **this**.\_user.openSnackBar(res.status.message, **null**, **2000**);  **this**.closeDialog(res.payload);  }, err => {  err = **this**.\_user.\_decrypt(err.error.a);  });  }  }  saveSubForum() {  **let** load = {  data: {  forumcode\_fld: **this.data.data.forumcode\_fld**,  authorid\_fld: **this.\_user.getUserID**(),  subtitle\_fld: **this.ForumsFormGroup.value.topic\_title**,  subdesc\_fld: **this.ForumsFormGroup.value.topic\_desc**,  isapproved\_fld: **1**,  },  notif: **null**  }  **if** (**this**.ForumsFormGroup.valid) {  **this**.\_ds.\_httpRequest('editdisc/' + **this**.data.data.subcode\_fld + '/', load, **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a);  **this**.closeDialog(res.payload);  }, err => {  err = **this**.\_user.\_decrypt(err.error.a);  });  }  }  closeDialog(data) {  **this**.dialogReg.close(data);  }  } |

**FORUMS CONTENT COMMENTS**

|  |
| --- |
| **export** **class** CrudForumsCommentsComponent **implements** OnInit {  **constructor**(**public** \_user: **UserService**, **public** \_ds: **DataService**,  **private** dialogReg: **MatDialogRef**<CrudForumsCommentsComponent>,  @Inject(MAT\_DIALOG\_DATA) **public** data,  **private** dialog: **MatDialog**,) { }  forumContent = [];  content: **string** = '';  ngOnInit(): **void** {  **this**.initializeComponents();  }  initializeComponents() {  **this**.getComments();  }  getComments() {  **this**.\_ds.\_httpRequest('getsubcontent', { subcode: **this.data.data.subcode\_fld** }, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.forumContent = dt.payload;  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  });  }  addReply() {  **let** load = {  data: {  subcode\_fld: **this.data.data**[**0**].subcode\_fld,  authorid\_fld: **this.\_user.getUserID**(),  content\_fld: **this.content**,  },  notif: **null**  }  **this**.\_ds.\_httpRequest('addreply', load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.forumContent.unshift(dt.payload[**0**]);  **this**.content = '';  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  });  }  } |

**ADD TOPIC FORUMS**

|  |
| --- |
| **export** **class** CrudTopicComponent **implements** OnInit {  Topicformgroup: **FormGroup**;  getRecipients = **new** getRecipients();  **constructor**(  **private** dialogReg: **MatDialogRef**<CrudTopicComponent>,  @Inject(MAT\_DIALOG\_DATA) **public** data: **any**,  **public** \_user: **UserService**,  **private** \_ds: **DataService**,  **private** fb: **FormBuilder**,  ) { }  ngOnInit(): **void** {  **if** (**this**.data.type == 'edit\_topic') {  **this**.Topicformgroup = **this**.fb.group({  topicname: [**this**.data.data.topicname\_fld, Validators.required],  });  } **else** {  **this**.Topicformgroup = **this**.fb.group({  topicname: ['', Validators.required],  });  }  }  getErrorMessage() {  **if** (**this**.Topicformgroup.controls.topicname.hasError('required')) **return** 'You must enter a value';  }  addTopic() {  **if** (**this**.Topicformgroup.invalid) **return**;  **this**.\_user.setLoading(**true**)  **let** recipients = **this**.getRecipients.memberMap(**this**.\_user.getClassMembers().student);  **let** load = {  data: {  topicname\_fld: **this.Topicformgroup.value.topicname**,  classcode\_fld: **this.\_user.getClassroomInfo**().classcode\_fld,  }, notif: {  id: **this.\_user.getUserID**(),  recipient: **recipients.join**(':'),  message: **this.\_user.getUserFullname**() + " Added Topic in " + **this**.\_user.getClassroomInfo().subjdesc\_fld,  module: 'Classroom',  }  }  **this**.\_ds.\_httpRequest('addtopic', load, **1**).subscribe((dt: **any**) => {  **this**.\_user.setLoading(**false**)  dt = **this**.\_user.\_decrypt(dt.a);  **this**.closeDialog(dt.payload);  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  });  }  saveTopic() {  **this**.\_user.setLoading(**true**);  **let** recipients = **this**.getRecipients.memberMap(**this**.\_user.getClassMembers().student);  **let** load = {  data: {  topicname\_fld: **this.Topicformgroup.value.topicname**,  }, notif: {  id: **this.\_user.getUserID**(),  recipient: **recipients.join**(':'),  message: **this.\_user.getUserFullname**() + " Deleted Topic in " + **this**.\_user.getClassroomInfo().subjdesc\_fld,  module: 'Classroom',  }  }  **this**.\_ds.\_httpRequest('edittopic/' + **this**.data.data.topiccode\_fld + '/' + 'edit', load, **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a);  **this**.\_user.setLoading(**false**);  **this**.\_user.openSnackBar('The topic has been saved successfully.', **null**, **3000**);  **this**.closeDialog(load.data);  }, err => {  err = **this**.\_user.\_decrypt(err.error.a);  **this**.\_user.openSnackBar('Oh, no! mistake has happened!', **null**, **3000**);  **this**.\_user.setLoading(**false**);  **this**.closeDialog(load.data);  });  }  closeDialog(data) {  **this**.dialogReg.close(data);  }  } |

**CREATE AND EDIT POST CLASSFEED**

|  |
| --- |
| **export** **class** CrudPostComponent **implements** OnInit {  postForms: **FormGroup**;  editFiles = [];  editedFile\_dir: **any** = [];  filesToBeUpdated = [];  filePreviewAndUpload: **any** = [];  getRecipients = **new** getRecipients();  **constructor**(  **private** dialogReg: **MatDialogRef**<CRUDForumsComponent>,  @Inject(MAT\_DIALOG\_DATA) **public** data,  **private** fb: **FormBuilder**,  **public** \_user: **UserService**,  **public** \_ds: **DataService**,  **public** \_uploadservice: **UploadingService**,  **private** \_common: **CommonService**,  ) {  }  ngOnInit(): **void** {  **this**.\_uploadservice.files = [];  **if** (**this**.data.type == 'edit\_post') {  **this**.editPost();  }  }  editPost() {  **this**.postForms = **this**.fb.group({  content\_fld: [**this**.data.data.content\_fld, Validators.required],  });  **this**.editedFile\_dir = **this**.data.data.dir\_fld;  **if** (**this**.editedFile\_dir !== '') {  **this**.editFiles = **this**.\_uploadservice.splitFilestring(**this**.editedFile\_dir);  }  }  async savePost() {  **if** ((**this**.filePreviewAndUpload.length == **0** && **this**.postForms.value.content\_fld == '')) **return**;  **this**.\_user.setLoading(**true**);  await **this**.\_uploadservice.uploadFiles(**this**.filePreviewAndUpload).then((files: **any**) => {  **let** arr: **any**;  **let** recipients = **this**.getRecipients.memberMap(**this**.\_user.getClassMembers().student);  **if** (files != **null**) {  arr = [...**this**.editedFile\_dir.split(':'), ...files.split(':')];  arr = arr.filter((f) => {  **return** f != "";  });  arr = arr.join(':');  } **else** **if** (files == **null**) {  arr = **this**.editedFile\_dir;  }  **let** load = {  data: {  "content\_fld": **this**.postForms.value.content\_fld,  "withfile\_fld": arr == '' ? 0 : **1**,  "dir\_fld": arr,  "datetime\_fld": **this**.data.data.datetime\_fld,  }, notif: {  id: **this.\_user.getUserID**(),  recipient: **recipients.join**(':'),  message: **this.\_user.getUserFullname**() + " Edited Post in " + **this**.\_user.getClassroomInfo().subjdesc\_fld,  module: 'Classroom',  }  };  **this**.\_common.commonSubscribe('editpost/' + **this**.data.data.postcode\_fld + '/' + 'post', load, **1**)  .then((data: **any**) => {  **this**.\_user.setLoading(**false**);  **this**.closeDialog(data.payload);  **this**.\_user.setLoading(**false**);  });  });  }  async deleteFiles(i, path) {  **let** newFiles: **any** = [];  **this**.\_common.commonSubscribe('deletefile', { dir\_fld: **path** }, **1**)  .then(() => {  **this**.editFiles.splice(i, **1**);  **this**.editFiles.map((f) => {  newFiles.push(f.name + '?' + f.path);  })  })  .then(() => {  newFiles = newFiles.join(':');  **let** load = {  data: {  "withfile\_fld": newFiles == '' ? 0 : **1**,  "dir\_fld": newFiles  }, notif: {  id: **this.\_user.getUserID**(),  recipient: **this.\_user.getUserID**(),  message: **this.\_user.getUserFullname**() + " Edited Post in " + **this**.\_user.getClassroomInfo().subjdesc\_fld,  module: 'Classroom',  }  }  **this**.\_common.commonSubscribe('editpost/' + **this**.data.data.postcode\_fld + '/' + 'post', load, **1**)  .then((data: **any**) => {  **this**.editedFile\_dir = data.payload[**0**].dir\_fld;  });  }).**catch**((error) => {  error  });  }  getFile(event: **any**) {  **this**.filePreviewAndUpload = **this**.\_uploadservice.getFilePreview(event);  }  removeFilePreviews(i) { **this**.filePreviewAndUpload.splice(i, **1**); }  closeDialog(data) {  **this**.dialogReg.close(data);  }  } |

**CREATE AND EDIT RESOURCES**

|  |
| --- |
| **export** **class** CrudResourcesComponent **implements** OnInit {  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(  Breakpoints.XSmall  );  ResourceFormGroup!: FormGroup;  filePreviewAndUpload: **any** = [];  editedFile\_dir: **any** = [];  editFiles: **any** = [];  editLink: **any** = [];  filesToBeUpdated = [];  getRecipients = **new** getRecipients();  file = '';  linkResourcesList = [];  **constructor**(  **private** dialogReg: **MatDialogRef**<CrudResourcesComponent>,  @Inject(MAT\_DIALOG\_DATA) **public** data,  **private** dialog: **MatDialog**,  **private** fb: **FormBuilder**,  **public** \_user: **UserService**,  **private** \_ds: **DataService**,  **private** \_uploadservice: **UploadingService**,  **public** uploadservice: **UploadingService**,  **private** \_common: **CommonService**,  **private** breakpointObserver: **BreakpointObserver**  ) { }  ngOnInit(): **void** {  **this**.intializeFunctions();  }  intializeFunctions() {  **this**.uploadservice.files = [];  **this**.ResourceFormGroup = **this**.fb.group({  res\_title: ['', Validators.required],  res\_desc: [''],  res\_topic: [''],  });  **if** (**this**.data.type == 'edit\_resources') **return** **this**.editResources();  }  getErrorMessage() {  **if** (**this**.ResourceFormGroup.controls.res\_title.hasError('required'))  **return** 'You must enter a value';  }  editResources() {  **this**.ResourceFormGroup = **this**.fb.group({  res\_title: [**this**.data.data.title\_fld, Validators.required],  res\_desc: [**this**.data.data.desc\_fld],  res\_topic: [**this**.data.data.topiccode\_fld],  });  **if** (!**this**.data.data.filedir\_fld.includes('%=!!=%')) {  **this**.editedFile\_dir = **this**.data.data.filedir\_fld;  **this**.editFiles = **this**.uploadservice.splitFilestring(  **this**.data.data.filedir\_fld  );  } **else** {  **let** temp = **this**.data.data.filedir\_fld.split('%=!!=%');  **this**.editedFile\_dir = temp[**0**];  **this**.editFiles = **this**.uploadservice.splitFilestring(temp[**0**]);  **this**.linkResourcesList = temp[**1**].split('//\*//');  }  }  async addResources() {  **let** linkTempString = **this**.linkResourcesList.join('//\*//');  **if** (**this**.filePreviewAndUpload.length == **0** || **this**.ResourceFormGroup.invalid)  **return**;  **this**.\_user.setLoading(**true**);  await **this**.\_uploadservice  .uploadFiles(**this**.filePreviewAndUpload)  .then((files: **string**) => {  files = `${files}%=!!=%${linkTempString}`;  **if** (linkTempString.length == **0**) {  files = files.replace('%=!!=%', '');  }  *// let recipients = this.getRecipients.memberMap(this.\_user.getClassMembers().student);*  **let** load = {  data: {  authorid\_fld: **this.\_user.getUserID**(),  topiccode\_fld: !**this**.ResourceFormGroup.value.res\_topic  ? 0  : **this.ResourceFormGroup.value.res\_topic**,  title\_fld: **this.ResourceFormGroup.value.res\_title**,  desc\_fld: **this.ResourceFormGroup.value.res\_desc**,  classcode\_fld: **this.\_user.getClassroomInfo**().classcode\_fld,  withfile\_fld: **files** == **null** ? 0 : **1**,  filedir\_fld: **files** == **null** ? '' : files,  },  notif: {  id: **this.\_user.getUserID**(),  *// recipient: recipients.join(':'),*  recipient: **null**,  message:  **this.\_user.getUserFullname**() +  ' Added Resources in ' +  **this**.\_user.getClassroomInfo().subjdesc\_fld,  module: 'Classroom',  },  };  **this**.\_common  .commonSubscribe('addresource', load, **1**)  .then((data: **any**) => {  **this**.\_user.setLoading(**false**);  **this**.filePreviewAndUpload = [];  **this**.closeDialog(data.payload);  **this**.\_user.openSnackBar(  'The resources has been successfully added. 🐱‍🏍 ',  'Close',  **3000**  );  })  .**catch**((error) => {  **this**.\_user.setLoading(**false**);  **this**.\_user.openSnackBar(  'An error occurred; please try again!',  'Close',  **3000**  );  });  });  }  async saveResources() {  **let** tempArray: **any**;  await **this**.\_uploadservice  .uploadFiles(**this**.filePreviewAndUpload)  .then((files: **any**) => {  **if** (files != **null**) {  tempArray = [...**this**.editedFile\_dir.split(':'), ...files.split(':')];  tempArray = tempArray.filter((element: **any**) => {  **return** element != '';  });  tempArray = tempArray.join(':');  }  **if** (files == **null**) {  tempArray = **this**.editedFile\_dir;  }  tempArray = `${tempArray}%=!!=%${**this**.linkResourcesList.join('//\*//')}`;  **if** (**this**.linkResourcesList.length == **0**) {  tempArray = tempArray.replace('%=!!=%', '');  }  **let** load = {  data: {  rescode\_fld: **this.data.data.rescode\_fld**,  authorid\_fld: **this.\_user.getUserID**(),  topiccode\_fld: **this.ResourceFormGroup.value.res\_topic**,  title\_fld: **this.ResourceFormGroup.value.res\_title**,  desc\_fld: **this.ResourceFormGroup.value.res\_desc**,  classcode\_fld: **this.\_user.getClassroomInfo**().classcode\_fld,  withfile\_fld: **tempArray** == '' || tempArray == **null** ? 0 : **1**,  filedir\_fld: **tempArray**,  datetime\_fld: **this.data.data.datetime\_fld**,  },  notif: {  id: **this.\_user.getUserID**(),  *// recipient: recipients.join(':'),*  recipient: **null**,  message:  **this.\_user.getUserFullname**() +  ' Edited Resources in ' +  **this**.\_user.getClassroomInfo().subjdesc\_fld,  module: 'Classroom',  },  };  **this**.\_common  .commonSubscribe('editres/' + **this**.data.data.rescode\_fld, load, **1**)  .then(() => {  **this**.closeDialog(load.data);  **this**.\_user.setLoading(**false**);  });  });  }  async deleteFiles(i, path) {  **this**.\_user.setLoading(**true**);  **let** newFiles: **any** = [];  **this**.\_common  .commonSubscribe('deletefile', { dir\_fld: **path** }, **1**)  .then(() => {  **this**.editFiles.splice(i, **1**);  **this**.editFiles.map((f) => {  newFiles.push(f.name + '?' + f.path);  });  })  .then(() => {  newFiles = newFiles.join(':');  **let** load = {  data: {  withfile\_fld: **newFiles** == '' ? 0 : **1**,  filedir\_fld: **newFiles**,  },  notif: {  id: **this.\_user.getUserID**(),  recipient: **this.\_user.getUserID**(),  message:  **this.\_user.getUserFullname**() +  ' Edited Rescources in ' +  **this**.\_user.getClassroomInfo().subjdesc\_fld,  module: 'Classroom',  },  };  **this**.\_common  .commonSubscribe('editres/' + **this**.data.data.rescode\_fld, load, **1**)  .then((data: **any**) => {  **this**.\_user.setLoading(**false**);  **this**.editedFile\_dir = data.payload.filedir\_fld;  });  })  .**catch**((error) => {  error;  });  }  getFile(event: **any**) {  **this**.filePreviewAndUpload = **this**.\_uploadservice.getFilePreview(event);  }  removeFilePreviews(i) {  **this**.filePreviewAndUpload.splice(i, **1**);  }  closeDialog(data) {  **this**.dialogReg.close(data);  }  getFileExtension(filename) {  **if** (filename == 'pdf') {  **return** '#EA462E';  } **else** **if** (filename == 'docx') {  **return** '#2D5292';  } **else** **if** (filename == 'ppt') {  **return** '#CA4223';  } **else** **if** (filename == 'zip') {  **return** '#B23333';  } **else** **if** (filename == 'txt') {  **return** '#546A7A';  } **else** {  **return** '#222';  }  }  addLinkModal() {  **let** data = {  type: 'AddResourcesLink',  };  **const** dialogRef = **this**.dialog.open(ViewDataOnlyComponent, {  maxWidth: '30%',  maxHeight: '50%',  data: **data**,  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(  (result) => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('50%', 'auto');  }  }  );  dialogRef.afterClosed().subscribe((result) => {  responsiveDialogSubscription.unsubscribe();  **if** (result == **undefined**) **return**;  **this**.linkResourcesList.push(result);  });  }  removeLink(i) {  **this**.linkResourcesList.splice(i, **1**);  }  *// removeEditLink(i) { this.editLink.splice(i, 1); }*  } |

**GLOBAL DELETE DIALOG**

|  |
| --- |
| **export** **class** DeleteRecordComponent **implements** OnInit {  getRecipients = **new** getRecipients();  **constructor**(**public** dialogReg: **MatDialogRef**<DeleteRecordComponent>,  @Inject(MAT\_DIALOG\_DATA) **public** data: **any**,  **private** dialog: **MatDialog**,  **public** \_user: **UserService**,  **public** \_ds: **DataService**) { }  ngOnInit(): **void** { }  onNoClick(): **void** {  **this**.dialogReg.close();  }  deletePost() {  **let** load = {  data: { isdeleted\_fld: **1** }  , notif: **null**  }  **this**.\_ds.\_httpRequest('delpost/' + **this**.data.data.postcode\_fld, load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.closeDialog(**true**);  }, err => {  err = **this**.\_user.\_decrypt(err.error.a);  });  }  deleteComments() {  **let** load = {  data: {  isdeleted\_fld: **1**  }, notif: **null**  }  **this**.\_ds.\_httpRequest('delcomment/' + **this**.data.data.commentcode\_fld, load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.closeDialog(**true**);  }, err => {  err = **this**.\_user.\_decrypt(err.error.a);  });  }  deleteAct() {  **let** load = {  data: {  isdeleted\_fld: **1**  }, notif: **null**  }  **this**.\_ds.\_httpRequest('delact/' + **this**.data.data.actcode\_fld, load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.closeDialog(**true**);  }, err => {  err = **this**.\_user.\_decrypt(err.error.a);  });  }  deleteRes() {  **let** load = {  data: {  isdeleted\_fld: **1**  }, notif: **null**  }  **this**.\_ds.\_httpRequest('editres/' + **this**.data.data.rescode\_fld, load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.closeDialog(**true**);  }, err => {  err = **this**.\_user.\_decrypt(err.error.a);  **this**.closeDialog(**true**);  });  }  deleteForums() {  **let** load = {  data: {  isdeleted\_fld: **1**  }, notif: **null**  }  **this**.\_ds.\_httpRequest('editforum/' + **this**.data.data.forumcode\_fld, load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.closeDialog(**true**);  }, err => {  err = **this**.\_user.\_decrypt(err.error.a);  **this**.closeDialog(**true**);  });  }  deleteSubForums() {  **let** load = {  data: { isdeleted\_fld: **1** },  notif: **null**  }  **this**.\_ds.\_httpRequest('delsubforum/' + **this**.data.data.subcode\_fld, load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.closeDialog(**true**);  }, err => {  err = **this**.\_user.\_decrypt(err.error.a);  **this**.closeDialog(**true**);  });  }  deleteReply() {  **let** load = {  data:  { isdeleted\_fld: **1** }  , notif: **null**  }  **this**.\_ds.\_httpRequest('delforumcont/' + **this**.data.data.contentcode\_fld, load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.closeDialog(**true**);  }, err => {  err = **this**.\_user.\_decrypt(err.error.a);  **this**.closeDialog(**true**);  });  }  deleteOption(num){  **this**.closeDialog(num)  }  closeDialog(data) {  **this**.dialogReg.close(data);  }  } |

**VIEW RESOURCES**

|  |
| --- |
| **export** **class** ResourcePreviewComponent **implements** OnInit {  linkPath: **any**;  name: **string**;  **constructor**(**public** \_upload:**UploadingService**, **private** \_ds: **DataService**, **public** \_user:**UserService**, @Inject(MAT\_DIALOG\_DATA) **public** data: **any**, **private** sanitizer: **DomSanitizer**) { }  ngOnInit(): **void** {  **this**.disableRightClick()  **this**.getLink(**this**.data.resourceString);  }  disableRightClick(){  *// window.frames['resourceframe'].document.oncontextmenu = ()=>{*  *// return false;*  *// }*  *// document.getElementById("resourceframe").oncontextmenu = function(){alert("No way!"); return false;}*  }  getLink(link){  **let** embbed:**string**;  **this**.name = **this**.\_upload.getfileExt(link);  **if** (**this**.name.includes('docx') || **this**.name.includes('xlsx') || **this**.name.includes('pptx')) {  embbed = `https:*//view.officeapps.live.com/op/embed.aspx?src=${link}&embedded=true`*  }  **if** (**this**.name.includes('txt') || **this**.name.includes('pdf') || **this**.name.includes('csv')) {  embbed = link;  }  **if** (**this**.name.includes('jpg') || **this**.name.includes('jpeg') || **this**.name.includes('png')) {  embbed = link;  }  *// console.log(embbed);*    **if** (embbed != **undefined**) {  **this**.linkPath = **this**.sanitizer.bypassSecurityTrustResourceUrl(embbed);  }**else**{  **this**.linkPath = **null**  }    *// this.linkPath = this.sanitizer.bypassSecurityTrustResourceUrl(embbed);*  }  } |

**STUDENTS WORK FILES**

|  |
| --- |
| **export** **class** StudentworksComponent **implements** OnInit, OnDestroy {  assigned: **boolean**;  studentworks: **Object**;  filterObject: **any** = [];  **default**: any;  studentActivitySummary: **any** = [];  **constructor**(  **private** dialogReg: **MatDialogRef**<StudentworksComponent>,  @Inject(MAT\_DIALOG\_DATA) **public** data: **any**,  **public** \_user: **UserService**,  **public** \_ds: **DataService**,  **private** \_common: **CommonService**  ) { }  ngOnInit(): **void** {  **this**.getStudentSummary();  }  ngOnDestroy(): **void** {  **this**.data = **null**;  }  getStudentSummary() {  **this**.\_user.setLoading(**true**);  **let** load = {  userid: **this.data.data.studnum\_fld**,  classcode: **this.\_user.getClassroomInfo**().classcode\_fld,  };  **this**.\_common.commonSubscribe('getstudentsummary', load, **1**).then((data: **any**) => {  **this**.studentActivitySummary = data.payload;  **this**.filterByKeys({ value: **4** });  **this**.\_user.setLoading(**false**);  }).**catch**((error) => {  console.log(error);  **this**.\_user.setLoading(**false**);  })  }  filterByKeys(filterkey) {  **let** filteredObject = [];  filteredObject = **this**.studentActivitySummary.filter(el => {  **if** (filterkey.value == **4**) **return** filteredObject.push(el);  **if** (filterkey.value == **0**) **return** el.issubmitted\_fld == **null**;  **return** el.issubmitted\_fld == filterkey.value;  })  **this**.filterObject = filteredObject;  }  } |

**UPDATE PROFILE**

|  |
| --- |
| **export** **class** UpdateProfileComponent **implements** OnInit {  firstFormGroup: **FormGroup**;  minDate = **new** Date();  userCred: **any** = **this**.data.data;  **constructor**(**private** dialogReg: **MatDialogRef**<UpdateProfileComponent>,  @Inject(MAT\_DIALOG\_DATA) **public** data: **any**,  **private** \_dialog: **MatDialog**,  **public** \_user: **UserService**, **private** \_ds: **DataService**,  **private** fb: **FormBuilder**,  **private** datepipe: **DatePipe**,) {  }  ngOnInit(): **void** {  **this**.intializeFunctions();  }  intializeFunctions() {  **this**.firstFormGroup = **this**.fb.group({  fname\_fld: [''],  mname\_fld: [''],  lname\_fld: [''],  specialization\_fld: [],  email\_fld: ['', Validators.required],  extname\_fld: [''],  bdate\_fld: ['', Validators.required],  position\_fld: [''],  educ\_fld: [''],  assignment\_fld: ['']  });  **this**.editProfile();  }  editProfile() {  **this**.firstFormGroup = **this**.fb.group({  fname\_fld: [**this**.data.fname\_fld, [Validators.required, Validators.pattern(/^[A-Za-z ]+$/)]],  mname\_fld: [**this**.data.mname\_fld, Validators.pattern(/^[A-Za-z]+$/)],  lname\_fld: [**this**.data.lname\_fld, [Validators.required, Validators.pattern(/^[A-Za-z ]+$/)]],  specialization\_fld: [**this**.data.specialization\_fld],  email\_fld: [**this**.data.email\_fld, Validators.required],  extname\_fld: [**this**.data.extname\_fld],  bdate\_fld: [**this**.data.bdate\_fld, Validators.required],  position\_fld: [**this**.data.position\_fld],  educ\_fld: [**this**.data.educ\_fld],  assignment\_fld: [**this**.data.assignment\_fld],  });  }  getErrorMessage() {  **if** (**this**.firstFormGroup.controls.fname\_fld.hasError('pattern')) **return** 'Invalid Input - Only letters may be alowed.';  **if** (**this**.firstFormGroup.controls.mname\_fld.hasError('pattern')) **return** 'Invalid Input - Only letters may be alowed.';  **if** (**this**.firstFormGroup.controls.lname\_fld.hasError('pattern')) **return** 'Invalid Input - Only letters may be alowed.';  **if** (**this**.firstFormGroup.controls.fname\_fld.hasError('required')) **return** 'You must enter a value';  **if** (**this**.firstFormGroup.controls.email\_fld.hasError('required')) **return** 'You must enter a value';  **if** (**this**.firstFormGroup.controls.lname\_fld.hasError('required')) **return** 'You must enter a value';  **if** (**this**.firstFormGroup.controls.bdate\_fld.hasError('required')) **return** 'You must enter a value';  }  updateProfile() {  **if** (**this**.firstFormGroup.invalid) **return**;  **this**.\_user.setLoading(**true**);  **let** load = {  data: {  fname\_fld: **this.firstFormGroup.value.fname\_fld**,  mname\_fld: **this.firstFormGroup.value.mname\_fld**,  lname\_fld: **this.firstFormGroup.value.lname\_fld**,  specialization\_fld: **this.firstFormGroup.value.specialization\_fld**,  email\_fld: **this.firstFormGroup.value.email\_fld**,  extname\_fld: **this.firstFormGroup.value.extname\_fld**,  position\_fld: **this.firstFormGroup.value.position\_fld**,  educ\_fld: **this.firstFormGroup.value.educ\_fld**,  assignment\_fld: **this.firstFormGroup.value.assignment\_fld**,  bdate\_fld: **this.datepipe.transform**(**this**.firstFormGroup.value.bdate\_fld, 'yyyy-MM-dd'),  }, notif: {  id: **this.\_user.getUserID**(),  recipient: **this.\_user.getUserID**(),  message: **this.\_user.getUserFullname**() + "Update Profile",  module: 'Profile',  }  }  **this**.\_ds.\_httpRequest('updateprofile/' + **this**.\_user.getUserID(), load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.\_user.openSnackBar('Profile has been successfully updated.', **null**, **3000**);  **this**.\_user.updateUserData({  fullname: `${dt.payload[**0**].fname\_fld} ${dt.payload[**0**].mname\_fld} ${dt.payload[**0**].lname\_fld} ${dt.payload[**0**].extname\_fld}`,  emailadd: **dt.payload**[**0**].email\_fld  });  **this**.closeDialog(dt.payload);  **this**.\_user.setLoading(**false**);  }, err => {  err = **this**.\_user.\_decrypt(err.error.a);  });  }  closeDialog(data) {  **this**.dialogReg.close(data);  }  } |

**UPDATE PROFILE PIC**

|  |
| --- |
| **export** **class** UpdateProfilePictureComponent **implements** OnInit {  imageUrl: **ArrayBuffer** | **string** = **this**.\_ds.downloadURL + **this**.\_user.getUserImage();  files: **any** = [];  filepath: **string**;  base64Image: **string** = **null**;  croppedImage: **string** = **null**;  orientation: **any**;  **constructor**(  **public** \_user: **UserService**,  **public** \_ds: **DataService**,  @Inject(MAT\_DIALOG\_DATA) **public** data: **any**,  **private** dialogReg: **MatDialogRef**<UpdateProfilePictureComponent>,  **private** \_imageCompress: **NgxImageCompressService**,  ) { }  ngOnInit(): **void** { }  compressFile() {  **this**.\_imageCompress.uploadFile().then(({ image, orientation }) => {  **this**.base64Image = image;  **this**.orientation = orientation;  });  }  imageCropped(e: **ImageCroppedEvent**) {  **this**.croppedImage = e.base64;  }  convertToBlob(e) {  **let** arr = e.split(','),  mime = arr[**0**].match(/:(.\*?);/)[**1**],  bstr = atob(arr[**1**]),  n = bstr.length,  u8arr = **new** Uint8Array(n);  **while** (n--) {  u8arr[n] = bstr.charCodeAt(n);  }  **return** **new** File([u8arr], 'Profile.png', { type: **mime** });  }  async updateProfile() {  **this**.\_user.setLoading(**true**);  **this**.\_imageCompress.compressFile(**this**.croppedImage, **this**.orientation, **50**, **100**).then(res => {  **let** imgData = **new** FormData();  **let** img: **string**;  **let** imgToUpload: **File**;  img = res;  imgToUpload = **this**.convertToBlob(img)  imgData.append('file[]', imgToUpload);  **this**.\_ds.\_httpRequest('editProfile/' + **this**.\_user.getUserID(), imgData, **3**).subscribe(async (data: **any**) => {  data = await **this**.\_user.\_decrypt(data.a);  **let** load = {  data: {  profilepic\_fld: **data.payload.filepath**  },  notif: {  id: **this.\_user.getUserID**(),  recipient: **this.\_user.getUserID**(),  message: **this.\_user.getUserFullname**() + ' Profile picture update.',  module: 'Profile Component'  }  }  **this**.\_ds.\_httpRequest('updateprofile/' + **this**.\_user.getUserID(), load, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.\_user.updateUserData({ img: **dt.payload**[**0**].profilepic\_fld + "?e=" + Math.random().toString() });  **this**.dialogReg.close();  **this**.\_user.setLoading(**false**);  }, err => {  err = **this**.\_user.\_decrypt(err.error.a);  **this**.\_user.setLoading(**false**);  });  });  });  }  } |

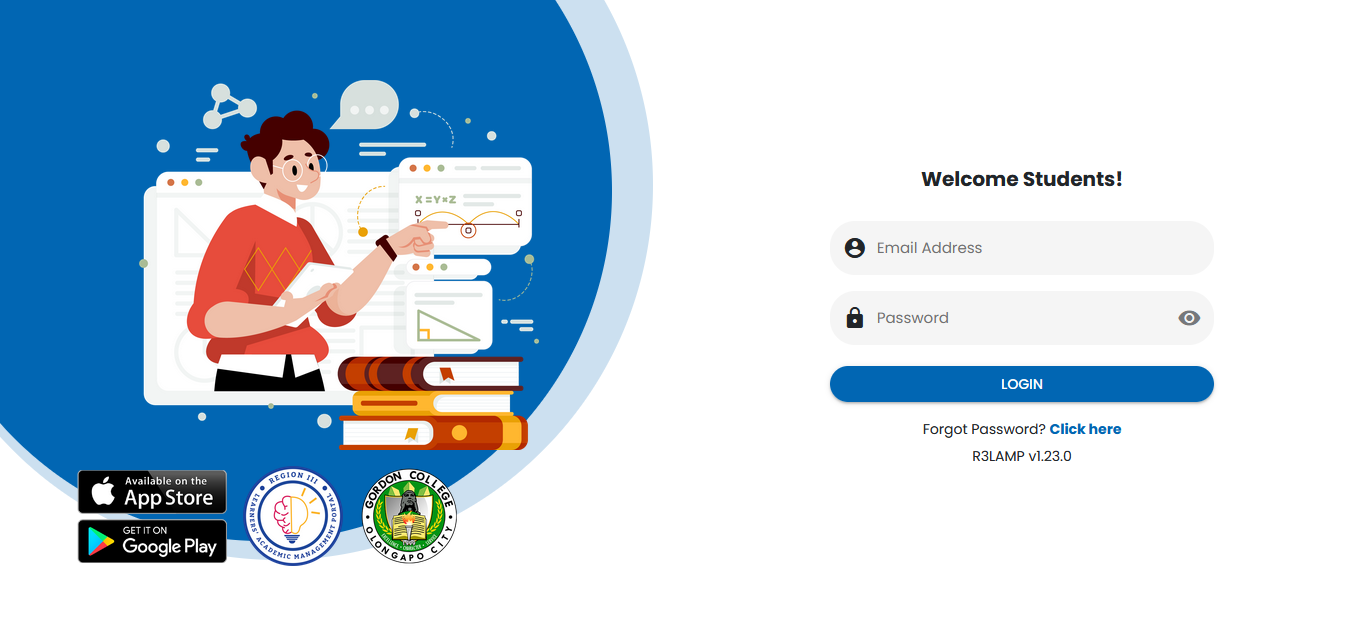
**VIEW DATA ONLY**

|  |
| --- |
| **export** **class** ViewDataOnlyComponent **implements** OnInit {  Content: **any** = [];  date = **new** Date();  resourcesfiles = [];  ScoringForm: **FormGroup**;  userQuestData: **any** = []  questList = **new** QuestDataItem();  shopItems = **new** ShopItems();  itemsAcquired = **new** Array();  badgesAcquired = **new** Array();  AddlinkForm: **FormGroup**;  resourcesLinkList = [];  **constructor**(**private** dialogReg: **MatDialogRef**<ViewDataOnlyComponent>, @Inject(MAT\_DIALOG\_DATA) **public** data,  **private** dialog: **MatDialog**, **public** uploadservice: **UploadingService**, **public** \_ds: **DataService**, **public** \_user: **UserService**, **private** fb: **FormBuilder**, **private** \_router: **Router**) {  **this**.Content = data;  }  ngOnInit(): **void** {  **this**.initializeComponents();  }  ngOnDestroy() {  }  getFileExtension(filename) {  **if** (filename == 'pdf') {  **return** '#EA462E';  } **else** **if** (filename == 'docx') {  **return** '#2D5292';  }  **else** **if** (filename == 'ppt') {  **return** '#CA4223';  }  **else** **if** (filename == 'zip') {  **return** '#B23333';  }  **else** **if** (filename == 'txt') {  **return** '#546A7A';  } **else** {  **return** '#222';  }  }  viewerSelected(data) {  **this**.closeDialog(data);  }  closeDialog(data) {  **this**.dialogReg.close(data);  }  getErrorMessage() {  **if** (**this**.ScoringForm.controls.score.hasError('required')) **return** 'You must enter a value';  }  savePoints() {  **if** (**this**.ScoringForm.invalid) **return**  **this**.closeDialog(**this**.ScoringForm.value.score);  }  initializeComponents() {  **if** (**this**.Content['type'] === 'AddResourcesLink') {  **this**.AddlinkForm = **this**.fb.group({  link: ['', [Validators.required, Validators.pattern(/(^|\s)((https?:\/\/)?[\w-]+(\.[\w-]+)+\.?(:\d+)?(\/\S\*)?)/gi)]],  })  }  **var** days = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday'];  days[**this**.date.getDay()];  **if** (**this**.Content['type'] == 'Resources') {  **if** (!**this**.data.data.filedir\_fld.includes("%=!!=%")) {  **this**.resourcesfiles = **this**.uploadservice.splitFilestring(**this**.Content.data.filedir\_fld);  } **else** {  **let** temp = **this**.Content.data.filedir\_fld.split("%=!!=%");  **this**.resourcesfiles = **this**.uploadservice.splitFilestring(temp[**0**]);  **this**.resourcesLinkList = temp[**1**].split("//\*//");  }  }  **if** (**this**.Content['type'] == 'score\_essay')  **this**.ScoringForm = **this**.fb.group({  score: [**this**.data.data.points, Validators.required],  });  *// this.getQuestData()*  }  userLogout() {  window.sessionStorage.clear();  **this**.dialog.closeAll();  **this**.\_router.navigate(['login']);  **this**.\_user.openSnackBar('Logged out successfully', **null**, **3000**);  }  getQuestData() {  **this**.\_ds.\_httpRequest('gameprogress', { studnum\_fld: **this.data.data.studnum\_fld** }, **1**).subscribe((data: **any**) => {  data = **this**.\_user.\_decrypt(data.a);  **this**.userQuestData = data.payload[**0**]  **this**.getAchievements();  **this**.getItems()  }, error => {  error = **this**.\_user.\_decrypt(error.error.a)  })  }  getAchievements() {  **let** userDataArrayListCode = **this**.userQuestData.badges\_acquired\_fld.split(';')  **this**.questList.quest.filter((questData) => {  userDataArrayListCode.forEach(badgeAcquired => {  questData.questcode\_fld === badgeAcquired ? **this**.badgesAcquired.push(questData) : **null**  });  })  }  getItems() {  **let** itemDataArrayListCode = **this**.userQuestData.items\_acquired\_fld.split(';')  **this**.shopItems.shopItems.filter((shopItems) => {  itemDataArrayListCode.forEach((itemAcquired) => {  shopItems.itemcode\_fld === itemAcquired ? **this**.itemsAcquired.push(shopItems) : **null**  })  })  }  addLinkResource() {  **if** (**this**.AddlinkForm.invalid) **return**;  **this**.closeDialog(**this**.AddlinkForm.value.link);  }  getAddLinkFormErrorMessage() {  **if** (**this**.AddlinkForm.controls.link.hasError('required')) **return** 'This field is required.';  **if** (**this**.AddlinkForm.controls.link.hasError('pattern')) **return** 'This is invalid link.';  }  } |

**VIEW POLL**

|  |
| --- |
| **export** **class** ViewpollComponent **implements** OnInit {  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(Breakpoints.XSmall);  getRecipients = **new** getRecipients();  editmode: **number**;  editcontent: **string** = '';  selectedIndex: **number**;  **constructor**  (@Inject(MAT\_DIALOG\_DATA) **public** data,  **public** \_common: **CommonService**,  **private** breakpointObserver: **BreakpointObserver**,  **public** \_user: **UserService**,  **public** \_ds: **DataService**,  **public** dialog: **MatDialog**,  **private** dialogReg: **MatDialogRef**<ViewpollComponent>,) { }  ngOnInit(): **void** {  **this**.editmode = **0**;  }  editOption(index, content: **any**) {  **this**.selectedIndex = index;  **this**.editcontent = content.choice;  **this**.editmode = **1**;  }  deleteRecords(type, item, i) {  **let** data = {  type: **type**,  data: **item**,  };  **const** dialogRef = **this**.dialog.open(DeleteRecordComponent, {  panelClass: '',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**,  disableClose: **true**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('20%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result) {  **this**.data.pollresponse\_fld.splice(i, **1**);  **this**.data.pollchoices\_fld.splice(i,**1**);  **this**.savePoll();  }  });  }  canceledit() {  **this**.editmode = **0**;  **this**.editcontent = '';  }  addOption(e) {  e.preventDefault();  **let** f = e.target.elements;  **this**.data.pollresponse\_fld.push({  choice: **f.option\_name.value**,  votes: **0**,  respondents: []  });  **this**.data.pollchoices\_fld.push(f.option\_name.value);  f.option\_name.value = **null**;  **this**.savePoll();  }  editChoices(e, i) {  e.preventDefault();  **let** f = e.target.elements;  **this**.data.pollresponse\_fld[i].choice = f.choice\_name.value;  **this**.data.pollchoices\_fld[i] = f.choice\_name.value;  **this**.savePoll();  }  savePoll() {  **this**.\_user.setLoading(**true**);  **let** load = {  data: {  "content\_fld": **this**.data.content\_fld + '($)' + **this**.data.pollchoices\_fld.join('(\*)'),  }, notif: {  id: **this.\_user.getUserID**(),  recipient: **null**,  message: **this.\_user.getUserFullname**() + " Edited Post in " + **this**.\_user.getClassroomInfo().subjdesc\_fld,  module: 'Classroom',  }  };  **this**.\_common.commonSubscribe('editpost/' + **this**.data.postcode\_fld + '/' + 'post', load, **1**)  .then((data: **any**) => {  **this**.\_user.setLoading(**false**);  **this**.editmode = **0**;  **this**.\_user.openSnackBar('Option has been successfully added and saved.', **null**, **2000**)  })  }  closeDialog(data) {  **this**.dialogReg.close(data);  }  } |

**STUDENT PANEL**

****

**STUDENT LOGIN.html**

|  |
| --- |
| **<div** id="box-container" class="bg-container"**>**  **<div** class="d-flex align-items-center justify-content-center flex-column"**>**  **<img** class="logo" appLazyLoad src="assets/logo/R3LAMP.png" alt="" **/>**  **</div>**  *<!--FORM SECTION-->*  **<div** class="row"**>**  *<!--SIGN UP-->*  **<div**  class="col d-flex align-items-center justify-content-center flex-column sign-up"  **>**  **<div** class="form-wrapper"**>**  **<div** class="form sign-up"**>**  **<button** mat-icon-button class="backtn" (click)="toggle()"**>**  **<mat**-icon**>**arrow\_back</mat-icon>  **</button>**  **<form**  class="forms"  (submit)="forgotPassword($event)"  ngNativeValidate  **>**  **<h1** class="text-center"**>**Forgot password?**</h1>**  **<label>**Student ID**</label>**  **<div** class="forms\_\_box"**>**  **<mat**-icon**>**account\_circle</mat-icon>  **<input** type="number" placeholder="Ex: 2020XXXX" required **/>**  **</div>**  **<label>**Email Address**</label>**  **<div** class="forms\_\_box"**>**  **<mat**-icon**>**email</mat-icon>  **<input**  type="email"  placeholder="Ex: Juandelacruz@lamp.com"  required  **/>**  **</div>**  **<label>**Birthdate**</label>**  **<div** class="forms\_\_box"**>**  **<mat**-icon**>**event</mat-icon>  **<input**  type="text"  placeholder="Birthdate"  onfocus="(this.type='date')"  required  **/>**  **</div>**  **<label>**Last Name**</label>**  **<div** class="forms\_\_box"**>**  **<mat**-icon**>**engineering</mat-icon>  **<input** type="password" placeholder="Last name" required **/>**  **</div>**  **<button** mat-raised-button type="submit" class="submitbutton"**>**  Submit  **</button>**  **</form>**  **<p>**  **<strong** class="mainbtn" (click)="toggle()"  **>**Back to the main Page**</strong**  **>**  **</p>**  **</div>**  **</div>**  **</div>**  *<!--END OF SIGN UP-->*  *<!--SIGN IN-->*  **<div**  class="col d-flex align-items-center justify-content-center flex-column sign-in"  **>**  **<div** class="form-wrapper"**>**  **<form** class="form sign-in" ngNativeValidate (submit)="onSubmit($event)"**>**  **<h1** class="text-center"**>**Welcome Students!**</h1>**  *<!-- <p class="subtitle">Login first</p> -->*  *<!-- <label for="">Email Address</label> -->*  **<div** class="form\_\_box"**>**  **<mat**-icon**>**account\_circle</mat-icon>  **<input**  type="text"  placeholder="Email Address"  id="param1"  required  **/>**  **</div>**  *<!-- <label for="">Password</label> -->*  **<div** class="form\_\_box"**>**  **<mat**-icon**>**lock</mat-icon>  **<input**  [type]="hide ? 'text' : 'password'"  id="param2"  type="password"  placeholder="Password"  required  **/>**  **<mat**-icon  class="fa-eye show"  [ngStyle]="{ 'opacity.%': 60 }"  \*ngIf="!hide; else showpass"  matTooltip="asdas"  (click)="hide = !hide"  **>**  visibility  </mat-icon>  **<ng**-template #showpass**>**  **<mat**-icon  class="fa-eye show"  [ngStyle]="{ 'opacity.%': 60 }"  (click)="hide = !hide"  **>**  visibility\_off  </mat-icon>  </ng-template>  **</div>**  **<p**  class="error"  style="text-align: center; color: red; margin: 0"  \*ngIf="errMsg != ''"  **>**  {{ errMsg }}  **</p>**  **<button** mat-raised-button class="submitbutton" type="submit"**>**  LOGIN  **</button>**  **<div** class="forget"**>**  **<small>**  Forgot Password?  **<strong** class="click-here" (click)="toggle()"**>**Click here**</strong>**  **</small>**  **<small>**R3LAMP v{{ user.version }}**</small>**  **</div>**  **</form>**  **</div>**  **</div>**  *<!--END OF SIGN IN-->*  **</div>**  **<div** class="row content-row"**>**  *<!--SIGN IN CONTENT-->*  **<div**  class="col flex-column align-items-center justify-content-center text-center"  **>**  **<div** class="text sign-in"**>**  *<!-- <div class="d-flex link-page">*  *<li>View home Page</li>*  *<li>Contact</li>*  *</div> -->*  **<div** class="img sign-in"**>**  **<img** appLazyLoad src="assets/images/login.png" alt="" **/>**  **</div>**  **<div** class="other-img"**>**  **<div** class="imgcontain"**>**  **<div** class="appstore"**>**  **<img** appLazyLoad src="assets/logo/apple.png" alt="app" **/>**  **<img** appLazyLoad src="assets/logo/playstore.png" alt="app" **/>**  **</div>**  **<img** appLazyLoad src="assets/logo/R3LAMP.png" alt="mainlogo" **/>**  *<!-- <a><img style="margin-left: 12px;" appLazyLoad src="assets/images/BHC.png" alt="" width="150px"></a> -->*  **<img** appLazyLoad src="assets/logo/logo.png" alt="schoologo" **/>**  **</div>**  **</div>**  **</div>**  **</div>**  *<!--END OF SIGN IN CONTENT-->*  *<!--SIGN UP CONTENT-->*  **<div**  class="col flex-column align-items-center justify-content-center text-center"  **>**  **<div** class="img sign-up"**>**  **<img**  appLazyLoad  src="assets/images/forgotpass.png"  alt=""  class="forgetimg"  **/>**  **</div>**  **<div** class="text sign-up"**></div>**  **</div>**  *<!--END OF SIGN UP CONTENT-->*  *<!--END OF CONTENT SECTION-->*  **</div>**  **</div>** |

**STUDENT LOGIN.ts**

|  |
| --- |
| **export** **class** LoginComponent **implements** OnInit {  errMsg: **any**;  hide: **any**;  isMobile: **boolean** = **false**;  isTablet: **boolean** = **false**;  isDesktopDevice: **boolean** = **false**;  **constructor**(  **public** dialog: **MatDialog**,  **private** router: **Router**,  **private** \_snackbar: **MatSnackBar**,  **private** deviceService: **DeviceDetectorService**,  **private** ds: **DataService**,  **public** user: **UserService**){ }  showconfirmpassword: **boolean** = **false**;  ngOnInit(): **void** {  **this**.getSettings();  setTimeout(() => {  **let** container:**any** = document.getElementById('box-container');  container.classList.toggle('sign-in');  }, **200**)  }  getSettings(){  **this**.ds.\_httpRequest('getsettings',**null**,**2**).subscribe((data:**any**)=>{  data = **this**.user.\_decrypt(data.a);  **this**.user.setSettings(data.payload[**0**]);  },er=>{  er = **this**.user.\_decrypt(er.error.a);  })  }  detectDevice() {  **this**.isMobile = **this**.deviceService.isMobile();  **this**.isTablet = **this**.deviceService.isTablet();  **this**.isDesktopDevice = **this**.deviceService.isDesktop();  }  toggle(){  **let** container:**any** = document.getElementById('box-container');  container.classList.toggle('sign-in');  container.classList.toggle('sign-up');  }    onSubmit(e){  e.preventDefault();  **this**.user.setLoading(**true**);  **let** f = e.target.elements;  **let** param1 = f.param1.value;  **let** param2 = f.param2.value;  **let** param3: **number** = **1**;  **let** param4: **string** = **this**.user.isMobile() ? 'Web Access using Mobile phone' : 'Web Access using Desktop/Laptop';    **this**.ds.\_httpRequest('studentlogin',{param1,param2,param3,param4},**2**).subscribe((data:**any**)=>{  data = **this**.user.\_decrypt(data.a);  *// console.log(data);*    **if** (data.status.remarks === 'success') {  **this**.user.setLoading(**false**);  **this**.\_snackbar.open("Welcome, "+data.payload.fullname +"!", "", {duration:**1500**});  **this**.user.setUserData(data.payload);  **this**.user.setProfilePicture(data.payload.img);  **this**.user.setLoginState();  **if** (data.payload.reset == **1**) {  **this**.changePassword();  } **else** {  **this**.router.navigate(['main/classes']);  }    } **else** {  **this**.errMsg = data.status.message;  }  }, (er) => {  **let** data = **this**.user.\_decrypt(er.error.a);  **this**.user.setLoading(**false**);  **this**.errMsg = data.status.message;  });  }  openShowPassword(pw: **string**) {  **let** condel  **const** dialogConfig = **new** MatDialogConfig();  **if** (**this**.user.isMobile()) {  dialogConfig.maxHeight = '90vw';  dialogConfig.minHeight = 'auto';  dialogConfig.minWidth = '70vw';  dialogConfig.maxWidth = '90vw';  } **else** {  dialogConfig.minHeight = '20vh';  dialogConfig.minWidth = '15vw';  dialogConfig.maxHeight = 'auto';  }  dialogConfig.disableClose = **true**;  dialogConfig.data = {  option: 'forgotpass',  newpw: **pw**  }  **const** dialogRef = **this**.dialog.open(DialogComponent, dialogConfig);  dialogRef.afterClosed().subscribe(result => {  **this**.toggle();  });  }  changePassword(){  **const** dialogConfig = **new** MatDialogConfig();  **const** dialogRef = **this**.dialog.open(ChangePassModalComponent,dialogConfig);  }  forgotPassword(e){  e.preventDefault();  **let** param1 = e.target[**0**].value;  **let** param2 = e.target[**2**].value;  **let** param3 = e.target[**1**].value;  **let** param4 = e.target[**3**].value;  **let** load = { param1, param2, param3, param4 }  *// console.log(load);*    **this**.ds.\_httpRequest('forgotpass',load,**2**).subscribe((data:**any**)=>{  data = **this**.user.\_decrypt(data.a);  *// console.log(data);*  **this**.\_snackbar.open(data.status.message, **null**, {duration:**1500**});  *// this.sendMail(param3,data.payload.pass);*  **this**.openShowPassword(data.payload.pass);  e.target[**0**].value = **null**;  e.target[**1**].value = **null**;  e.target[**2**].value = **null**;  e.target[**3**].value = **null**;    },er=>{  er = **this**.user.\_decrypt(er.error.a);  **this**.\_snackbar.open(er.status.message, **null**, {duration:**1500**});  *// console.log(er);*    });    }  sendMail(email,password){  **let** load = {  emailArray:[email],  body: "Hello, Your new password is "+password,  subject:"Reset Password"  }  *// console.log(load);*    **this**.ds.\_httpRequest('mailer',load,**5**).subscribe((data:**any**)=>{  data = **this**.user.\_decrypt(data.a);  *// console.log("success");*    },er =>{  er = **this**.user.\_decrypt(er.error.a);  *// console.log(er);*  })  }  } |

**LAYOUT (SIDEMENU)**

|  |
| --- |
| **export** **class** LayoutComponent **implements** OnInit {  @ViewChild('drawer') drawer: **any**;  date = **new** Date();  value = **70**;  isHandset$: **Observable**<**boolean**> = **this**.\_breakpointObserver.observe(Breakpoints.Handset)  .pipe(  map(result => result.matches),  shareReplay()  );  notifications = [];  notifObject: **any** = [];  count=**0**;  **constructor**(  **public** dialog:**MatDialog**, **private** \_breakpointObserver: **BreakpointObserver**, **private** router: **Router**, **public** ds: **DataService**, **public** user: **UserService**, **private** \_snackbar: **MatSnackBar**) { }  showNotification: **boolean** = **false**;  ngOnInit(): **void** {  *// this.getnotification()*  **this**.getQuestData()  *// this.chat.openWebSocket();*  setInterval(() => {  **this**.date = **new** Date();  }, **1000**);  }  closeSideNav() {  **if** (**this**.drawer.\_mode == 'over') {  **this**.drawer.close();  }  }  logout(option) {  **const** dialogConfig = **new** MatDialogConfig();  dialogConfig.autoFocus = **true**;  dialogConfig.disableClose = **true**;  **if** (**this**.user.isMobile()) {  dialogConfig.width = '100%';  } **else** {  dialogConfig.width = '18%';  }  dialogConfig.data = {  option:**option**  }  **const** logout = **this**.dialog.open(DialogComponent,dialogConfig);  logout.afterClosed().subscribe((result:**any**)=>{  **if** (result.data) {  window.sessionStorage.clear();  **this**.\_snackbar.open("Successfully logged out", "", {duration:**1500**});  **this**.router.navigate(['login']);  }  })  }  openNotification() {  **this**.showNotification = !**this**.showNotification;  }  closeNotification(state:**boolean**) {  **this**.showNotification = state;  }  getnotification() {  **this**.ds.\_httpRequest('getnotif',**null**,**1**).subscribe((data:**any**)=>{  data = **this**.user.\_decrypt(data.a);  **this**.notifObject = data.payload;  **this**.count = data.payload.length;    },er=>{  er = **this**.user.\_decrypt(er.error.a);  **this**.count = **0**;  })  }  getQuestData() {  **this**.ds.\_httpRequest('gameprogress',**null**,**1**).subscribe((data:**any**)=>{  data = **this**.user.\_decrypt(data.a);    **this**.user.setQuestData(data.payload[**0**]);  },error => {  error = **this**.user.\_decrypt(error.error.a)    })  }    } |

**CLASSES**

|  |
| --- |
| **export** **class** ClassComponent **implements** OnInit {  allClasses: **any**[]= [];  classes: **any**[] = [];  teachersList: **any**[] = [];  isLoading: **boolean** = **true**;  schedule: **any** = [];  perfectscore: **any** = [];  score: **number**;  showLoader : **boolean** = **true**;  quests: **any**;  date = **new** Date();  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(Breakpoints.XSmall);  **constructor**(**public** \_user:**UserService**,**public** \_ds:**DataService**, **public** dialog: **MatDialog**, **private** breakpointObserver: **BreakpointObserver**) { }  ngOnInit(): **void** {  **this**.getClass();  setInterval(() => {  **this**.date = **new** Date();  }, **1000**);  }  getClass() {  **this**.\_ds.\_httpRequest('getclasslist', { userid: **this.\_user.getUserID**(), acadyear:**this.\_user.getAcadYear**(), semester:**this.\_user.getSemester**() }, **1**).subscribe((res: **any**)=>{  res = **this**.\_user.\_decrypt(res.a);  **this**.allClasses = **this**.sortClass(res.payload);  **this**.classes = **this**.allClasses;  **this**.\_user.setStudentSchedule(**this**.allClasses);  **this**.isLoading = **false**;  **this**.showLoader = **false**;  },er =>{  er = **this**.\_user.\_decrypt(er.error.a);  **this**.allClasses = [];  **this**.classes = [];  **this**.\_user.setStudentSchedule(**this**.allClasses);  **this**.showLoader = **false**  });  }  viewclass(data) {  **this**.\_user.setSelectedClass(data);  }  sortClass(payload): any {  **let** arrDays: **string**[] = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'];  **let** tmp: **any** = payload;  **let** items: **any**[]=[];  arrDays.forEach(x => {  tmp.forEach(y => {  **let** found=**0**;  **if**(y.day\_fld.includes(x)) {  items.forEach(z=>{  **if**(y.classcode\_fld==z.classcode\_fld) found = **1**;  });  **if**(found!=**1**) items.push(y);  }  });  });  **return** items  }  filterClass(ev) {  **let** searchItem: **string** = ev.target.value;  **this**.classes = **this**.allClasses.filter((x)=>{  **return** (  x.classcode\_fld.toUpperCase().includes(searchItem.toUpperCase()) ||  x.subjcode\_fld.toUpperCase().includes(searchItem.toUpperCase()) ||  x.subjdesc\_fld.toUpperCase().includes(searchItem.toUpperCase())  );  });  }  openDialog() {  **let** data = {  "data": **this**.allClasses,  };  **const** dialogRef = **this**.dialog.open(ViewScheduleModalComponent, {  panelClass: 'dialogpadding',  maxWidth: '90vw',  maxHeight: '90vh',  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('100%', 'auto');  } **else** {  dialogRef.updateSize('100%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  });  }  getDays(vDays): **string** {  **let** tmp: **string** = '';  **if**(vDays.includes('Mon')) { tmp+='M'; }  **if**(vDays.includes('Tue')) { tmp+='T'; }  **if**(vDays.includes('Wed')) { tmp+='W'; }  **if**(vDays.includes('Thu')) { tmp+='Th'; }  **if**(vDays.includes('Fri')) { tmp+='F'; }  **if**(vDays.includes('Sat')) { tmp+='Sat'; }  **if**(vDays.includes('Sun')) { tmp+='Sun'; }  **return** tmp;  }  } |

**CLASSROOM**

|  |
| --- |
| **export** **class** ClassroomComponent **implements** OnInit {  isWideScreen$: **Observable**<**boolean**> | **undefined**;  showtab: **boolean** = **false**;  instructor: **any** = [];  students: **any** = [];  **constructor**(**private** \_location: **Location**, **private** \_ds: **DataService**, **public** \_user: **UserService**, **private** route: **Router**) { }  ngOnInit(): **void** {  **this**.InitializeClassmembers()  **this**.getUnread()  }  *// ngOnDestroy(): void {*  *// window.sessionStorage.removeItem(btoa('teacher'));*  *// window.sessionStorage.removeItem(btoa('students'));*  *// }*  InitializeClassmembers(){  **let** load = {  data:{  classcode:**this.\_user.getSelectedClass**().classcode\_fld,  acadyear:**this.\_user.getAcadYear**(),  semester:**this.\_user.getSemester**()  }  }  **this**.\_ds.\_httpRequest('getmembers',load,**1**).subscribe((data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a);      **if** (data.payload != **null**) {  **this**.instructor = data.payload.instructor;  **this**.\_user.setTeachers(**this**.instructor);  **this**.students = data.payload.student;  **this**.\_user.setStudents(**this**.students);  }  },er => {  er = **this**.\_user.\_decrypt(er.error.a);    })  }  unreadCount = **null**  getUnread() {  **this**.\_ds.\_httpRequest('getunread', { roomcode: **this.route.url.split**('/')[**3**], recipient: **this.\_user.getUserID**() }, **1**).subscribe(async(res:**any**) => {  **const** data = await **this**.\_user.\_decrypt(res.a);  **this**.unreadCount = data.payload.length;    **this**.\_user.setMessageBadge(**this**.unreadCount)  }, er => {  er = **this**.\_user.\_decrypt(er.error.a)  **this**.\_user.setMessageBadge('')  })  }  } |

**CLASS SCHEDULE**

|  |
| --- |
| **export** **class** ViewScheduleModalComponent **implements** OnInit {  *// Schedules*  m = [];  t = [];  w = [];  th = [];  f = [];  sat = [];  sun = [];  index = **1**;  isLoading: **boolean** = **true**;  schedule: **any** = [];  **constructor**(**public** \_ds: **DataService**, **private** \_user: **UserService**, @Inject(MAT\_DIALOG\_DATA) **public** data: **any**) { }  ngOnInit(): **void** {  **this**.schedule = **this**.data.data;  **this**.setSchedule();    }  convertTime12to24 = (time12h) => {  **const** [time, modifier] = time12h.split(' ');  **let** [hours, minutes] = time.split(':');  **if** (hours === '12') {  hours = '00';  }  **if** (modifier === 'PM') {  hours = parseInt(hours, **10**) + **12**;  }  **if** (hours.length == **1**) {  hours = '0'.concat(hours);  }  **return** `${hours}:${minutes}`;  }  sortField(field, isOrdered) {  **return** (a, b) => {  **if** (isOrdered) {  **return** a[field].toString().localeCompare(b[field].toString());  }  **return** b[field].toString().localeCompare(a[field].toString());  };  }  setSchedule(){  **for** (**let** i = **0**; i < **this**.schedule.length; i++) {  **let** sched = **this**.schedule[i].day\_fld;  **if** (sched.match('Mon')) {  **let** time = **this**.convertTime12to24(**this**.schedule[i].starttime\_fld);  **let** mondaySched = {  'id': i + **this**.index,  'subject': **this**.schedule[i].subjdesc\_fld,  'startTime': **this**.schedule[i].starttime\_fld,  'endTime': **this**.schedule[i].endtime\_fld,  'date': '2016-12-07T' + time,  'SubjectCode': **this**.schedule[i].subjcode\_fld  }  **this**.m.push(mondaySched);  }  **if** (sched.match('Tue')) {  **let** time = **this**.convertTime12to24(**this**.schedule[i].starttime\_fld);  **let** mondaySched = {  'id': i + **this**.index,  'subject': **this**.schedule[i].subjdesc\_fld,  'startTime': **this**.schedule[i].starttime\_fld,  'endTime': **this**.schedule[i].endtime\_fld,  'date': '2016-12-07T' + time,  'SubjectCode': **this**.schedule[i].subjcode\_fld  }  **this**.t.push(mondaySched);  }  **if** (sched.match('Wed')) {  **let** time = **this**.convertTime12to24(**this**.schedule[i].starttime\_fld);  **let** mondaySched = {  'id': i + **this**.index,  'subject': **this**.schedule[i].subjdesc\_fld,  'startTime': **this**.schedule[i].starttime\_fld,  'endTime': **this**.schedule[i].endtime\_fld,  'date': '2016-12-07T' + time,  'SubjectCode': **this**.schedule[i].subjcode\_fld  }  **this**.w.push(mondaySched);  }  **if** (sched.match('Thu')) {  **let** time = **this**.convertTime12to24(**this**.schedule[i].starttime\_fld);  **let** mondaySched = {  'id': i + **this**.index,  'subject': **this**.schedule[i].subjdesc\_fld,  'startTime': **this**.schedule[i].starttime\_fld,  'endTime': **this**.schedule[i].endtime\_fld,  'date': '2016-12-07T' + time,  'SubjectCode': **this**.schedule[i].subjcode\_fld  }  **this**.th.push(mondaySched);  }  **if** (sched.match('Fri')) {  **let** time = **this**.convertTime12to24(**this**.schedule[i].starttime\_fld);  **let** mondaySched = {  'id': i + **this**.index,  'subject': **this**.schedule[i].subjdesc\_fld,  'startTime': **this**.schedule[i].starttime\_fld,  'endTime': **this**.schedule[i].endtime\_fld,  'date': '2016-12-07T' + time,  'SubjectCode': **this**.schedule[i].subjcode\_fld  }  **this**.f.push(mondaySched);  }  **if** (sched.match('Sat')) {  **let** time = **this**.convertTime12to24(**this**.schedule[i].starttime\_fld);  **let** mondaySched = {  'id': i + **this**.index,  'subject': **this**.schedule[i].subjdesc\_fld,  'startTime': **this**.schedule[i].starttime\_fld,  'endTime': **this**.schedule[i].endtime\_fld,  'date': '2016-12-07T' + time,  'SubjectCode': **this**.schedule[i].subjcode\_fld  }  **this**.sat.push(mondaySched);  }  **if** (sched.match('Sun')) {  **let** time = **this**.convertTime12to24(**this**.schedule[i].starttime\_fld);  **let** mondaySched = {  'id': i + **this**.index,  'subject': **this**.schedule[i].subjdesc\_fld,  'startTime': **this**.schedule[i].starttime\_fld,  'endTime': **this**.schedule[i].endtime\_fld,  'date': '2016-12-07T' + time,  'SubjectCode': **this**.schedule[i].subjcode\_fld  }  **this**.sun.push(mondaySched);  }  **this**.m.sort(**this**.sortField('date', **true**));  **this**.t.sort(**this**.sortField('date', **true**));  **this**.w.sort(**this**.sortField('date', **true**));  **this**.th.sort(**this**.sortField('date', **true**));  **this**.f.sort(**this**.sortField('date', **true**));  **this**.sat.sort(**this**.sortField('date', **true**));  **this**.sun.sort(**this**.sortField('date', **true**));  }  }  } |

**CLASSFEED**

|  |
| --- |
| **xport** **class** ClassFeedComponent **implements** OnInit {  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(  Breakpoints.XSmall  );  classobject: **any**;  globalvar: **any**;  subjcode: **any**;  postobject: **any** = [];  pageslice: **any** = [];  resources: **any**[] = [];  teacherlistobject: **any**;  instructor: **any**;  students: **any**;  activities: **any**[] = [];  files: **any** = [];  fileButton: **boolean** = **false**;  photoButton: **boolean** = **false**;  videoButton: **boolean** = **false**;  openButton: **number** = **0**;  withfile: **number**;  isUploading: **boolean**;  filepathtoupload: **any**;  textcontent: **string** = '';  datatoedit: **any**;  filepath: **FormData**;  act: **any**[] = [];  acceptedFiles =  '.doc,.docx,.html,.htm,.odt,.pdf,.xls,.xlsx,.ods,.ppt,.pptx,.txt';  commentcount: **number**;  activityObject: **any** = [];  **constructor**(  **public** \_user: **UserService**,  **public** \_ds: **DataService**,  **public** uploadservice: **UploadsService**,  **public** dialog: **MatDialog**,  **private** breakpointObserver: **BreakpointObserver**,  **private** sanitizer: **DomSanitizer**,  **private** \_snackbar: **MatSnackBar**,  **public** \_upload: **UploadsService**,  **public** \_commonservice: **CommonService**,  **public** \_quest: **QuestService**  ) { }  ngOnInit(): **void** {  **this**.getClasspost();  **this**.getActivities();  **this**.getResource();  }  ngAfterViewInit(): **void** {  console.clear();  }  getFileExtension(filename) {  **if** (filename == 'pdf') {  **return** '#EA462E';  } **else** **if** (filename == 'docx') {  **return** '#2D5292';  } **else** **if** (filename == 'ppt') {  **return** '#CA4223';  } **else** **if** (filename == 'zip') {  **return** '#B23333';  } **else** **if** (filename == 'txt') {  **return** '#546A7A';  } **else** {  **return** '#222';  }  }  getfileExt(filename) {  **return** filename.split('.').pop();  }  getClasspost() {  **this**.\_ds  .\_httpRequest(  'getcpost',  {  classcode: **this.\_user.getSelectedClass**().classcode\_fld,  type: 'post',  },  **1**  )  .subscribe(  (data: **any**) => {  data = **this**.\_user.\_decrypt(data.a);  **this**.postobject = data.payload;  **if** (**this**.postobject != **null**) {  **this**.postobject.sort((a, b) => {  **return** (  <any>**new** Date(b.timestamp\_fld) - <any>**new** Date(a.timestamp\_fld)  );  });  **this**.pageslice = **this**.addVideoUrl(**this**.postobject);  **this**.pageslice = **this**.addfiles(**this**.postobject);  }  },  (er) => {  er = **this**.\_user.\_decrypt(er.error.a);  **this**.postobject = [];  }  );  }  getActivities() {  **let** load = {  classcode: **this.\_user.getSelectedClass**().classcode\_fld,  type: '',  };  **this**.\_ds.\_httpRequest('getcpost', load, **1**).subscribe(  (data: **any**) => {  data = **this**.\_user.\_decrypt(data.a);  **this**.activities = data.payload;  **for** (**let** i = **0**; i < **this**.activities.length; i++) {  **if** (  **this**.activities[i].type\_fld === **0** ||  (**this**.activities[i].type\_fld === **1** &&  **this**.activities[i].datesched\_fld < **this**.\_user.getDateToday())  ) {  **this**.activities[i].openQuiz = **true**;  } **else** {  **this**.activities[i].openQuiz = **false**;  }  }  },  (er) => {  er = **this**.\_user.\_decrypt(er.error.a);  }  );  }  getResource() {  **this**.\_ds  .\_httpRequest(  'getreslist',  { classcode: **this.\_user.getSelectedClass**().classcode\_fld },  **1**  )  .subscribe(  (data: **any**) => {  data = **this**.\_user.\_decrypt(data.a);  **this**.resources = **this**.splitLinkResource(data.payload);  **this**.\_user.setResources(**this**.resources);  },  (er) => {  er = **this**.\_user.\_decrypt(er.error.a);  }  );  }  *// Fetching Data End*  *// Start of file uploading*  async submitPost() {  **this**.\_user.setLoading(**true**);  **this**.fileButton = **true**;  **this**.photoButton = **true**;  **this**.videoButton = **true**;  await **this**.\_upload  .Filepath(**this**.files)  .then((filepath: **any**) => {  **this**.filepathtoupload = filepath;  })  .**catch**((e) => {  **return**;  });  **let** load = {  data: {  classcode\_fld: **this.\_user.getSelectedClass**().classcode\_fld,  authorid\_fld: **this.\_user.getUserID**(),  content\_fld: **this.textcontent**,  withfile\_fld: **this.withfile**,  dir\_fld: **this.filepathtoupload**,  },  notif: {  id: **this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message:  **this.\_user.getFullname**() +  ' created a post on ' +  **this**.\_user.getSelectedClass().subjdesc\_fld,  module: 'classroom',  },  };  **this**.\_ds.\_httpRequest('addpost', load, **1**).subscribe(  (data: **any**) => {  data = **this**.\_user.\_decrypt(data.a);  **this**.postobject.unshift(data.payload[**0**]);  **this**.pageslice = **this**.addVideoUrl(**this**.postobject);  **this**.pageslice = **this**.addfiles(**this**.postobject);  **this**.textcontent = '';  **this**.files = [];  **this**.filepathtoupload = '';  **this**.withfile = **0**;  **this**.fileButton = **false**;  **this**.photoButton = **false**;  **this**.videoButton = **false**;  **this**.\_user.setLoading(**false**);  **this**.\_snackbar.open('Post added.', **null**, { duration: **1500** });  *//Update Quest User Data*  **this**.\_quest.UpdateCountAchievement(Crud.ADD, fldNames.piccnt\_fld);  },  (er) => {  er = **this**.\_user.\_decrypt(er.error.a);  **this**.\_user.setLoading(**false**);  **this**.\_snackbar.open(er.status.message, **null**, { duration: **1500** });  }  );  }  async getFile(event) {  **let** sum: **number** = **0**;  **for** (**let** i = **0**; i < event.target.files.length; i++) {  sum += event.target.files[i].size;  **this**.files.push(event.target.files[i]);  **let** ext = **this**.getext(**this**.files[i].name);  **switch** (ext) {  **case** 'txt':  **case** 'doc':  **case** 'docx':  **case** 'html':  **case** 'htm':  **case** 'odt':  **case** 'pdf':  **case** 'xls':  **case** 'xlsx':  **case** 'ods':  **case** 'pdf':  **case** 'ppt':  **case** 'pptx':  **this**.fileButton = **false**;  **this**.photoButton = **true**;  **this**.videoButton = **true**;  **this**.withfile = **5**;  **break**;  **case** 'm4v':  **case** 'avi':  **case** 'mpg':  **case** 'mp4':  **this**.fileButton = **true**;  **this**.photoButton = **true**;  **this**.videoButton = **false**;  **this**.withfile = **2**;  **break**;  **case** 'jpg':  **case** 'gif':  **case** 'bmp':  **case** 'png':  **case** 'jpeg':  **this**.fileButton = **true**;  **this**.photoButton = **false**;  **this**.videoButton = **true**;  **this**.withfile = **1**;  **break**;  }  **this**.\_upload.getFile(**this**.files);  }  **if** (sum > **30000000**) {  **this**.\_snackbar.open(  'Your total upload file(S) is ' +  (sum / **1048576**).toFixed(**2**).toString() +  'MB. Please upload only atleast 30mb of file(s)',  '',  { duration: **1500** }  );  **this**.fileButton = **false**;  **this**.photoButton = **false**;  **this**.videoButton = **false**;  **this**.files = [];  (<HTMLInputElement>document.getElementById('files')).value = '';  } **else** {  (<HTMLInputElement>document.getElementById('files')).value = '';  }  **this**.files = [];  }  async removeFile(i) {  **this**.files.splice(i, **1**);  **if** (**this**.files.length == **0**) {  **this**.fileButton = **false**;  **this**.photoButton = **false**;  **this**.videoButton = **false**;  **this**.filepathtoupload = '';  **this**.withfile = **0**;  } **else** {  await **this**.\_upload.Filepath(**this**.files).then((filepath: **any**) => {  **this**.filepathtoupload = filepath;  });  }  }  *// End of file uploading*  *// Delete Posts*  delpost(i, postcode, option) {  **let** condel;  **const** dialogConfig = **new** MatDialogConfig();  dialogConfig.panelClass = 'dialogpadding';  dialogConfig.autoFocus = **true**;  **if** (**this**.\_user.isMobile()) {  dialogConfig.maxHeight = '90vw';  dialogConfig.minHeight = 'auto';  dialogConfig.minWidth = '70vw';  dialogConfig.maxWidth = '90vw';  } **else** {  dialogConfig.minHeight = '20vh';  dialogConfig.minWidth = '20vw';  }  dialogConfig.data = {  option: **option**,  isConfirmed: **condel**,  data: **postcode**,  };  **const** dialogRef = **this**.dialog.open(DialogComponent, dialogConfig);  dialogRef.afterClosed().subscribe((result) => {  **if** (result.data) {  **this**.pageslice.splice(i, **1**);  **let** load = {  data: {  classcode\_fld: **this.\_user.getSelectedClass**().classcode\_fld,  authorid\_fld: **this.\_user.getUserID**(),  isdeleted\_fld: **1**,  },  notif: {  id: **this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message:  **this.\_user.getFullname**() +  ' deleted a post in ' +  **this**.\_user.getSelectedClass().classcode\_fld,  module: 'classroom',  },  };  **this**.\_ds.\_httpRequest('editpost/' + postcode, load, **1**).subscribe(  (res) => {  res = **this**.\_user.\_decrypt(res.a);  },  (er) => {  **let** err = **this**.\_user.\_decrypt(er.error.a);  }  );  }  });  }  *// Edit Posts*  editPost(index, postcode, load, option) {  **let** conedit;  **let** data = {  option: **option**,  isConfirmed: **conedit**,  data: **load**,  };  **const** dialogRef = **this**.dialog.open(PostsEditComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**,  autoFocus: **true**,  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(  (result) => {  **if** (result.matches) {  dialogRef.updateSize('95%', 'auto');  } **else** {  dialogRef.updateSize('50%', 'auto');  }  }  );  dialogRef.afterClosed().subscribe((result) => {  responsiveDialogSubscription.unsubscribe();  **if** (result.data) {  **let** load = result.data;  **this**.\_ds.\_httpRequest('editpost/' + postcode, load, **1**).subscribe(  (res) => {  res = **this**.\_user.\_decrypt(res.a);  **if** (res.status.remarks == 'success') {  **this**.postobject[index] = res.payload[**0**];  **this**.postobject.sort((a, b) => {  **return** (  <any>**new** Date(b.timestamp\_fld) -  <any>**new** Date(a.timestamp\_fld)  );  });  **this**.pageslice = **this**.addVideoUrl(**this**.postobject);  **this**.pageslice = **this**.addfiles(**this**.postobject);  **this**.\_snackbar.open('Edited successfully.', **null**, {  duration: **1500**,  });  } **else** {  **this**.pageslice = **null**;  }  },  (er) => {  **let** err = **this**.\_user.\_decrypt(er.error.a);  **if** (err.payload == **null**) {  **this**.getClasspost();  }  }  );  }  });  }  setAnswers(post, j, i) {  **let** load = {  data: {  postcode\_fld: **post.postcode\_fld**,  studnum\_fld: **this.\_user.getUserID**(),  response\_fld: **j**,  respcode\_fld: **post.respcode\_fld**,  },  filepath: {  options: **post.pollresponse\_fld**,  },  notif: {  id: **this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message:  **this.\_user.getFullname**() +  ' reponded in a poll on ' +  **this**.\_user.getSelectedClass().subjdesc\_fld,  module: 'classroom',  },  };  *// this.\_commonservice.commonSubscribe('answerpoll', load, 1).then((dt: any) => {*  *// this.getClasspost();*  *// });*  **this**.\_ds.\_httpRequest('answerpoll', load, **1**).subscribe(  (data: **any**) => {  data = **this**.\_user.\_decrypt(data.a);  **this**.postobject[i].pollresponse\_fld.map((poll: **any**) => {  poll.userpollvoted = **false**;  poll.respondents.map((e, indx) => {  **if** (e.studnum\_fld === **this**.\_user.getUserID()) {  poll.respondents.splice(indx, **1**);  poll.votes -= **1**;  }  });  });  **this**.postobject[i].pollresponse\_fld[j].userpollvoted = **true**;  **this**.postobject[i].pollresponse\_fld[j].votes += **1**;  **this**.postobject[i].pollresponse\_fld[j].respondents.push({  fullname: **this.\_user.getFullname**(),  profilepic\_fld: **this.\_user.getProfilePic**(),  studnum\_fld: **this.\_user.getUserID**(),  });  },  (er) => {  er = **this**.\_user.\_decrypt(er.error.a);  }  );  }  addfiles(array) {  array.forEach(async (e) => {  e.images\_fld = [];  e.uploadvideo\_fld = [];  e.attachment\_fld = [];  e.poll\_fld = [];  e.pollchoices\_fld = [];  e.pollresponse\_fld = [];  e.respcode\_fld = '';  e.authorid\_fld == **this**.\_user.getUserID()  ? (e.author\_owner = **true**)  : (e.author\_owner = **false**);  **if** (e.dir\_fld) {  **let** files = **this**.\_upload.splitFilestring(e.dir\_fld);  files.forEach((f) => {  **if** (  f.path.includes('gif') ||  f.path.includes('jpeg') ||  f.path.includes('jpg') ||  f.path.includes('png') ||  f.path.includes('bmp')  ) {  e.images\_fld.push(f);  }  **if** (  f.path.includes('mp4') ||  f.path.includes('m4v') ||  f.path.includes('avi') ||  f.path.includes('mpg') ||  f.path.includes('wmv')  ) {  e.uploadvideo\_fld.push(f);  }  **if** (  f.path.includes('doc') ||  f.path.includes('docx') ||  f.path.includes('html') ||  f.path.includes('htm') ||  f.path.includes('pdf') ||  f.path.includes('xls') ||  f.path.includes('xlsx') ||  f.path.includes('ods') ||  f.path.includes('csv') ||  f.path.includes('ppt') ||  f.path.includes('pptx') ||  f.path.includes('txt')  ) {  e.attachment\_fld.push(f);  }  });  }  **if** (  e.content\_fld.includes('youtube.com') ||  e.content\_fld.includes('google.com/file/d/')  ) {  e.embedvideo\_fld = **this**.getVideoEmbedURL(e.content\_fld);  }  **if** (e.withfile\_fld == **3** || e.withfile\_fld == **4**) {  **let** pollObj = await **this**.\_upload.splitPollContent(e);  e.content\_fld = pollObj.content\_fld;  e.pollchoices\_fld = pollObj.pollchoices\_fld;  **let** load = {  postcode: **e.postcode\_fld**,  filepath: {  options: [],  },  };  e.pollchoices\_fld.map((chc) => {  load.filepath.options.push({  choice: **chc**,  votes: **0**,  respondents: [],  });  });  **this**.\_commonservice  .commonSubscribe('getpoll', load, **1**)  .then(async (dt: **any**) => {  e.pollresponse\_fld = await dt.payload.options;  e.respcode\_fld = dt.payload.respcode;  e.pollresponse\_fld.map((poll) => {  poll.respondents.map((e) => {  **if** (e.studnum\_fld === **this**.\_user.getUserID()) {  poll.userpollvoted = **true**;  }  });  });  });  }  });  **return** array;  }  getFilename(name) {  **return** name.replace(/^.\*[\\\/]/, '');  }  *// getFileExtension(filename) { return filename.split('.').pop(); }*  getfileExtension(filename) {  **let** regex = **new** RegExp('[^.]+$');  *// return filename.match(regex);*  **if** (filename.match(regex) == 'docx') {  **return** '#004db3';  }  **if** (filename.match(regex) == 'ppt') {  **return** '#c34524';  }  **if** (filename.match(regex) == 'txt') {  **return** '#95a5a6';  }  **if** (filename.match(regex) == 'pdf') {  **return** '#ad0b00';  } **else** {  **return** '#1f6499';  }  }  getext(data) {  **return** data.split('.').pop();  }  addVideoUrl(array) {  **for** (**let** i = **0**; i < array.length; i++) {  **if** (  array[i].content\_fld.includes('youtube.com') ||  array[i].content\_fld.includes('google.com/file/d/')  ) {  array[i].video\_fld = **this**.getVideoEmbedURL(array[i].content\_fld);  } **else** {  array[i].video\_fld = **null**;  }  }  **return** array;  }  *// Start of opening the dialog for preview docx,xlsx,pptx*  previewResource(link, name) {  **let** data = {  resourceString: **link**,  reourcename: **name**,  };  **const** dialogRef = **this**.dialog.open(ResourcePreviewComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  disableClose: **true**,  data: **data**,  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(  (result) => {  **if** (result.matches) {  dialogRef.updateSize('100%', '100%');  } **else** {  dialogRef.updateSize('100%', '100%');  }  }  );  dialogRef.afterClosed().subscribe((result) => {  responsiveDialogSubscription.unsubscribe();  });  }  *// End of opening the dialog for preview docx,xlsx,pptx*  *// Addedd by Melner*  getVideoEmbedURL(content: **string**): any {  **let** replacePattern1 =  /(\b(https?|ftp):\/\/[-A-Z0-9+&@#\/%?=~\_|!:,.;]\*[-A-Z0-9+&@#\/%=~\_|])/gim;  **let** replacedText = content.replace(replacePattern1, '<\*>$1<\*>');  **let** content\_arr: **string**[] = replacedText.split('<\*>');  **let** a: **any** = content\_arr.filter((x) => {  **if** (x.includes('http')) {  **return** x;  }  });  **for** (**let** i = **0**; i < a.length; i++) {  **if** (a[i].includes('youtube.com')) {  a[i] = a[i].replace('watch?v=', 'embed/');  } **else** {  a[i] = a[i].replace('view?usp=sharing', 'preview');  }  a[i] = **this**.sanitizer.bypassSecurityTrustResourceUrl(a[i]);  }  **return** a;  }  imgpreview(img) {  **let** data = img;  **this**.dialog.open(ImagePreviewComponent, {  panelClass: 'dialogpadding',  data: **data**,  });  }  openComments(acode, ccnt, i) {  **let** data = {  type: 'post-comments',  acode: **acode**,  commentcnt: **ccnt**,  };  **const** dialogRef = **this**.dialog.open(PostCommentsComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**,  disableClose: **true**,  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(  (result) => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('45%', 'auto');  }  }  );  dialogRef.afterClosed().subscribe((result) => {  responsiveDialogSubscription.unsubscribe();  **this**.pageslice[i].commentcount = result;  });  *// const dialogConfig = new MatDialogConfig();*  *// dialogConfig.autoFocus = true;*  *// dialogConfig.disableClose = true;*  *// if (this.\_user.isMobile()) {*  *// dialogConfig.minHeight = 'auto';*  *// dialogConfig.maxHeight = '90vh';*  *// dialogConfig.minWidth = '90vw';*  *// } else {*  *// dialogConfig.minHeight = 'auto';*  *// dialogConfig.maxWidth = '40vw';*  *// dialogConfig.maxHeight = '80vh';*  *// }*  *// dialogConfig.data = {*  *// type: 'post-comments',*  *// acode: acode,*  *// commentcnt: ccnt*  *// }*  *// const dialogRef = this.dialog.open(PostCommentsComponent, dialogConfig);*  *// const responsiveDialogSubscription = this.isExtraSmall.subscribe(result => {*  *// if (result.matches) {*  *// dialogRef.updateSize('98%', 'auto');*  *// } else {*  *// dialogRef.updateSize('50%', 'auto');*  *// }*  *// });*  *// dialogRef.afterClosed().subscribe(result => {*  *// this.pageslice[i].commentcount = result;*  *// responsiveDialogSubscription.unsubscribe();*  *// });*  }  splitLinkResource(data: **any**) {  data.forEach(element => {  **if** (!element.filedir\_fld.includes("%=!!=%")) {  element.filedir\_fld = **this**.uploadservice.splitFilestring(element.filedir\_fld);  } **else** {  **let** temp = element.filedir\_fld.split("%=!!=%");  element.filedir\_fld = **this**.uploadservice.splitFilestring(temp[**0**]);  element.link\_fld = temp[**1**].split("//\*//");  }  });  **return** data;  }  } |

**CLASSFEED POST EDIT**

|  |
| --- |
| **export** **class** PostCommentsComponent **implements** OnInit {  postcomments: **any** = [];  count: **any**;  replystatus:**number** = **0**;  editstatus:**number** = **0**;  recipientcode: **any**;  commentcode: **any**;  commentcontent: **any**;  i: **any**;  replyobject: **any** = [];  commentcodeforreply: **any**;  showLoader : **boolean** = **true**;  **constructor**(**public** \_upload:**UploadsService**,  **public** \_user:**UserService**,  **public** \_ds:**DataService**,@Inject(MAT\_DIALOG\_DATA) **public** data: **any**,  **private** dialogReg: **MatDialogRef**<DialogComponent>,  **public** dialog: **MatDialog**,  **public** \_quest:**QuestService**  ) { }  ngOnInit(): **void** {  **this**.getcomments();  **this**.count = **this**.data.commentcnt;  }  getcomments(){  **this**.\_user.setLoading(**true**);  **this**.\_ds.\_httpRequest('getccomment',{acode:**this.data.acode**},**1**).subscribe((data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a);  **this**.postcomments = data.payload;  **this**.\_user.setLoading(**false**);  *// this.showLoader = false;*  *// console.log(this.replyobject);*  },er=>{  er = **this**.\_user.\_decrypt(er.error.a);  **this**.\_user.setLoading(**false**);  **this**.postcomments = [];  **this**.count = **0**;  })  }    addComment(e){  e.preventDefault();    **let** content = e.target[**0**].value;  **if** (content == '' || content == **undefined** || content == **null**) {  }**else**{  **let** load = {  data:{  classcode\_fld:**this.\_user.getSelectedClass**().classcode\_fld,  authorid\_fld : **this.\_user.getUserID**(),  actioncode\_fld:**this.data.acode**,  content\_fld : **content**,  withfile\_fld: **0**,  dir\_fld : ''  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message: **this.\_user.getFullname**()+' added a comment on '+**this**.\_user.getSelectedClass().subjdesc\_fld,  module: 'classroom'  }    }  *// console.log(load);*    **this**.\_ds.\_httpRequest('addcomment',load,**1**).subscribe((data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a);  *// this.getcomments();*  e.target[**0**].value = **null**;  **this**.postcomments.push(data.payload[**0**]);  **this**.count = **this**.count + **1**;  *//Update Quest User Data*  **this**.\_quest.UpdateCountAchievement(Crud.ADD,fldNames.ciccnt\_fld);  *// console.log(data.payload);*  },er=>{  er = **this**.\_user.\_decrypt(er.error.a);  *// console.log(er);*    })  }  }  addReply(e,comments,i){  e.preventDefault();    **let** replycontent = e.target[**0**].value;  *// console.log(replycontent);*    **if** (replycontent == '' || replycontent == **undefined** || replycontent == **null**) {  *// console.log("put replycontent");*  }**else**{  **let** load = {  data:{  classcode\_fld:**this.\_user.getSelectedClass**().classcode\_fld,  authorid\_fld : **this.\_user.getUserID**(),  actioncode\_fld:**comments.commentcode\_fld**,  content\_fld : **replycontent**,  withfile\_fld: **0**,  dir\_fld : ''  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message: **this.\_user.getFullname**()+' replied to '+ comments.commentcode\_fld,  module: 'classroom'  }  }  *// console.log(load);*    **this**.\_ds.\_httpRequest('addcomment',load,**1**).subscribe((data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a);  **if** (**this**.postcomments[i].reply == **null**) {  **this**.postcomments[i].reply = [];  **this**.postcomments[i].reply.push(data.payload[**0**]);  } **else** {  **this**.postcomments[i].reply.push(data.payload[**0**]);  }  e.target[**0**].value = **null**;  **this**.replystatus = **0**;  **this**.editstatus = **0**;  *// this.postcomments.unshift(this.\_user.\_convert(data.payload[0]));*  },er=>{  er = **this**.\_user.\_decrypt(er.error.a);    })  }    }  showReply(i){  **var** x = document.getElementById('replydiv'+i);  x.style.display === "none" ? x.style.display = "block" : x.style.display = "none";  **let** j = document.getElementById('replybtn' + i);  j.style.display === "none" ? j.style.display = "block" : j.style.display = "none";  }  *// reply(authorid\_fld,actioncode\_fld){*  *// this.replystatus = 1;*  *// this.editstatus = 4;*  *// this.recipientcode = actioncode\_fld;*  *// }*  cancelreply(){  **this**.replystatus = **0**;  }  edit(index,code,content){    **this**.commentcode = code;  **this**.i = index;  (<HTMLInputElement>document.getElementById('content\_fld' + index)).value = content;  **let** editbox = document.getElementById('edit-box' + index);  **let** comments = document.getElementById('comments' + index);  comments.style.display === "none" ? comments.style.display = "block" : comments.style.display = "none";    editbox.style.display === "none" ? editbox.style.display = "block" : editbox.style.display = "none";    }  editComment(e){  e.preventDefault();  **let** commentcontent = e.target[**0**].value;  **if** (commentcontent == '' || commentcontent == **undefined** || commentcontent == **null**) {  *// console.log("put commentcontent");*  }**else**{  **let** load = {  data:{  classcode\_fld:**this.\_user.getSelectedClass**().classcode\_fld,  authorid\_fld:**this.\_user.getUserID**(),  content\_fld:**commentcontent**  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message: **this.\_user.getFullname**()+' edited a comment '+**this**.\_user.getSelectedClass().classcode\_fld,  module: 'classroom'  }    }  **this**.\_ds.\_httpRequest('editcomm/'+**this**.commentcode, load , **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a);  *// console.log(res);*    **this**.postcomments[**this**.i] = res.payload[**0**];  *// this.editstatus = 0;*  *// this.replystatus = 0;*  }, er => {  **let** err = **this**.\_user.\_decrypt(er.error.a);  *// console.log(err);*  })  }  }  cancelEdit(i){  **let** editbox = document.getElementById('edit-box' + i);  **let** comments = document.getElementById('comments' + i);  **let** img = document.getElementById('img-container' + i);  editbox.style.display === "flex" ? editbox.style.display = "none" : editbox.style.display = "none";  comments.style.display = 'block';  img.style.display = 'block';  }  deletecomment(index,commentcode , option){  **let** condel  **const** dialogConfig = **new** MatDialogConfig();  dialogConfig.autoFocus = **true**;  **if** (**this**.\_user.isMobile()) {  dialogConfig.maxHeight = '90vw';  dialogConfig.minHeight = 'auto';  dialogConfig.maxWidth = '70vw';  dialogConfig.maxWidth = '90vw';  } **else** {  dialogConfig.minHeight = '20vh';  dialogConfig.minWidth = '15vw';  }  dialogConfig.data = {  option:**option**,  isConfirmed:**condel**  }  **const** dialogRef = **this**.dialog.open(DialogComponent, dialogConfig);  dialogRef.afterClosed().subscribe(result => {  **if** (result.data) {  **let** load = {  data:{  classcode\_fld:**this.\_user.getSelectedClass**().classcode\_fld,  authorid\_fld:**this.\_user.getUserID**(),  isdeleted\_fld:**1**  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message: **this.\_user.getFullname**()+' deleted a comment '+**this**.\_user.getSelectedClass().classcode\_fld,  module: 'classroom'  }    }  **this**.\_ds.\_httpRequest('editcomm/'+commentcode, load , **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a);  *// console.log(res);*  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  **this**.postcomments.splice(index,**1**);  **this**.count = **this**.count - **1**;  })  }  });  }  deletereply(index,j,commentcode , option){  **let** condel  **const** dialogConfig = **new** MatDialogConfig();  dialogConfig.autoFocus = **true**;  **if** (**this**.\_user.isMobile()) {  dialogConfig.maxHeight = '90vw';  dialogConfig.minHeight = 'auto';  dialogConfig.minWidth = '70vw';  dialogConfig.maxWidth = '90vw';  } **else** {  dialogConfig.minHeight = '20vh';  dialogConfig.minWidth = '15vw';  }  dialogConfig.data = {  option:**option**,  isConfirmed:**condel**  }  **const** dialogRef = **this**.dialog.open(DialogComponent, dialogConfig);  dialogRef.afterClosed().subscribe(result => {  *// console.log(result.data);*    **if** (result.data) {  *// console.log(commentcode);*    **let** load = {  data:{  classcode\_fld:**this.\_user.getSelectedClass**().classcode\_fld,  authorid\_fld:**this.\_user.getUserID**(),  isdeleted\_fld:**1**  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message: **this.\_user.getFullname**()+' deleted a comment '+**this**.\_user.getSelectedClass().classcode\_fld,  module: 'classroom'  }    }  **this**.\_ds.\_httpRequest('editcomm/'+commentcode, load , **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a);  *// console.log(res);*  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  **this**.postcomments[index].reply.splice(j,**1**);    })  }  });  }  editthis(index,reply,j){  console.log(index,reply,j);    *// console.log(reply.content\_fld);*  **this**.commentcode = reply.commentcode\_fld;  **this**.commentcontent = reply.content\_fld;  **this**.i = j;  (<HTMLInputElement>document.getElementById(index + 'reply\_content\_fld' + j)).value = **this**.commentcontent;  **let** reditbox = document.getElementById(index + 'redit-box' + j);    reditbox.style.display === "none" ? reditbox.style.display = "flex" : reditbox.style.display = "none";    *// this.editstatus = 3;*    }  editReply(e,comments,i){  e.preventDefault();  **let** commentcontent = e.target[**0**].value;  **if** (commentcontent == '' || commentcontent == **undefined** || commentcontent == **null**) {  *// console.log("put commentcontent");*  }**else**{  **let** load = {  data:{  classcode\_fld:**this.\_user.getSelectedClass**().classcode\_fld,  authorid\_fld:**this.\_user.getUserID**(),  content\_fld:**commentcontent**  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message: **this.\_user.getFullname**()+' edited a reply '+**this**.\_user.getSelectedClass().classcode\_fld,  module: 'classroom'  }    }  **this**.\_ds.\_httpRequest('editcomm/'+comments.commentcode\_fld, load , **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a);  **this**.postcomments[i].reply[**this**.i] = res.payload[**0**];  **this**.editstatus = **0**;  }, er => {  **let** err = **this**.\_user.\_decrypt(er.error.a);  *// console.log(err);*  })  }    }  cancelEditReply(i,j){  **let** replybox = document.getElementById(i + 'replybox' + j);  **let** redit = document.getElementById(i + 'redit-box' + j);  redit.style.display === "flex" ? redit.style.display = "none" : redit.style.display = "none";  replybox.style.display = 'flex';  *// this.editstatus = 0;*  *// this.replystatus = 0;*  }  } |

**ACTIVITIES**

|  |
| --- |
| **export** **class** ActivitiesComponent **implements** OnInit {  activities: **any**[] = [];  resources: **any**[] = [];  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(Breakpoints.XSmall);  topicObj: **any** = [];  activityObject: **any** = [];  **constructor**(  **public** uploadservice: **UploadsService**,  **public** \_ds: **DataService**,  **public** \_user: **UserService**,  **private** route: **Router**,  **public** dialog: **MatDialog**,  **private** breakpointObserver: **BreakpointObserver**,  **public** \_upload: **UploadsService**) { }  ngOnInit(): **void** {  **this**.getActivity();  **this**.getResources();  **this**.getTopic();  }  getFileExtension(filename) {  **if** (filename == 'pdf') {  **return** '#EA462E';  } **else** **if** (filename == 'docx') {  **return** '#2D5292';  }  **else** **if** (filename == 'ppt') {  **return** '#CA4223';  }  **else** **if** (filename == 'zip') {  **return** '#B23333';  }  **else** **if** (filename == 'txt') {  **return** '#546A7A';  } **else** {  **return** '#222';  }  }  getfileExt(filename) {  **return** filename.split('.').pop();  }  getActivity() {  **this**.\_ds.\_httpRequest('getcpost', { classcode: **this.\_user.getSelectedClass**().classcode\_fld, type: '' }, **1**).subscribe((data: **any**) => {  data = **this**.\_user.\_decrypt(data.a);  **this**.activities = data.payload;  **for** (**let** i = **0**; i < **this**.activities.length; i++) {  **if** (**this**.activities[i].type\_fld === **0** || **this**.activities[i].type\_fld === **1** && **this**.activities[i].datesched\_fld < **this**.\_user.getDateToday()) {  **this**.activities[i].openQuiz = **true**;  } **else** {  **this**.activities[i].openQuiz = **false**;  }  }  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  *// console.log(er);*  })  }  getResources() {  **this**.\_ds.\_httpRequest('getreslist', { classcode: **this.\_user.getSelectedClass**().classcode\_fld }, **1**).subscribe((data: **any**) => {  data = **this**.\_user.\_decrypt(data.a);  **this**.resources = **this**.splitLinkResource(data.payload);  console.log(**this**.resources);  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  })  }  getTopic() {  **let** load = {  classcode: **this.\_user.getSelectedClass**().classcode\_fld  }  **this**.\_ds.\_httpRequest('gettopic', load, **1**).subscribe((data: **any**) => {  data = **this**.\_user.\_decrypt(data.a)  **this**.topicObj = data.payload;  **this**.\_user.setTopic(data.payload);  *// console.log(data);*  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  **this**.topicObj = [];  *// console.log(er);*  })  }  previewResource(link, name) {  **let** data = {  resourceString: **link**,  reourcename: **name**  }  **const** dialogRef = **this**.dialog.open(ResourcePreviewComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  disableClose: **true**,  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('100%', '100%');  } **else** {  dialogRef.updateSize('100%', '100%');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  })  }  splitLinkResource(data: **any**) {  data.forEach(element => {  **if** (!element.filedir\_fld.includes("%=!!=%")) {  element.filedir\_fld = **this**.uploadservice.splitFilestring(element.filedir\_fld);  } **else** {  **let** temp = element.filedir\_fld.split("%=!!=%");  element.filedir\_fld = **this**.uploadservice.splitFilestring(temp[**0**]);  element.link\_fld = temp[**1**].split("//\*//");  }  });  **return** data;  }  } |

**View Assignment**

|  |
| --- |
| **export** **class** ViewAssignComponent **implements** OnInit {    isLoading = **0**  activity: **any** =[];  actTitle: **any**;  datePosted: **any**;  dueDate: **any**;  description: **any**;  totalScore: **any**;  attachments: **any**;  withfile: **number** = **0**;  type: **any**;  commentcount: **number**;  comments: **any** =[];  works: **any**;  files:**any** = [];  issubmitted: **any** = **0**;  isscored: **any**;  filepathtoupload: **any**;  withfileact: **number**;  isUploading: **boolean** = **false**;  submittedFiles=[];  filesToBeUpdated: **any**=[];  *// saveNewWork: number = 0;*  existingDir: **any**;  split: **any** = [];  combined: **any** = [];  replyobject: **any** = [];  replystatus:**number** = **0**;  recipientcode: **any**;  i: **any**;  commentcodeforreply: **any**;  commentcode: **any**;  commentcontent: **any**;  actObj: **any** = [];  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(Breakpoints.XSmall);    *//For Url Submission Settings*  urlToSubmit = [];  listOfUploadedUrl = [];  submissionControl:**number** = SubmissionType.DEFAULT  existingUrl = []  urlControlUI:**number** = GeneralControl.ADD;  *//Controlled by GeneralControl Enums*  urlStatus:**number** = **0**;  existingUrlStatus:**number** = **0**;  urlControlUISubmitOrResubmit:**number** = GeneralControl.CLOSE;  *//Mat menu for file and url choices*  file\_attachment:**boolean** = **true**;  url\_attachment:**boolean** = **true**;  **constructor**(**public** \_upload:**UploadsService**,  **public** route: **Router**,  **private** breakpointObserver: **BreakpointObserver**,  **public** dialog: **MatDialog**,  **public** \_ds:**DataService**,  **public** \_user:**UserService**,  **public** \_us: **UploadsService**,  **private** \_snackbar:**MatSnackBar**,  **public** \_quest:**QuestService**  ) { }  arr = **this**.route.url.split('/');  questObject = **this**.\_user.getQuestData();  ngOnInit(): **void** {  **this**.getActivity();  **this**.getComments();  **this**.getWorks();  }  getActivity(){  **this**.\_ds.\_httpRequest('getcpost',{classcode:**this.\_user.getSelectedClass**().classcode\_fld, actcode:**this.arr**[**4**],type:'act'},**1**).subscribe((data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a);  **this**.actObj = data.payload;    **for** (**let** i = **0**; i < **this**.actObj.length; i++) {  **if** (**this**.arr[**4**] == **this**.actObj[i].actcode\_fld) {  **this**.activity.push(**this**.actObj[i]);  }  }    **this**.actTitle = **this**.activity[**0**].title\_fld;  **this**.datePosted = **this**.activity[**0**].datetime\_fld;  **this**.dueDate = **this**.activity[**0**].deadline\_fld;  **this**.description = **this**.activity[**0**].desc\_fld;  **this**.totalScore = **this**.activity[**0**].totalscore\_fld;  **this**.attachments = **this**.\_upload.splitFilestring(**this**.activity[**0**].filedir\_fld);  **this**.withfileact = **this**.activity[**0**].withfile\_fld;  **this**.type = **this**.activity[**0**].type\_fld;  **this**.commentcount = **this**.activity[**0**].commentcount;    },er => {  er = **this**.\_user.\_decrypt(er.error.a);  **this**.\_snackbar.open("Failed to load activity.",**null**,{duration:**1500**});  })  }  getComments(){  **let** load = {  acode:**this.arr**[**4**]  }  **this**.\_ds.\_httpRequest('getccomment',load,**1**).subscribe((data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a);    **this**.comments = data.payload;    },er=>{  er = **this**.\_user.\_decrypt(er.error.a);  **this**.comments = [];  })  }  getWorks(){  **let** load = {  userid:**this.\_user.getUserID**(),  classcode:**this.\_user.getSelectedClass**().classcode\_fld,  actcode:**this.arr**[**4**],  type:'act'  }  **this**.\_ds.\_httpRequest('getstudworks',load,**1**).subscribe((data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a);  **this**.works = data.payload;  **switch** (**this**.works[**0**].type\_fld) {  **case** ActivityType.DOCUMENTS:  **this.urlOrFileAttachment**(SubmissionType.FILE)  **this**.submissionControl = SubmissionType.FILE    **if** (**this**.works[**0**].dir\_fld == '') {  **this**.submittedFiles = [];  **return**  }    **this**.submittedFiles = **this**.\_upload.splitFilestring(**this**.works[**0**].dir\_fld)  **this**.existingDir = **this**.works[**0**].dir\_fld;  **break**;  **case** ActivityType.URL:  **this.urlOrFileAttachment**(SubmissionType.URL)  **this**.submissionControl = SubmissionType.URL    **this**.urlControlUI = GeneralControl.EDIT  **if** (**this**.works[**0**].dir\_fld == '') {  **this**.listOfUploadedUrl = []  **return**  }  **this**.listOfUploadedUrl = **this**.works[**0**].dir\_fld.split(',');  **this**.existingUrl = **this**.works[**0**].dir\_fld;  **this**.existingUrlStatus = GeneralControl.SUCCESS;  **break**;    **default**:  **break**;  }  **this**.issubmitted = **this**.works[**0**].issubmitted\_fld;  **this**.isscored = **this**.works[**0**].isscored\_fld;  **this**.urlStatus = GeneralControl.SUCCESS  },(er) => {  **let** err = **this**.\_user.\_decrypt(er.error.a);  })  }  addComment(e){  **this**.\_user.setLoading(**true**);  e.preventDefault()  **let** textcontent:**string** = e.target[**0**].value;  **if** (textcontent == **null** || textcontent == **undefined** || textcontent == '') {  **this**.\_user.setLoading(**false**);  **this**.\_snackbar.open('Please write a content for your comment.', **null**, {duration:**1500**});  }**else**{  **let** load = {  data:{  classcode\_fld:**this.\_user.getSelectedClass**().classcode\_fld,  authorid\_fld : **this.\_user.getUserID**(),  actioncode\_fld:**this.arr**[**4**],  content\_fld : **textcontent**,  withfile\_fld: **0**,  dir\_fld : ''  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message: **this.\_user.getFullname**()+' added a comment on '+**this**.\_user.getSelectedClass().classcode\_fld,  module: 'classroom'  }  }    **this**.\_ds.\_httpRequest('addcomment',load,**1**).subscribe((data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a);  e.target[**0**].value = **null**;  **this**.commentcount = **this**.commentcount + **1**;    **this**.comments.unshift(data.payload[**0**]);  **this**.\_user.setLoading(**false**);  })  }    }  async addWork(){  await **this**.Filepath().then((filepath:**any**)=>{  **new** Promise<**void**>(() => {  **let** load = **this**.dataToSubmit(filepath,ActivityType.DOCUMENTS,'file')    **this**.\_ds.\_httpRequest('addwork',load,**1**).subscribe((data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a);  **this**.isUploading = **false**;  **this**.works = data.payload;  **this**.files = [];  **this**.existingDir = **this**.works[**0**].dir\_fld;  **this**.submittedFiles = **this**.\_upload.splitFilestring(**this**.works[**0**].dir\_fld);  **this**.issubmitted = **this**.works[**0**].issubmitted\_fld;  **this**.isscored = **this**.works[**0**].isscored\_fld;    *//Update Quest User Data*  **if** (**this**.questObject.taskcnt\_fld\_limit > **0**) {  **this**.\_quest.UpdateCountAchievement(Crud.ADD,fldNames.taskcnt\_fld);  }    },er => {  er = **this**.\_user.\_decrypt(er.error.a);  **this**.isUploading = **false**;  })  });  }).**catch**(e=>{  });    }  editComment(index,data,code,options){  **let** condel  **let** load = {  option: **options**,  isConfirmed: **condel**,  data: **data**  };  **const** dialogRef = **this**.dialog.open(DialogComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  data: **load**,    disableClose: **true**,  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('30%', 'auto');  }  });    dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result.data) {  **let** load = {  data:{  classcode\_fld:**this.\_user.getSelectedClass**().classcode\_fld,  authorid\_fld:**this.\_user.getUserID**(),  content\_fld:**result.data**,  *// withfile\_fld:0,*  *// dir\_fld:'',*  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message: **this.\_user.getFullname**()+' edited a comment on this activity'+ **this**.actTitle,  module: 'classroom'  }  }  **this**.\_ds.\_httpRequest('editcomm/'+code, load , **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a);  **this**.comments[index] = res.payload[**0**];  }, er => {  **let** err = **this**.\_user.\_decrypt(er.error.a);  **if** (err.payload == **null**) {  **this**.getComments();  }    })  }  });  }  *// Start of uploading files*  getFile(event) {  **let** sum: **number** = **0**;  **for** (**var** i = **0**; i < event.target.files.length; i++) {  sum += event.target.files[i].size;  **this**.files.push(event.target.files[i]);  }    **if** (sum > **30000000**) {  **this**.\_snackbar.open((sum / **1048576**).toFixed(**2**).toString() + "File to big error",**null**,{duration:**1500**})  **this**.\_snackbar.open('Your total upload file(S) is ' + ((sum / **1048576**).toFixed(**2**).toString()) + 'MB. Please upload only atleast 30MB of file(s)', "", {duration:**1500**});  **this**.files = [];  (<HTMLInputElement>document.getElementById('files')).value = '';  }**else**{  **if** (**this**.files.length > **0**) {  **this**.urlOrFileAttachment(SubmissionType.FILE)  **this**.submissionControl = SubmissionType.FILE  **if** (**this**.dueDate < **this**.\_user.getDateToday()) {  **this**.withfile = **2**; *//issubmitted 1 ontime 2 late*  }**else**{  **this**.withfile = **1**;  }  }  (<HTMLInputElement>document.getElementById('files')).value = '';  }  }    deleteDialog(i, option){  **let** condel  **const** dialogConfig = **new** MatDialogConfig();  **if** (**this**.\_user.isMobile()) {  dialogConfig.maxHeight = '90vw';  dialogConfig.minHeight = 'auto';  dialogConfig.minWidth = '70vw';  dialogConfig.maxWidth = '90vw';  } **else** {  dialogConfig.minHeight = '15vh';  dialogConfig.minWidth = '15vw';  dialogConfig.maxHeight = 'auto';  }  dialogConfig.data = {  option: **option**,  isConfirmed: **condel**  }  **const** dialogRef = **this**.dialog.open(DialogComponent, dialogConfig);  dialogRef.afterClosed().subscribe(result => {  **if** (result.data) {  **this**.unlinkedFile(i);  }  });  }  async unlink(path){  **let** load = {dir\_fld:**path**}  **this**.isLoading = **1**  **return** **new** Promise((resolve,rejects)=>{  **this**.\_ds.\_httpRequest('deletefile',load,**1**).subscribe((data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a);  resolve(data.status.remarks);  },er=>{  er = **this**.\_user.\_decrypt(er.error.a);  **return** **new** Promise(()=>{  rejects(er.status.remarks)  })  })  })  }  async unlinkedFile(i){  **this**.\_user.setLoading(**true**);  **let** newFiles:**any** = [];  await **this**.unlink(**this**.submittedFiles[i].path).then(()=>{  **this**.submittedFiles.splice(i,**1**);  **this**.submittedFiles.map((e)=>{  newFiles.push(e.name + '?' + e.path);  })  newFiles = newFiles.join(':');  **let** load = {  data:{  dir\_fld:**newFiles**  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message: **this.\_user.getFullname**()+' remove a file on '+**this**.\_user.getSelectedClass().subjdesc\_fld,  module: 'classroom'  }    }  **let** api = `editsubmit/${**this**.works[**0**].submitcode\_fld}/${**this**.works[**0**].actcode\_fld}`  **this**.\_ds.\_httpRequest(api,load,**1**).subscribe((data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a);  **this**.existingDir = data.payload[**0**].dir\_fld;  **if** (data.payload[**0**].dir\_fld == '') {  **this**.urlOrFileAttachment(SubmissionType.DEFAULT)  **this**.withfile = **0**  }**else**{  **this**.withfile = data.payload[**0**].issubmitted\_fld;  }  **this**.isLoading = **0**  **this**.\_user.setLoading(**false**);    },er=>{  er = **this**.\_user.\_decrypt(er.error.a);  **this**.\_user.setLoading(**false**);  })    })      }    async Filepath(){  **let** formdata: **FormData**;  formdata = **this**.uploadFile(**this**.files);  **return** **new** Promise((resolve, rejects) => {  **this**.\_ds.\_httpRequest('upload/' + **this**.\_user.getUserID(), formdata, **3**).subscribe((data: **any**) => {  data = **this**.\_user.\_decrypt(data.a);  resolve(data.payload.filepath);  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  rejects(er.error.a)  });  })  }    uploadFile(payload) {  **this**.isUploading = **true**;  **const** formData = **new** FormData();  **for** (**var** i = **0**; i < **this**.files.length; i++) {  formData.append('file[]', **this**.files[i]);  }  formData.append('payload', JSON.stringify(payload));  **return** formData;  }  *// End of uploading files*  unsubmit(){  **let** load ={  data:{  dir\_fld:**this.filepathtoupload**,  issubmitted\_fld: **0**  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message: **this.\_user.getFullname**()+' save a work in '+**this**.\_user.getSelectedClass().subjdesc\_fld,  module: 'classroom'  }  }  **let** api = `editsubmit/${**this**.works[**0**].submitcode\_fld}/${**this**.works[**0**].actcode\_fld}`    **this**.\_ds.\_httpRequest(api,load,**1**).subscribe((data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a)  **this**.issubmitted = **0**  **this**.isUploading = **false**;  },er=>{  er = **this**.\_user.\_decrypt(er.error.a)  })  }  removeFile(i) {    **this**.files.splice(i, **1**);  **if** (**this**.files.length == **0**) {  **this**.urlOrFileAttachment(SubmissionType.DEFAULT)  **this**.withfile = **0**;  }  }  *//start of saving activity*  async save(){  **if** (**this**.files.length != **0**) {  await **this**.Filepath().then((filepath)=>{  **if** (**this**.existingDir === '' || **this**.existingDir === **undefined** || **this**.existingDir === **null**) {  **this**.filepathtoupload = filepath;  **return**  }  **this**.split = **this**.existingDir.split(':');  **this**.split.push(filepath);  **this**.combined = **this**.split.join(':');  **this**.filepathtoupload = **this**.combined;  }).**catch**(e=>{  **this**.filepathtoupload = '';  });  }**else**{  **this**.filepathtoupload = **this**.existingDir;  }  **if**(**this**.filepathtoupload == '' || **this**.filepathtoupload == **undefined**) {  **this**.\_snackbar.open('Add a File or Url',**null**,{duration:**1500**});  **return**  }    **let** load = **this**.dataToSubmit(**this**.filepathtoupload,ActivityType.DOCUMENTS,'file')  **this**.isLoading = **1**    **let** api = `editsubmit/${**this**.works[**0**].submitcode\_fld}/${**this**.works[**0**].actcode\_fld}`  **this**.\_ds.\_httpRequest(api,load,**1**).subscribe((data:**any**)=>{  **this**.isLoading = **0**  data = **this**.\_user.\_decrypt(data.a);  **this**.existingDir = data.payload[**0**].dir\_fld;  **this**.works = data.payload;  **this**.files = [];  **this**.issubmitted = data.payload[**0**].issubmitted\_fld;  **this**.isUploading = **false**;  **if** (data.payload[**0**].dir\_fld == '') {  **this**.submittedFiles = []  **return**  }  **this**.submittedFiles = **this**.\_upload.splitFilestring(data.payload[**0**].dir\_fld);    },er => {  er = **this**.\_user.\_decrypt(er.error.a);  })    }  *// end of saving activity*  back() {  **this**.route.navigateByUrl('main/classes/'+**this**.\_user.getSelectedClass().classcode\_fld);  }    viewQuiz(i){  **let** load: **any**;  **if**(i==**0**){  load = {view: **0**, load: **this.activity**};  }  **else**{  load = {view: **1**, load: **this.works**[**0**]};  }  **const** dialogRef = **this**.dialog.open(ViewQuizComponent, {  panelClass: 'quiz',  maxWidth: '100vw',  maxHeight: '100vh',  data: **load**,  disableClose: **true**,  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('100%', '100%');  } **else** {  dialogRef.updateSize('100%', '100%');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result.data) {  **let** data = {  data:{  classcode\_fld:**this.\_user.getSelectedClass**().classcode\_fld,  actcode\_fld: **this.arr**[**4**],  studnum\_fld:**this.\_user.getUserID**(),  type\_fld:**result.data.type**,  dir\_fld:**result.data.dir\_fld**,  issubmitted\_fld: **result.data.submitted**,  isscored\_fld:**result.data.isscored\_fld**,  score\_fld:**result.data.score\_fld**  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message: **this.\_user.getFullname**()+' submitted the quiz ',  module: 'classroom'  }  }  **this**.\_ds.\_httpRequest('addwork',data,**1**).subscribe((data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a);  **this**.works = data.payload;    },er=>{  er = **this**.\_user.\_decrypt(er.error.a);  })  }  });    }  deletecomment(i,commentcode , option){  **let** condel  **const** dialogConfig = **new** MatDialogConfig();  dialogConfig.data = {  option: **option**,  isConfirmed: **condel**,  data:**commentcode**  }  **const** dialogRef = **this**.dialog.open(DialogComponent, dialogConfig);  dialogRef.afterClosed().subscribe(result => {  **if** (result.data) {  **let** load = {    data:{  classcode\_fld:**this.\_user.getSelectedClass**().classcode\_fld,  authorid\_fld:**this.\_user.getUserID**(),  isdeleted\_fld:**1**  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message: **this.\_user.getFullname**()+' created a comment on '+**this**.\_user.getSelectedClass().subjdesc\_fld,  module: 'classroom'  }  }  **this**.\_ds.\_httpRequest('editcomm/'+commentcode, load , **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a);  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  **this**.comments.splice(i,**1**);  **this**.commentcount = **this**.commentcount - **1**;  })  }  });  }  deletereply(i,j,commentcode , option){  **let** condel  **const** dialogConfig = **new** MatDialogConfig();  dialogConfig.minWidth = '30vw',  dialogConfig.minHeight = '20vh',  dialogConfig.width = '30vw',  dialogConfig.data = {  option: **option**,  isConfirmed: **condel**,  data:**commentcode**  }  **const** dialogRef = **this**.dialog.open(DialogComponent, dialogConfig);  dialogRef.afterClosed().subscribe(result => {  **if** (result.data) {  **let** load = {  data:{  classcode\_fld:**this.\_user.getSelectedClass**().classcode\_fld,  authorid\_fld:**this.\_user.getUserID**(),  isdeleted\_fld:**1**  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message: **this.\_user.getFullname**()+' created a comment on '+**this**.\_user.getSelectedClass().subjdesc\_fld,  module: 'classroom'  }  }  **this**.\_ds.\_httpRequest('editcomm/'+commentcode, load , **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a);  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  **this**.comments[i].reply.splice(j,**1**);  })  }  });  }  reply(authorid\_fld,actioncode\_fld){  **this**.replystatus = **1**;  **this**.recipientcode = actioncode\_fld;    }  cancelreply(){  **this**.replystatus = **0**;  }  addReply(e,item,i){  e.preventDefault();  **this**.\_user.setLoading(**true**);    **let** replycontent = e.target[**0**].value;    **if** (replycontent == '' || replycontent == **undefined** || replycontent == **null**) {  **this**.\_snackbar.open('Please write a content for your reply.', **null**, {duration:**1500**});  **this**.\_user.setLoading(**false**);  }**else**{  **let** load = {  data:{  classcode\_fld:**this.\_user.getSelectedClass**().classcode\_fld,  authorid\_fld : **this.\_user.getUserID**(),  actioncode\_fld:**item.commentcode\_fld**,  content\_fld : **replycontent**,  withfile\_fld: **0**,  dir\_fld : ''  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message: **this.\_user.getFullname**()+' added a reply in'+ **this**.actTitle,  module: 'classroom'  }    }    **this**.\_ds.\_httpRequest('addcomment',load,**1**).subscribe((data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a);    **if** (**this**.comments[i].reply == **null**) {  **this**.comments[i].reply = [];  **this**.comments[i].reply.push(data.payload[**0**]);  } **else** {  **this**.comments[i].reply.push(data.payload[**0**]);  }  e.target[**0**].value = **null**;  **this**.replystatus = **0**;  **this**.\_user.setLoading(**false**);  *// this.editstatus = 0;*  *// this.postcomments.unshift(this.\_user.\_convert(data.payload[0]));*  },er=>{  er = **this**.\_user.\_decrypt(er.error.a);  **this**.\_user.setLoading(**false**);  *// this.\_snackbar.open(er.message, null, {duration:1500});*    })  }    }  editthis(index,reply,i){  **this**.commentcode = reply.commentcode\_fld;  **this**.commentcontent = reply.content\_fld;  **this**.i = index;  (<HTMLInputElement>document.getElementById(i + 'reply\_content\_fld' + index)).value = **this**.commentcontent;    **let** reditbox = document.getElementById(i + 'redit-box' + index);    reditbox.style.display === "none" ? reditbox.style.display = "flex" : reditbox.style.display = "none";  }  showReply(i){  **var** x = document.getElementById('replydiv'+i);  x.style.display === "none" ? x.style.display = "block" : x.style.display = "none";  **let** j = document.getElementById('replybtn' + i);  j.style.display === "none" ? j.style.display = "block" : j.style.display = "none";  }  getFileExtension(filename) {  **if** (filename == 'docx') { **return** '#004db3'; }  **if** (filename == 'pptx') { **return** '#c34524'; }  **if** (filename == 'txt') { **return** '#95a5a6'; }  **if** (filename == 'pdf') { **return** '#ad0b00'; }  **else** {  **return** '#1f6499';  }  }  getfileExt(filename) {  **return** filename.split('.').pop();  }  editReply(e,i){  e.preventDefault();  **this**.\_user.setLoading(**true**);  **let** commentcontent = e.target[**0**].value;  **if** (commentcontent == '' || commentcontent == **undefined** || commentcontent == **null**) {  **this**.\_snackbar.open('Please write a content for your comment.', **null**, {duration:**1500**});  **this**.\_user.setLoading(**false**);  }**else**{  **let** load = {  data:{  classcode\_fld:**this.\_user.getSelectedClass**().classcode\_fld,  authorid\_fld:**this.\_user.getUserID**(),  content\_fld:**commentcontent**  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message: **this.\_user.getFullname**()+' added a comment in'+ **this**.actTitle,  module: 'classroom'  }  }  **this**.\_ds.\_httpRequest('editcomm/'+**this**.commentcode, load , **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a);  **this**.comments[i].reply[**this**.i] = res.payload[**0**];  *// this.replyobject[this.i] = res.payload[0];*  *// this.editstatus = 0;*  **this**.replystatus = **0**;  **this**.\_user.setLoading(**false**);  }, er => {  **let** err = **this**.\_user.\_decrypt(er.error.a);  })  }    }  cancelEditReply(i,j){  **let** replybox = document.getElementById(i + 'replybox' + j);  **let** redit = document.getElementById(i + 'redit-box' + j);  redit.style.display === "flex" ? redit.style.display = "none" : redit.style.display = "none";  replybox.style.display = 'flex';  }  previewResource(link, name) {  **let** data = {  resourceString: **link**,  reourcename: **name**  }  **const** dialogRef = **this**.dialog.open(ResourcePreviewComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  disableClose: **true**,  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('100%', '100%');  } **else** {  dialogRef.updateSize('100%', '100%');  }  });    dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  })  }  */\*\**  *\* Open Dialog to insert url in the array*  *\* and push it to urlToSubmit Array*  *\*/*  addLink(){  **const** dialogConfig = **new** MatDialogConfig;  dialogConfig.data = {  option:'addLink'  }  **const** dilaogRef = **this**.dialog.open(DialogComponent, dialogConfig)  dilaogRef.afterClosed().subscribe(result => {  **if** (result) {    **this**.urlToSubmit.push(result.data)  **this**.works == **null** ? **this**.urlControlUISubmitOrResubmit = GeneralControl.SUBMIT : **this.urlControlUISubmitOrResubmit** = GeneralControl.SAVE    *// this.dueDate < this.\_user.getDateToday() ? this.withfile = 2 : this.withfile = 1;*  **this**.urlOrFileAttachment(SubmissionType.URL)  **this**.submissionControl = SubmissionType.URL  **this**.urlStatus = GeneralControl.REMOVE    }  })  }  */\*\**  *\* Remove a item before uploading to the database*  *\* @method removeLinkBeforeUpload*  *\* @param i*  *\*/*  removeLinkBeforeUpload(i){  **this**.urlToSubmit.splice( i, **1**);  **if** (**this**.urlToSubmit.length == **0**) {  **this**.urlOrFileAttachment(SubmissionType.DEFAULT)  **this**.urlControlUISubmitOrResubmit = GeneralControl.CLOSE  }  }  removeExistingLink(i){  **this**.listOfUploadedUrl.splice(i,**1**)  **if** (**this**.listOfUploadedUrl.length == **0**) {  **this**.urlOrFileAttachment(SubmissionType.DEFAULT)  **this**.urlControlUISubmitOrResubmit = GeneralControl.SAVE  }  }  **public** get generalControl(): **typeof** GeneralControl {  **return** GeneralControl;  }  **public** get submissionType(): **typeof** SubmissionType {  **return** SubmissionType  }  */\*\**  *\* @method submitLinkToDatabase*  *\* Submit list of links to the database*  *\*/*  submitLinkToDatabase(){  **let** urlList = **this**.urlToSubmit.join(',')  **let** load = **this**.dataToSubmit(urlList,ActivityType.URL,'URL')    **this**.\_ds.\_httpRequest('addwork',load,**1**).subscribe((data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a);  **this**.works = data.payload;  **this**.urlToSubmit = []; *//To remove duplicate in display list of links*    **this**.listOfUploadedUrl = **this**.works[**0**].dir\_fld.split(',');  **this**.issubmitted = **this**.works[**0**].issubmitted\_fld;  **this**.isscored = **this**.works[**0**].isscored\_fld;  **this**.urlControlUI = GeneralControl.EDIT;  **this**.urlStatus = GeneralControl.SUCCESS;  **this**.existingUrlStatus = GeneralControl.SUCCESS;      *//Update Quest User Data*  **if** (**this**.questObject.taskcnt\_fld\_limit > **0**) {  **this**.\_quest.UpdateCountAchievement(Crud.ADD,fldNames.taskcnt\_fld);    }    },er => {  er = **this**.\_user.\_decrypt(er.error.a);  **this**.isUploading = **false**;  })  }  resubmitLink(){  *//Merge two array then submit as edit in db*  **let** mergeUrlResult = [...**this**.listOfUploadedUrl,...**this**.urlToSubmit]  **if** (mergeUrlResult.length == **0**) {  **this**.\_snackbar.open('Add Url or File',**null**,{duration:**1500**});  **return**  }  **let** urlListToResubmit = mergeUrlResult.join(',')  **let** load = **this**.dataToSubmit(urlListToResubmit,ActivityType.URL,'url')  **let** api = `editsubmit/${**this**.works[**0**].submitcode\_fld}/${**this**.works[**0**].actcode\_fld}`    **this**.\_ds.\_httpRequest(api,load,**1**).subscribe((data:**any**)=> {  data = **this**.\_user.\_decrypt(data.a)    **this**.urlToSubmit = []  **this**.listOfUploadedUrl = data.payload[**0**].dir\_fld.split(',')  **this**.issubmitted = data.payload[**0**].issubmitted\_fld  **this**.urlControlUI = GeneralControl.EDIT  **this**.existingUrlStatus = GeneralControl.SUCCESS  **this**.urlStatus = GeneralControl.SUCCESS;  })    }  editLinkLists(){  **this**.issubmitted = **0**;  **this**.existingUrlStatus = GeneralControl.REMOVE\_EXISTING;  **this**.urlStatus = GeneralControl.REMOVE  **this**.urlControlUISubmitOrResubmit = GeneralControl.SAVE;  **this**.urlControlUI = GeneralControl.ADD;  }  dataToSubmit(dir,type,message){  **this**.dueDate < **this**.\_user.getDateToday() ? **this**.withfile = 2 : **this.withfile** = **1**;  **let** load = {  data:{  classcode\_fld:**this.\_user.getSelectedClass**().classcode\_fld,  actcode\_fld: **this.arr**[**4**],  studnum\_fld:**this.\_user.getUserID**(),  type\_fld:**type**,  dir\_fld:**dir**,  issubmitted\_fld: **this.withfile**  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message: `${**this**.\_user.getFullname()} submitted a ${message} on ${**this**.actTitle}`,  module: 'classroom'  }  }  **return** load  }  urlOrFileAttachment(state){  **switch** (state) {  **case** SubmissionType.URL:  **this.url\_attachment** = **true**  **this**.file\_attachment = **false**  **break**;  **case** SubmissionType.FILE:  **this.url\_attachment** = **false**  **this**.file\_attachment = **true**;  **break**;  **case** SubmissionType.DEFAULT:  **this.url\_attachment** = **true**  **this**.file\_attachment = **true**;  **break**;  **default**:  **this**.\_snackbar.open('Error in url or file attachment',**null**,{duration:**1000**})  **break**;  }  }  } |

**View Quiz**

|  |
| --- |
| **export** **class** ViewQuizComponent **implements** OnInit {  quizpath: **any**;  status: **number**;  totalquestion: **any**;  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(Breakpoints.XSmall);  **constructor**(  **private** breakpointObserver: **BreakpointObserver**,  **public** \_snackbar:**MatSnackBar**,  **public** \_ds:**DataService**,**public** \_user:**UserService**,  @Inject(MAT\_DIALOG\_DATA) **public** data: **any**,  **public** dialogReg: **MatDialogRef**<ViewQuizComponent>,  **public** dialog: **MatDialog**) { }  count:**number** = **0**;  quizObject = [];  questionObject;  answerkeyObject = [];  isSubmitConfirmed = **false**  quizScore = [];  index = **0**;  submit = **0**;  answerObject = [{  questionid: '',  question: '',  answer: ''  }];  question: **string**;  questiontype: **any**;  choice: **string**;  answer: **string**;  start:**number** = **0**;  open: **boolean** = **false**;  newdata = [];  overall: **number** = **0**;  quiztotal: **number** = **0**;  totalscore: **number** = **0**;  passingscore: **number** = **0**;  randomQuestion = [];  ngOnInit(): **void** {  **if**(**this**.data.view == **1** ){  **this**.getQuizScore(**this**.data.load['dir\_fld']);  **this**.start = **2**;  }  **else**{  **this**.getQuiz(**this**.data.load[**0**]['filedir\_fld']);  **if** (**this**.data.load[**0**]['deadline\_fld'] < **this**.\_user.getDateToday()) {  **this**.status = **2**;  }**else**{  **this**.status = **1**;  }      }    }  initializeAnswerObject() {  **for** (**let** i = **0**; i < **this**.questionObject.length - **1**; i++) {  **this**.answerObject.push({ questionid: '', question: '', answer: '' })  }    }  setAnswers(i, answer) {    **let** questionItem = **this**.questionObject[i]  **this**.answerObject[i] = {  questionid: **questionItem.questionid**,  question: **questionItem**,  answer: **answer**  };  }  isSubmitButtonDisabled = **false**;  async confirmQuiz(){  **this**.isSubmitButtonDisabled = **true**    **let** data = {  option: 'quizConfirmation',  isConfirmed: '',  data: ''  };  **const** dialogRef = await **this**.dialog.open(DialogComponent, {  panelClass: 'dialogpadding',  autoFocus: **true**,  maxWidth: '100vw',  maxHeight: '100vh',  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('20%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if**(result.data == **1**) **this**.SubmissionQuiz();  **this**.isSubmitButtonDisabled = **false**  });  }  async SubmissionQuiz() {  **for** (**let** index = **0**; index < **this**.answerObject.length; index++) {  **if**(**this**.answerObject[index] != **null** && **this**.answerObject[index].question != '' ){  }**else**{  **let** emptyAnswer = {  question: "Not Answered Yet!",  questionid: **index.toString**(),  answer: "Not Answered Yet!"  }    **this**.answerObject.splice(index,**1**,emptyAnswer);  }  }  **for** (**let** index = **0**; index < **this**.answerObject.length; index++) {  **if**(**this**.answerObject[index].question == "Not Answered Yet!"){  **this**.\_snackbar.open("Please answer all questions.",**null**,{duration:**1500**});  **return**;  }  }    **let** load = {  data:{  filepath:**this.data.load**[**0**]['filedir\_fld'],  answer: **await** **this**.calculateQuiz()  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message: **this.\_user.getFullname**()+' submitted the quiz ',  module: 'classroom'  }  }  **this**.\_ds.\_httpRequest('addanswer', load, **1**).subscribe((res: **any**) => {  res = **this**.\_user.\_decrypt(res.a);  **this**.quizpath = res.payload.filepath;  **this**.quizScore = res.payload.result.result;  **this**.totalscore = res.payload.result.totalscore;  **this**.quiztotal = res.payload.result.quiztotal;  **this**.passingscore = **this**.quiztotal \* .**6**;  **this**.start = **2**;  **this**.dialogReg.close({  data:{  dir\_fld:**this.quizpath**,  submitted:**this.status**,  type:**1**,  score\_fld:**this.totalscore**,  isscored\_fld:**1**  }  });    }, er => {  er = **this**.\_user.\_decrypt(er.error.a);    })  }  *// submitQuiz(){*  *// }*  Previewanswer() {  **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  **this**.dialogReg.updateSize('100%', '100%');  } **else** {  **this**.dialogReg.updateSize('100%', '100%');  }  });  **if**(**this**.data.load['quizoptions\_fld'] == **1**){  **this**.open = !**this**.open;  }  }  cancelQuiz() {  **this**.dialogReg.close('');  }  getQuiz(str) {  **this**.\_ds.\_httpRequest('getquiz', { filepath: **str** }, **1**).subscribe( async (res: **any**) => {  res = **this**.\_user.\_decrypt(res.a);  **this**.quizObject = await res.payload.quiz;  **this**.answerkeyObject = await res.payload.answerkey.content  **this**.questionObject = **this**.arrayshuffle(**this**.quizObject['content']);  **this**.initializeAnswerObject()  }, er => {  })  }  arrayshuffle(data){  **let** newPos:**number**, temp:**number**;  **for** (**let** i = data.length - **1**; i > **0**; i--) {  newPos = Math.floor(Math.random() \* ( i+**1** ));  temp = data[i];  data[i] = data[newPos];  data[newPos] = temp;  }  **return** data;  }  getQuizScore(str) {  **this**.\_ds.\_httpRequest('getresult', { filepath: **str** }, **1**).subscribe((res: **any**) => {  res = **this**.\_user.\_decrypt(res.a);  **this**.quizScore = res.payload.result;  **this**.totalquestion = res.payload.result.length;  **this**.totalscore = res.payload.totalscore;  **this**.quiztotal = res.payload.quiztotal;  **this**.passingscore = **this**.quiztotal \* .**6**;  **this**.start = **2**;  }, er => {  })  }  getAnswers(id) {  **for** (**let** i = **0**; i <= **this**.answerkeyObject.length; i++) {  **if** (id == **this**.answerkeyObject[i].questionid) {  **return** **this**.answerkeyObject[i].answer;  }  }  }  startQuiz() {  **this**.start = **1**;  }  splitChoices(**string**) {  **return** **string**.split('//\*//');  }  calculateQuiz(){  **let** answer = **this**.answerObject  **let** answerkey = **this**.answerkeyObject  **let** totalScore = **0**  **let** totalQuizScore = **0**  answer.forEach((element, i) => {  **let** key = answerkey.find((item: **any** ) => { **return** element.questionid == item.questionid });  answer[i]['points'] = **0**  answer[i]['mark'] = **0**  totalQuizScore += key.score  **let** studAnswer = element.answer.trim()  **let** quizAnswer = key.answer.trim()  **if**(studAnswer.toUpperCase() == quizAnswer.toUpperCase()){  answer[i]['points'] = key.score  answer[i]['mark'] = **1**  totalScore += key.score  }  })  **let** answerJson = {  totalscore: **totalScore**,  quiztotal: **totalQuizScore**,  result: **answer**  }  **return** answerJson  }  checkIfQuestionIsString(question, isImage){  **if**(**typeof** question === 'string' && isImage === **0**) **return** question  **if**(**typeof** question !== 'object' && isImage === **1**) **return** **null**  **return** isImage == **1** ? question[**1**] : question[**0**]  }  viewimg(i){  *// const dialogConfig = new MatDialogConfig();*  *// dialogConfig.autoFocus = true;*  *// if (this.\_user.isMobile()) {*  *// dialogConfig.minHeight = 'auto';*  *// dialogConfig.minWidth = '90vw';*  *// dialogConfig.width = '90vw';*  *// } else {*  *// dialogConfig.minHeight = '83vh';*  *// dialogConfig.minWidth = '70vw';*  *// dialogConfig.width = '70vw';*  *// }*  *// dialogConfig.data = {*  *// height: '90vh',*  *// width: '70vw',*  *// resourceString: i*  *// }*  *// const dialogRef = this.dialog.open(ResourcePreviewComponent, dialogConfig)*  *// dialogRef.afterClosed().subscribe(result => {*  *// })*  *// const responsiveDialogSubscription = this.isExtraSmall.subscribe(result => {*  *// if (result.matches) {*  *// dialogRef.updateSize('1000px', '490px');*  *// } else {*  *// dialogRef.updateSize('1000px', '490px');*  *// }*  *// });*  }  } |

**Classlist**

|  |
| --- |
| **export** **class** MembersComponent **implements** OnInit {  instructor: **any**;  students: **any**;  fullname: **any**;  department: **any**;  profilepic: **any**;  pageslice;  **constructor**(**public** \_user:**UserService**, **public** \_ds:**DataService**) { }  ngOnInit(): **void** {  **this**.getMembers();    }  OnpageChange(event: **PageEvent**) {  **const** startIndex = event.pageIndex \* event.pageSize;  **let** endIndex = startIndex + event.pageSize;  **if** (endIndex > **this**.students.length) {  endIndex = **this**.students.length;  }  **this**.pageslice = **this**.students.slice(startIndex, endIndex);  }  getMembers(){  **this**.instructor = **this**.\_user.getTeachers();  **if** (**this**.instructor[**0**]['fname\_fld'] == **null**) {  **this**.fullname = "TBA";  **this**.department = "TBA";  **this**.profilepic = **null**;  }**else**{  **this**.fullname = **this**.instructor[**0**]['fname\_fld'] +" "+**this**.instructor[**0**]['lname\_fld'];  **this**.department = **this**.instructor[**0**]['dept\_fld'];  **this**.profilepic = **this**.instructor[**0**]['profilepic\_fld'];  }  **this**.students = **this**.\_user.getStudents();  **this**.pageslice = **this**.students.slice(**0**, **18**);  *// console.log(this.students,this.instructor);*    *// console.log(this.instructor);*    }  } |

**STUDENT WORKS**

|  |
| --- |
| **export** **class** StudentWorksComponent **implements** OnInit {  studworks: **any**;  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(Breakpoints.XSmall);  **constructor**(**public** \_upload:**UploadsService**,  **public** \_user:**UserService**,  **public** \_ds:**DataService**,  **private** dialog:**MatDialog**,  **private** breakpointObserver: **BreakpointObserver**) { }  ngOnInit(): **void** {  *// console.log(this.\_user.getSelectedClass());*  **this**.getWorks();  }    previewResource(link, name) {  **let** data = {  resourceString: **link**,  reourcename: **name**  }  **const** dialogRef = **this**.dialog.open(ResourcePreviewComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  disableClose: **true**,  data: **data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('100%', '100%');  } **else** {  dialogRef.updateSize('100%', '100%');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  })  }  getWorks(){  *// console.log(this.\_user.getSelectedClass().classcode\_fld);*  **let** load = {  userid:**this.\_user.getUserID**(),  classcode:**this.\_user.getSelectedClass**().classcode\_fld,  actcode:'',  type:''  }    **this**.\_ds.\_httpRequest('getstudworks',load,**1**).subscribe((data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a);  **this**.studworks = data.payload;  },er=>{  er = **this**.\_user.\_decrypt(er.error.a)  **throw** er  })  }  splitUrl(data){  **let** split = data.split(',')  **return** split;  }  } |

**FORUMS**

|  |
| --- |
| **export** **class** ForumComponent **implements** OnInit {  forumsdata: **any** = [];  showLoader: **boolean** = **true**;  filter: **string** = '';  allforum: **any** = [];  **constructor**(**private** router: **Router**, **public** \_user:**UserService**, **public** \_ds:**DataService**) { }  ngOnInit(): **void** {  **this**.getTopics();  }  getTopics(){  **this**.\_ds.\_httpRequest('getforumtopic',**null**,**1**).subscribe((data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a);  **this**.forumsdata = data.payload;  **this**.allforum = data.payload;  **this**.\_user.setMainForums(**this**.forumsdata);  *// console.log(this.forumsdata);*  **this**.showLoader = **false**;    },er => {  er = **this**.\_user.\_decrypt(er.error.a);  **this**.forumsdata = [];  **this**.allforum = [];  **this**.showLoader = **false**;  *// console.log(er);*    })  }  filterlist(){  **let** results: **any** = []  **if**(**this**.filter != ''){  **this**.allforum.forEach(index => {  **if**(index.forumtitle\_fld.toLowerCase().includes(**this**.filter.toLowerCase())||index.forumdesc\_fld.toLowerCase().includes(**this**.filter.toLowerCase())){  results.push(index);  *// console.log(results);*    }  })  **this**.forumsdata = results;  } **else** {  **this**.forumsdata = **this**.allforum;  }  }  } |

**SUBFORUMS**

|  |
| --- |
| **export** **class** ForumSubComponent **implements** OnInit {  subForumContent: **any** = [];  mainforumcontent: **any** = [];  forumcode;  **constructor**(**public** \_upload:**UploadsService**,**public** \_ds: **DataService**, **private** ar: **ActivatedRoute**, **public** \_user: **UserService**, **private** dialog: **MatDialog**, **private** breakpointObserver: **BreakpointObserver**, **private** \_snackbar: **MatSnackBar**) { }  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(Breakpoints.XSmall);  ngOnInit(): **void** {  **this**.getSubforum();  }  getSubforum() {  **let** mainforum = **this**.\_user.getMainForums();        **this**.ar.paramMap.subscribe((p) => { **this**.forumcode = p.get('id'); });  **for** (**let** i = **0**; i < mainforum.length; i++) {  **if** (mainforum[i].forumcode\_fld == **this**.forumcode) {  **this**.mainforumcontent.push(mainforum[i]);  }  }  *// console.log(this.forumcode);*    **this**.\_ds.\_httpRequest('getsubforum', { forumcode: **this.forumcode** }, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.subForumContent = dt.payload;  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  })  }  gotoContent(s: **Object**) {  **this**.\_user.setSubForums(s);  }  addSubForum() {  **const** dialogConfig = **new** MatDialogConfig();  dialogConfig.width = '100%',  dialogConfig.height = '100%',  dialogConfig.maxWidth = '100vw',  dialogConfig.maxHeight = '100vh',  dialogConfig.panelClass = 'dialogpadding',  dialogConfig.data = {  type: 'subforum',  code:**this.forumcode**  }  **const** dialogRef = **this**.dialog.open(ForumAddComponent, dialogConfig);  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('99%', 'auto');  } **else** {  dialogRef.updateSize('40%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  *// console.log(result.data);*  **if** (result.data) {  **this**.\_ds.\_httpRequest('addsubforum',result.data,**1**).subscribe((data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a);  **this**.subForumContent.push(data.payload[**0**]);  **this**.\_snackbar.open('Success!',**null**,{duration:**1000**})  },er=>{  er = **this**.\_user.\_decrypt(er.error.a);  **this**.\_snackbar.open('Failed!',**null**,{duration:**1000**})  })  }  });  }  } |

**Create Subforums**

|  |
| --- |
| **export** **class** ForumAddComponent **implements** OnInit {  **constructor**(**public** \_user:**UserService**, **public** \_ds:**DataService**,@Inject(MAT\_DIALOG\_DATA) **public** data: **any**,**private** dialogReg: **MatDialogRef**<DialogComponent>) { }  ngOnInit(): **void** {  *// console.log(this.data);*    }  addForum(e){  e.preventDefault();  **let** title = e.target[**0**].value;  **let** details = e.target[**1**].value;  **let** type = e.target[**2**].value;  **this**.dialogReg.close({  data : {  data:{  authorid\_fld: **this.\_user.getUserID**(),  subtitle\_fld: **title**,  subdesc\_fld: **details**,  type\_fld:**type**,  isapproved\_fld: **1**,  forumcode\_fld:**this.data.code**  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getUserID**(),  message: **this.\_user.getFullname**()+' requested to open a subforum.',  module: 'forums'  }  }  });    }  } |

**FORUMS CONTENT**

|  |
| --- |
| **export** **class** ForumContentComponent **implements** OnInit {  contents: **any** = [];  count: **any**;  replystatus:**number** = **0**;  recipientcode: **any**;  **constructor**(**public** \_upload:**UploadsService**,  **public** ar: **ActivatedRoute**, **public** \_ds: **DataService**,  **private** router: **Router**, **public** \_user: **UserService**,  **public** dialog: **MatDialog**,  **private** breakpointObserver: **BreakpointObserver**,  **public** \_quest:**QuestService**  ) { }  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(Breakpoints.XSmall);  ngOnInit(): **void** {  **this**.getContents();  }    getContents(){  **this**.\_ds.\_httpRequest('getsubcontent',{subcode:**this.\_user.getSubForums**().subcode\_fld},**1**).subscribe((data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a);  **this**.contents = data.payload;  **this**.count = **this**.contents.length;  console.log(**this**.contents);    },er=>{  er = **this**.\_user.\_decrypt(er.error.a);  *// console.log(er);*    **this**.contents = [];  **this**.count = **0**;  *// console.log(er);*    })  }  addContent(e){  e.preventDefault();  **this**.\_user.setLoadingInput(**true**);  **let** content = e.target[**0**].value;  **let** data = {  data:{  authorid\_fld:**this.\_user.getUserID**(),  content\_fld: **content**,  subcode\_fld:**this.\_user.getSubForums**().subcode\_fld  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getUserID**(),  message: **this.\_user.getFullname**()+' added a comment in a thread.',  module: 'forums'  }    }  *// console.log(data);*  **this**.\_ds.\_httpRequest('addreply',data,**1**).subscribe((data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a);  **this**.contents.push(data.payload[**0**]);  **this**.count = **this**.count + **1**;  e.target[**0**].value = '';  **this**.\_user.setLoadingInput(**false**);  *// console.log(data.payload);*  *//Update Quest User Data*  **this**.\_quest.UpdateCountAchievement(Crud.ADD,fldNames.cifcnt\_fld);      })  }  delcomment(forumcode , option){  **let** condel  **const** dialogConfig = **new** MatDialogConfig();  dialogConfig.panelClass = 'dialogpadding'  dialogConfig.data = {  option: **option**,  isConfirmed: **condel**  }  **const** dialogRef = **this**.dialog.open(DialogComponent, dialogConfig);  dialogRef.afterClosed().subscribe(result => {  *// console.log(result.data,forumcode);*    **if** (result.data) {  **let** load = {  data:{  isdeleted\_fld:**1**  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getUserID**(),  message: **this.\_user.getFullname**()+' deleted a comment in forum.',  module: 'forum'  }  }  **this**.\_ds.\_httpRequest('editreply/'+forumcode, load , **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a);  *// console.log(res);*    }, er => {  **let** err = **this**.\_user.\_decrypt(er.error.a);  **this**.getContents();  *// console.log(err);*      })  }  });  }  editSubForum(index,code,content) {  **const** dialogConfig = **new** MatDialogConfig();  dialogConfig.width = '100%',  dialogConfig.height = '100%',  dialogConfig.maxWidth = '100vw',  dialogConfig.maxHeight = '100vh',  dialogConfig.panelClass = 'dialogpadding'  dialogConfig.data = {  content:**content**  }  **const** dialogRef = **this**.dialog.open(ForumEditComponent, dialogConfig);  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('99%', 'auto');  } **else** {  dialogRef.updateSize('40%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  *// console.log(result.data);*  **if** (result.data) {  **this**.\_ds.\_httpRequest('editreply/'+code, result.data , **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a);  **this**.contents[index] = res.payload[**0**];  *// console.log(res);*    }, er => {  **let** err = **this**.\_user.\_decrypt(er.error.a);  *// this.getContents();*  *// console.log(err);*      })  }  });  }  addReply(e){  e.preventDefault();    **let** replycontent = e.target[**0**].value;  *// console.log(replycontent);*    **if** (replycontent == '' || replycontent == **undefined** || replycontent == **null**) {    *// console.log("put replycontent");*  }**else**{  **let** load = {    data:{  *// classcode\_fld:this.\_user.getSelectedClass().classcode\_fld,*  authorid\_fld : **this.\_user.getUserID**(),  actioncode\_fld:**this.recipientcode**,  content\_fld : **replycontent**,  withfile\_fld: **0**,  dir\_fld : ''  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getUserID**(),  message: **this.\_user.getFullname**()+' added a reply in a thread.',  module: 'forums'  }  }  *// console.log(load);*    **this**.\_ds.\_httpRequest('addcomment',load,**1**).subscribe((data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a);  *// this.getcomments();*  *// console.log(data);*    e.target[**0**].value = **null**;  **this**.replystatus = **0**;  *// this.postcomments.unshift(this.\_user.\_convert(data.payload[0]));*  },er=>{  er = **this**.\_user.\_decrypt(er.error.a);    })  }    }  replycontent(authorid,code){  **this**.replystatus = **1**;  **this**.recipientcode = code;  *// console.log(authorid,code);*  }  cancelreply(){  **this**.replystatus = **0**;  }  } |

**EDIT FORUMS COMMENTS**

|  |
| --- |
| **export** **class** ForumEditComponent **implements** OnInit {  content: **any**;  **constructor**(**public** \_user:**UserService**, **public** \_ds:**DataService**,@Inject(MAT\_DIALOG\_DATA) **public** data: **any**, **private** dialogReg: **MatDialogRef**<DialogComponent>) { }  ngOnInit(): **void** {  **this**.content = **this**.data.content;  *// console.log(this.data.content);*    }  editforum(e){  e.preventDefault();  **let** content = e.target[**0**].value;  **this**.dialogReg.close({  data : {  data:{  content\_fld:**content**  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getUserID**(),  message: **this.\_user.getFullname**()+' edited a comment in a thread.',  module: 'forums'  }  }  });  }  } |

**ELIBRARY**

|  |
| --- |
| **export** **class** LibraryComponent **implements** OnInit {  isactive: **boolean** = **false**;  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(  Breakpoints.XSmall  );  isTiled: **boolean** = **true**;  bookTitle: **string**;  bookObject: **any** = [];  bookObjectIndividual;  searchList;  **constructor**(  **private** breakpointObserver: **BreakpointObserver**,  **public** dialog: **MatDialog**,  **public** \_ds: **DataService**,  **public** \_user: **UserService**  ) {}  ngOnInit(): **void** {  **this**.getBooks();  }  toggleview() {  **this**.isTiled = !**this**.isTiled;  }  Search() {  **let** book: **any**;  **let** author: **any**;  book = **this**.searchList.filter((res: **any**) =>  res.title\_fld  .toLocaleLowerCase()  .match(**this**.bookTitle.toLocaleLowerCase())  );  author = **this**.searchList.filter((res: **any**) =>  res.author\_fld  .toLocaleLowerCase()  .match(**this**.bookTitle.toLocaleLowerCase())  );  **if** (**this**.bookTitle == '') {  **this**.getBooks();  } **else** **if** (**this**.bookTitle != '') {  **if** (book == '') {  **this**.bookObject = author;  } **else** {  **this**.bookObject = book;  }  }  }  getBooks() {  **this**.\_ds.\_httpRequest('getbooks', '', **1**).subscribe(  (dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.bookObject = dt.payload;  *// this.\_user.setBook(this.bookObject);*  **this**.searchList = **this**.bookObject;  },  (er) => {  *// let err = this.\_user.\_decrypt(er.error.a);*  console.clear();  **this**.bookObject = **null**;  }  );  }  viewBook(link) {  **let** data = {  resourceString: `${**this**.\_ds.booksURL}${link}#toolbar=**1**&navpanes=**1**`,  reourcename: **name**,  };  **const** dialogRef = **this**.dialog.open(ResourcePreviewComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  disableClose: **true**,  data: **data**,  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(  (result) => {  **if** (result.matches) {  dialogRef.updateSize('100%', '100%');  } **else** {  dialogRef.updateSize('100%', '100%');  }  }  );  dialogRef.afterClosed().subscribe((result) => {  responsiveDialogSubscription.unsubscribe();  });  }  } |

**NEWS AND ANNOUNCEMENTS**

|  |
| --- |
| **export** **class** NewsComponent **implements** OnInit {  image: **any**;  latestimg: **any**;  filter: **string**;  allnews: **any** = [];  latestreset: **any** = [];  **constructor**(**public** \_upload:**UploadsService**,**private** \_router: **Router**, **public** \_ds: **DataService**, **public** \_user: **UserService**) { }  newsdata=[**1**,**2**,**3**,**4**,**5**,**6**];  announcements = [];  latest = [];  showLoader: **boolean** = **true**;  ngOnInit(): **void** {  **this**.getAnnouncement();  }  viewContent() {  **this**.\_router.navigate(['./main/news/',**21**]);  }  getAnnouncement() {  **this**.\_ds.\_httpRequest('getannounce', { dept: **this.\_user.getDepartment**(), program: **this.\_user.getProgram**() }, **1**).subscribe((data: **any**) => {  data = **this**.\_user.\_decrypt(data.a);  **this**.announcements = data.payload;    **this**.allnews = data.payload;  **this**.latest = data.payload[**0**];  **this**.latestreset = data.payload[**0**];  **this**.showLoader = **false**;    },er => {  er = **this**.\_user.\_decrypt(er.error.a);  **this**.showLoader = **false**;    })  }  viewAnnouncement(i) {  **let** a = JSON.stringify([**this**.announcements[i]]);  **this**.\_user.setContentNews(a);  **this**.\_router.navigate(['./main/news', **this**.announcements[i].announcecode\_fld]);  }  filterlist(){  **let** results: **any** = []  **if**(**this**.filter != ''){  **this**.allnews.forEach(index => {  **if**(index.title\_fld.toLowerCase().includes(**this**.filter.toLowerCase()) || index.content\_fld.toLowerCase().includes(**this**.filter.toLowerCase())){  results.push(index);  }  })  **this**.announcements = results;  **this**.latest = [];  } **else** {  **this**.announcements = **this**.allnews;  **this**.latest = **this**.latestreset;  }  }  } |

**NEWS CONTENT**

|  |
| --- |
| **export** **class** NewContentComponent **implements** OnInit {  announcement = [];  randomannounce = [];  randomNumber: **any**;  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(Breakpoints.XSmall);  image: **any**;  randimage: **any** = [];  **constructor**(**private** \_router: **Router**, **private** user: **UserService**, **public** \_ds: **DataService**, **public** dialog: **MatDialog**, **private** breakpointObserver: **BreakpointObserver**) { }  ngOnInit(): **void** {  **let** a = **this**.user.getContentNews();  **this**.announcement = JSON.parse(a);  *// console.log(this.announcement);*    **this**.randomNumber = Math.floor(Math.random() \* **3**) + **1**;  **this**.getimg();  **this**.getAnnouncement();  }    viewAnnouncement(data) {  *// this.\_router.navigate(['./main/news', data['announcecode\_fld']]);*  **this**.announcement = [{  announcecode\_fld: **data**['announcecode\_fld'],  content\_fld: **data**['content\_fld'],  datetime\_fld: **data**['datetime\_fld'],  imgdir\_fld: **data**['imgdir\_fld'],  title\_fld: **data**['title\_fld'],  withimg\_fld: **data**['withimg\_fld'],  recipientcode\_fld: **data**['recipientcode\_fld'],  timestamp\_fld: **data**['timestamp\_fld']  }];    }  splitFilestring(filestring: **string**) {  **let** arr1 = filestring.split(':');  **let** filearray: { name: **string**; link: **string**,path:**string** }[] = [];  **for** (**let** i = **0**; i < arr1.length; i++) {  **let** arr2 = arr1[i].split('?');  filearray.push({ name: **arr2**[**0**], link: **this.\_ds.fileUrl** + arr2[**0**], path: **arr2**[**1**] });  }  **return** filearray;  }  getimg() {  **let** imagepath = **this**.splitFilestring(**this**.announcement[**0**]['imgdir\_fld']);  **this**.image = imagepath[**0**].link;  **return** **this**.image;  }  getAnnouncement() {  **this**.\_ds.\_httpRequest('getannounce', { dept: **this.user.getDepartment**(), program: **this.user.getProgram**() }, **1**).subscribe((data: **any**) => {  data = **this**.user.\_decrypt(data.a);  **this**.randomannounce = data.payload;  *// console.log(this.randomannounce);*    *// for (let i = 0; i < this.randomannounce.length; i++) {*  *// if (this.randomannounce[i].withimg\_fld == 1) {*  *// this.randimage = this.\_ds.fileUrl + this.randomannounce[i].imgdir\_fld*    *// // this.randimage = imagepath[i].link*  *// // console.log(imagepath[i].link);*    *// }*    *// }*    })  }  openComments(acode,count){  **let** data = {  acode: **acode**,  commentcnt: **count**  };  **const** dialogRef = **this**.dialog.open(NewsCommentsComponent, {  panelClass: '',  maxWidth: '100vw',  maxHeight: '100vh',  data:**data**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('50%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();    });  }  } |

**PROFILE**

|  |
| --- |
| **export** **class** ProfileComponent **implements** OnInit {  student;  value = **70**;  showLoader : **boolean** = **true**;  questList = **new** QuestDataItem();  shopItems = **new** ShopItems();  itemsAcquired = **new** Array();  badgesAcquired = **new** Array();  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(Breakpoints.XSmall);  **constructor**(**public** dialog: **MatDialog**, **public** \_ds: **DataService**, **public** \_user: **UserService**, **private** breakpointObserver: **BreakpointObserver**, **public** \_snackbar:**MatSnackBar**) { }  userQuestData = **this**.\_user.getQuestData();  ngOnInit(): **void** {  **this**.InitializeClaimed()  **this**.initializeProfileComponent();  }  InitializeClaimed()  {  **this**.getAchievements()  **this**.getItems()  }  changePassword():**void**{  **const** dialogRef = **this**.dialog.open(ChangePassModalComponent, {  maxWidth: '100vw',  maxHeight: '100vh',  panelClass: 'changepass',  disableClose: **true**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('50%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();    });  }  updateProfilePicture():**void**{  **const** dialogRef = **this**.dialog.open(ChangeProfilePictureComponent, {  panelClass: 'dialogpadding',  maxWidth: '100vw',  maxHeight: '100vh',  autoFocus: **true**,  disableClose: **true**,  data: **this.student**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('40%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  });  }  updateProfile():**void**{  **const** dialogRef = **this**.dialog.open(UpdateProfileModalComponent, {  maxWidth: '100vw',  maxHeight: '100vh',  disableClose: **true**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('98%', 'auto');  } **else** {  dialogRef.updateSize('70%', 'auto');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  **if** (result.data) {  **this**.\_ds.\_httpRequest('updateprofile/'+**this**.\_user.getUserID(),result.data,**1**).subscribe((data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a);  **this**.student = data.payload;  **this**.\_snackbar.open("Profile updated successfully",**null**,{duration:**1500**});  },er=>{  er = **this**.\_user.\_decrypt(er.error.a);  })  }  });  }  initializeProfileComponent(): **void**{  **if**(btoa('profile') **in** sessionStorage){  **this**.student = **this**.\_user.getStudentProfile();  **this**.showLoader = **false**;  }  **else**{  **this**.getStudentInfo();  }  }  getStudentInfo():**void** {  **this**.\_ds.\_httpRequest('studentprofile', {userid:**this.\_user.getUserID**()}, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.\_user.setStudentProfile(dt.payload);  **this**.student = dt.payload;  **this**.showLoader = **false**;  },er=>{  });  }  getAchievements () {  **let** userDataArrayListCode = **this**.userQuestData.badges\_acquired\_fld.split(';')  **this**.questList.quest.filter((questData) => {  userDataArrayListCode.forEach(badgeAcquired => {  questData.questcode\_fld === badgeAcquired ? **this**.badgesAcquired.push(questData) : **null**  });  })  }  getItems() {  **let** itemDataArrayListCode = **this**.userQuestData.items\_acquired\_fld.split(';')  **this**.shopItems.shopItems.filter((shopItems) => {  itemDataArrayListCode.forEach((itemAcquired) => {  shopItems.itemcode\_fld === itemAcquired ? **this**.itemsAcquired.push(shopItems) : **null**  })  })  }  } |

**CHANGE PASSWORD**

|  |
| --- |
| **export** **class** ChangePassModalComponent **implements** OnInit {  hide: **boolean**;  newpasshide:**boolean**;  newpasshide2:**boolean**;  **constructor**(**public** \_ds:**DataService**, **public** \_user: **UserService**,@Inject(MAT\_DIALOG\_DATA) **public** data: **any**,**private** dialogReg: **MatDialogRef**<DialogComponent>,**private** \_snackbar:**MatSnackBar**) { }  ngOnInit(): **void** {    }  changePass(e){  e.preventDefault();  **let** id = e.target[**0**].value;  **let** oldPass = e.target[**1**].value;  **let** newPass = e.target[**2**].value;  **let** confirmPass = e.target[**3**].value;  *// console.log(oldPass,newPass,confirmPass);*  **var** hasUpperCase = /[A-Z]/.test(newPass);  **var** hasLowerCase = /[a-z]/.test(newPass);  **var** hasNumbers = /\d/.test(newPass);  *// var hasNonalphas = /\W/.test(newPass);*    **if** (hasUpperCase == **true** && hasLowerCase == **true** && hasNumbers == **true** && newPass.length >= **8**) {  **if** (newPass == confirmPass) {    **this**.\_ds.\_httpRequest('changepass' ,{ id: **id**,param1: **oldPass**,param2: **newPass**,param3: **confirmPass** }, **1**).subscribe((dt: **any**)=>{  dt = **this**.\_user.\_decrypt(dt.a)  **this**.\_snackbar.open("Password change successfully",**null**,{duration:**1500**});  **this**.closeDialog(**true**);    },er=>{  er = **this**.\_user.\_decrypt(er.error.a);  console.log(er)  **this**.\_snackbar.open("Failed to update password",**null**,{duration:**1500**});  })    }**else**{  **this**.\_snackbar.open('Password Mismatch.', "", {duration:**1500**});  *// console.log("Mismatch password");*    }    }**else**{  **this**.\_snackbar.open('Your password is too weak.', "", {duration:**1500**});  *// console.log("Password too weak");*    }  }  closeDialog(data = **false**) {  **this**.dialogReg.close();  }  } |

**UPDATE PROFILE**

|  |
| --- |
| **export** **class** UpdateProfileModalComponent **implements** OnInit {  region: **any** = [];  province: **any** = [];  city: **any** = [];  brgy: **any**;  **constructor**(**public** \_user: **UserService**, **private** \_ds: **DataService**,@Inject(MAT\_DIALOG\_DATA) **public** data: **any**, **private** dialogReg: **MatDialogRef**<DialogComponent>) { }  student;  regCode;  ngOnInit(): **void** {  **this**.getStudentInfo()  **this**.initializeAddress()  }  initializeAddress(){  **this**.getRegion();  *// this.displayProvince();*  }  getStudentInfo() {  **this**.\_ds.\_httpRequest('studentprofile', {userid:**this.\_user.getUserID**()}, **1**).subscribe((dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.\_user.setStudentProfile(dt.payload);  **this**.student = dt.payload;    },er=>{  er = **this**.\_user.\_decrypt(er.error.a);  });  }  getRegion(){  **this**.\_ds.\_httpRequest('getRegion',{},**1**).subscribe((data:**any**) => {  data = **this**.\_user.\_decrypt(data.a);  **this**.region = data.payload;  })  }  closeDialog(data) {  **this**.dialogReg.close(data);  }  getProvince(e){  **this**.\_ds.\_httpRequest('getProvince',{region\_id:**e.target.value**},**1**).subscribe((data:**any**) => {  data = **this**.\_user.\_decrypt(data.a);  **this**.province = data.payload;  **for**(**let** i = **0**; i < **this**.province.length; i++) {  **if**(**this**.province[i].regCode != e.target.value){  **this**.province.splice(i,**1**);  i--;  }  }  **this**.province.sort(**function**(a, b){  **return** a.provDesc.localeCompare(b.provDesc)  })    })  }  getCity(e){  **this**.\_ds.\_httpRequest('getCity',{province\_id:**e.target.value**},**1**).subscribe((data:**any**) => {  data = **this**.\_user.\_decrypt(data.a);  **this**.city = data.payload;  **for**(**let** i = **0**; i < **this**.city.length; i++) {  **if**(**this**.city[i].provCode != e.target.value){  **this**.city.splice(i,**1**);  i--;  }  }  **this**.city.sort(**function**(a,b){  **return** a.citymunDesc.localeCompare(b.citymunDesc);  })  })  }  getBrgy(e){  **this**.\_ds.\_httpRequest('getBrgy',{municipality\_id:**e.target.value**},**1**).subscribe((data:**any**) => {  data = **this**.\_user.\_decrypt(data.a);  **this**.brgy = data.payload;  **for**(**let** i = **0**; i < **this**.city.length; i++) {  **if**(**this**.brgy[i].citymunCode != e.target.value){  **this**.brgy.splice(i,**1**);  i--;  }  }  **this**.brgy.sort(**function**(a,b){  **return** a.brgyDesc.localeCompare(b.brgyDesc);  })  })  }  updateProfile(e){  e.preventDefault();  **let** mname\_fld = e.target[**1**].value;  **let** extname\_fld = e.target[**3**].value;  **let** birthplace\_fld = e.target[**5**].value;  **let** sex\_fld = e.target[**6**].value;  **let** civilstatus\_fld = e.target[**7**].value;  **let** contactnum\_fld = e.target[**8**].value;  **let** nationality\_fld = e.target[**9**].value;  **let** religion\_fld = e.target[**10**].value;  **let** region\_fld = e.target[**11**].value;  **let** province\_fld = e.target[**12**].value;  **let** city\_fld = e.target[**13**].value;  **let** brgy\_fld = e.target[**14**].value;  **let** house\_fld = e.target[**15**].value;  **let** zipcode\_fld = e.target[**16**].value;  **let** load = {  data:{  mname\_fld:**mname\_fld**,  extname\_fld:**extname\_fld**,  birthplace\_fld:**birthplace\_fld**,  sex\_fld:**sex\_fld**,  civilstatus\_fld:**civilstatus\_fld**,  contactnum\_fld:**contactnum\_fld**,  nationality\_fld:**nationality\_fld**,  religion\_fld:**religion\_fld**,  region\_fld:**region\_fld**,  province\_fld:**province\_fld**,  city\_fld:**city\_fld**,  brgy\_fld:**brgy\_fld**,  house\_fld:**house\_fld**,  zipcode\_fld:**zipcode\_fld**  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getUserID**(),  message: **this.\_user.getFullname**()+' updated profile ',  module: 'profile'  }    }  **this**.dialogReg.close({  data : **load**    });  *// console.log(load);*    }  } |

**UPDATE PROFILE PICTURE**

|  |
| --- |
| **export** **class** ChangeProfilePictureComponent **implements** OnInit {  files: **any** = [];  filepath: **string**;  message: **string** = "Only accepts png file.";  orientation: **any**;  base64Image: **string** = **null**;  croppedImage: **string** = **null**;  **constructor**(  **private** \_snackbar: **MatSnackBar**,  **public** \_ds: **DataService**,  **public** \_user: **UserService**, @Inject(MAT\_DIALOG\_DATA)  **public** data: **any**,  **private** dialogReg: **MatDialogRef**<ChangeProfilePictureComponent>,  **private** \_imageCompress: **NgxImageCompressService**  ) { }  imageUrl: **string** | ArrayBuffer = **this**.\_ds.fileUrl + **this**.\_user.getProfilePicture();  ngOnInit(): **void** { }  compressFile() {  **this**.\_imageCompress.uploadFile().then(({ image, orientation }) => {  **this**.base64Image = image;  **this**.orientation = orientation;  });  }  imageCropped(e: **ImageCroppedEvent**) {  **this**.croppedImage = e.base64;  }  convertToBlob(e) {  **let** arr = e.split(','),  mime = arr[**0**].match(/:(.\*?);/)[**1**],  bstr = atob(arr[**1**]),  n = bstr.length,  u8arr = **new** Uint8Array(n);  **while** (n--) {  u8arr[n] = bstr.charCodeAt(n);  }  **return** **new** File([u8arr], 'Profile.png', { type: **mime** });  }  async updateProfile() {  **this**.\_user.setLoading(**true**);  **this**.\_imageCompress.compressFile(**this**.croppedImage, **this**.orientation, **50**, **100**).then(res => {  **let** imgData = **new** FormData();  **let** img: **string**;  **let** imgToUpload: **File**;  img = res;  imgToUpload = **this**.convertToBlob(img)  imgData.append('file[]', imgToUpload);  **this**.\_ds.\_httpRequest('editProfile/' + **this**.\_user.getUserID(), imgData, **3**).subscribe(async (data: **any**) => {  data = await **this**.\_user.\_decrypt(data.a);  **let** load = {  data: {  profilepic\_fld: **data.payload.filepath**  },  notif: {  id: **this.\_user.getUserID**(),  recipient: **this.\_user.getUserID**(),  message: **this.\_user.getFullname**() + ' profile picture update.',  module: 'profile'  }  }  **this**.\_ds.\_httpRequest('updateprofile/' + **this**.\_user.getUserID(), load, **1**).subscribe((data: **any**) => {  data = **this**.\_user.\_decrypt(data.a);  **this**.\_user.setLoading(**false**);  **this**.\_user.updateUserData({ img: **data.payload**[**0**].profilepic\_fld + "?e=" + Math.random().toString() },'studentuser');  **this**.dialogReg.close();  });  });  });  }  } |

**QUEST**

|  |
| --- |
| **export** **class** QuestComponent **implements** OnInit {  isExtraSmall: **Observable**<BreakpointState> = **this**.breakpointObserver.observe(Breakpoints.XSmall);  questList = **new** QuestDataItem()  */\*\**  *\* Declaration of Experience and levelup*  *\*/*  resetExperienceBar: **number** = **0**;  experienceGain:**number** = **0**;  levelCount: **number** = **0**;  */\*\**  *\* number of points decleration*  *\*/*  pointGained: **number** = **0**;  */\*\**  *\* Progress bar limit declaration*  *\*/*  progressBarLimit: **number** = **0**;  */\*\**  *\**  *\* @param \_user*  *\*/*  **constructor**(**public** \_user:**UserService**,  **public** \_ds:**DataService**,  **public** \_quest:**QuestService**,  **public** dialog:**MatDialog**,  **private** breakpointObserver: **BreakpointObserver**) { }  questDetails :**any**;  questProgress:**any** = **this**.\_user.getQuestData();  *// showLoader: boolean = true;*    ngOnInit(): **void** {  **this**.InitializeStudentAchievements();  }  */\*\**  *\* InitializeStudentAchievements method*  *\*/*  **public** InitializeStudentAchievements() {  */\*\* Get data for mapping of achievements \*/*  **this**.GetAchievementData();  }  */\*\**  *\* GetAchievementData in the database*  *\*/*  **public** GetAchievementData() {    */\*\* For resetting of experience \*/*  **this**.ExperienceResetAndLevelUp(**this**.questProgress.experience\_fld);  */\*\* For points gained \*/*  **this**.PointsGained(**this**.questProgress.points\_fld);  }  */\*\**  *\* Reseting Experience when reach 100%*  *\*/*  **public** ExperienceResetAndLevelUp(experience:**number**) {  */\*\* For reset experience \*/*  **this**.resetExperienceBar = experience % **100**;  */\*\* For leveling \*/*  **this**.levelCount = Math.floor(experience / **100**);  }  */\*\**  *\* PointsGained method*  *\*/*  **public** PointsGained(points:**number**) {  **this**.pointGained = points;  }  */\*\**  *\* ProgressBarControl*  *\*/*  **public** ProgressBarControl(data, count:**number**) {  **if**(data =='QC001' || data =='QC002' || data =='QC003' ||data =='QC004' ){  **if**(**this**.questProgress['cifcnt\_fld'] <= count ){  **return** count == **this**.questProgress['cifcnt\_fld'] ? count/count \* 100: **this.questProgress**['cifcnt\_fld']/count \* **100**  }**else** {  **return** count/count \* **100**;  }  }  **else** **if**(data =='QC005' || data =='QC006' || data =='QC007' ||data =='QC008' ){  **if**(**this**.questProgress['piccnt\_fld'] <= count ){  **return** count == **this**.questProgress['piccnt\_fld'] ? count/count \* 100: **this.questProgress**['piccnt\_fld']/count \* **100**  }**else** {  **return** count/count \* **100**;  }  }  **else** **if**(data =='QC009' || data =='QC010' || data =='QC011' ||data =='QC012' ){  **if**(**this**.questProgress['ciccnt\_fld'] <= count ){  **return** count == **this**.questProgress['ciccnt\_fld'] ? count/count \* 100: **this.questProgress**['ciccnt\_fld']/count \* **100**  }**else** {  **return** count/count \* **100**;  }  }  **else** **if**(data =='QC013' || data =='QC014' || data =='QC015' ||data =='QC016' ){  **if**(**this**.questProgress['taskcnt\_fld'] <= count ){  **return** count == **this**.questProgress['taskcnt\_fld'] ? count/count \* 100: **this.questProgress**['taskcnt\_fld']/count \* **100**  }**else** {  **return** count/count \* **100**;  }  }  **else** **if**(data =='QC017' || data =='QC018' || data =='QC019' ||data =='QC020' ){  **if**(**this**.questProgress['class\_on\_time\_fld'] <= count ){  **return** count == **this**.questProgress['class\_on\_time\_fld'] ? count/count \* 100: **this.questProgress**['class\_on\_time\_fld']/count \* **100**  }**else** {  **return** count/count \* **100**;  }  }  }  questCode(questCode, target){  **if**(questCode =='QC001' || questCode =='QC002' || questCode =='QC003' ||questCode =='QC004' ){  **if**(**this**.questProgress['cifcnt\_fld'] <= target ){  **return** target == **this**.questProgress['cifcnt\_fld'] ? target+'/'+target : **this.questProgress**['cifcnt\_fld']+'/'+target  }**else** {  **return** target+'/'+target;  }  }  **else** **if**(questCode =='QC005' || questCode =='QC006' || questCode =='QC007' ||questCode =='QC008' ){  **if**(**this**.questProgress['piccnt\_fld'] <= target ){  **return** target == **this**.questProgress['piccnt\_fld'] ? target+'/'+target : **this.questProgress**['piccnt\_fld']+'/'+target  }**else** {  **return** target+'/'+target;  }  }  **else** **if**(questCode =='QC009' || questCode =='QC010' || questCode =='QC011' ||questCode =='QC012' ){  **if**(**this**.questProgress['ciccnt\_fld'] <= target ){  **return** target == **this**.questProgress['ciccnt\_fld'] ? target+'/'+target : **this.questProgress**['ciccnt\_fld']+'/'+target  }**else** {  **return** target+'/'+target;  }  }  **else** **if**(questCode =='QC013' || questCode =='QC014' || questCode =='QC015' ||questCode =='QC016' ){  **if**(**this**.questProgress['taskcnt\_fld'] <= target ){  **return** target == **this**.questProgress['taskcnt\_fld'] ? target+'/'+target : **this.questProgress**['taskcnt\_fld']+'/'+target  }**else** {  **return** target+'/'+target;  }  }  **else** **if**(questCode =='QC017' || questCode =='QC018' || questCode =='QC019' ||questCode =='QC020' ){  **if**(**this**.questProgress['class\_on\_time\_fld'] <= target ){  **return** target == **this**.questProgress['class\_on\_time\_fld'] ? target+'/'+target : **this.questProgress**['class\_on\_time\_fld']+'/'+target  }**else** {  **return** target+'/'+target;  }  }  }  openShop():**void** {  **const** dialogRef = **this**.dialog.open(ShopdialogComponent, {  panelClass: 'shop',  maxWidth: '100vw',  maxHeight: '100vh',  disableClose: **true**  });  **const** responsiveDialogSubscription = **this**.isExtraSmall.subscribe(result => {  **if** (result.matches) {  dialogRef.updateSize('100%', 'auto');  } **else** {  dialogRef.updateSize('100%', '100%');  }  });  dialogRef.afterClosed().subscribe(result => {  responsiveDialogSubscription.unsubscribe();  })  }  } |

**QUEST SHOPS**

|  |
| --- |
| **export** **class** ShopdialogComponent **implements** OnInit {  shopList:**any** = **new** ShopItems();  **constructor**(**public** dialog:**MatDialog**,  **public** \_user:**UserService**,  **public** \_quest:**QuestService**,  **public** \_ds:**DataService**,  **public** \_snackbar:**MatSnackBar**,  **private** dialogReg: **MatDialogRef**<ShopdialogComponent>) { }    ngOnInit(): **void** {  **this**.itemChecker()  }  buyItem(itemcode\_fld,cost){  **let** questData = **this**.\_user.getQuestData()    **for** (**let** i = **0**; i < **this**.shopList.shopItems.length; i++) {  **if** (**this**.shopList.shopItems[i].cost\_fld > questData.points\_fld) {  **this**.\_snackbar.open('Not enough points.',**null**,{duration:**1000**})  **return**  }  }  **const** dialogConfig = **new** MatDialogConfig()  dialogConfig.autoFocus = **true**;  **if** (**this**.\_user.isMobile()) {  dialogConfig.maxHeight = '90vw';  dialogConfig.minHeight = 'auto';  dialogConfig.minWidth = '70vw';  dialogConfig.maxWidth = '90vw';  } **else** {  dialogConfig.minHeight = '20vh';  dialogConfig.minWidth = '15vw';  }  dialogConfig.data = {  option:'buyItem',  itemcode\_fld:**itemcode\_fld**  }  **const** dialogRef = **this**.dialog.open(DialogComponent,dialogConfig)  dialogRef.afterClosed().subscribe(result => {  **if** (result) {  **let** splitItems = **new** Array() *//for Splitting the items in the database*  **let** itemsRedeem:**string** = ''; *//data to pass in database*  **let** questData = **this**.\_user.getQuestData(); *//get data from session storage*  **if** (questData.items\_acquired\_fld === "")  {  itemsRedeem = itemcode\_fld  }  **else**  {  splitItems = questData.items\_acquired\_fld.split(';');  splitItems.push(itemcode\_fld) *//push the newly redeem item*  itemsRedeem = splitItems.join(';') *//joined by ';'*  }  **this**.\_ds.\_httpRequest('updategameprogress',{items\_acquired\_fld:**itemsRedeem**},**1**).subscribe(res => {  res = **this**.\_user.\_decrypt(res.a);  **this**.\_snackbar.open('Redeem Successfully!',**null**,{duration:**1000**})  **this**.\_user.updateUserData({items\_acquired\_fld:**itemsRedeem**},'questsdata');  **let** questDataUpdate = **this**.\_user.getQuestData().items\_acquired\_fld  **let** splitQuestUpdate = questDataUpdate.split(';');    **for** (**let** i = **0**; i < **this**.shopList.shopItems.length; i++) {  **let** item = **this**.shopList.shopItems[i].itemcode\_fld  **if** (splitQuestUpdate.includes(item)) {  **this**.shopList.shopItems[i].isRedeemed = **false**  }**else**{  **this**.shopList.shopItems[i].isRedeemed = **true**  }  }  *//Subtract cost to points*  **this**.\_quest.SubtractCostToPoints(cost)  })  }  })    }  *//init checker if item already redeemed*  itemChecker(){  **let** questData = **this**.\_user.getQuestData().items\_acquired\_fld  **let** splitQuestData = questData.split(';');  **for** (**let** i = **0**; i < **this**.shopList.shopItems.length; i++) {  **let** item = **this**.shopList.shopItems[i].itemcode\_fld  **if** (splitQuestData.includes(item)) {  **this**.shopList.shopItems[i].isRedeemed = **false**  }**else**{  **this**.shopList.shopItems[i].isRedeemed = **true**  }  }  console.log(**this**.shopList.shopItems);    }  closeDialog() {  **this**.dialogReg.close();  }  } |

**TODOLIST**

|  |
| --- |
| **export** **class** TodolistComponent **implements** OnInit {  list: **any** = [];  tasklist: **any**=[];  classArr:**any** = [];  showLoader: **boolean** = **true**;  showTable = **false**;  subObj: **any** = []  **constructor**(**public** \_user:**UserService**, **public** \_ds:**DataService**,**private** \_router:**Router**,) { }  ngOnInit(): **void** {  **this**.classArr = **this**.\_user.getStudentSchedule();  **this**.getlist();  }    ngAfterViewInit(): **void**{  console.clear()  }  **public** get activityState(): **typeof** ActivityState {  **return** ActivityState;  }  date\_diff\_indays(date1, date2) {  **var** dt1 = **new** Date(date1);  **var** dt2 = **new** Date(date2);  **return** Math.floor((Date.UTC(dt2.getFullYear(), dt2.getMonth(), dt2.getDate()) - Date.UTC(dt1.getFullYear(), dt1.getMonth(), dt1.getDate()) ) /(**1000** \* **60** \* **60** \* **24**));  }  showName(){  **if**(**this**.\_user.getLoginState()){  **return** **this**.\_user.getFullname();  }  }  getlist(){  **this**.\_ds.\_httpRequest('gettodolist',{param1:**this.\_user.getUserID**()},**1**).subscribe( async (data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a);  **this**.subObj = await data.payload.sub  **this**.list = await **this**.getTodoList(data.payload.**class**, data.payload.act);  console.log(**this**.list);    **this**.showLoader = **false**;  }, er => {    er = **this**.\_user.\_decrypt(er.error.a);  **this**.list = [];  **this**.showLoader = **false**;      });  }  isShowTodoList(li){  **let** actCount = **0**;  **let** doneCount = **0**;  **for** (**let** i = **0**; i < li.length; i++) {  actCount += li[i].activity.length  doneCount += li[i].done\_count  }  **if**(actCount== doneCount){  console.log(actCount);  **return** **true**  }  **return** **false**;  }  viewActivity(classcode){  *// console.log(this.classArr,classcode);*  **for** (**let** i = **0**; i < **this**.classArr.length; i++) {  **if** (**this**.classArr[i].classcode\_fld === classcode) {  **this**.\_user.setSelectedClass(**this**.classArr[i]);    }    }  }  getTodoList(sClass, sActivity){  **let** todoObj: **any** = []  sClass.forEach(element=>{  todoObj.push({  classcode: **element.classcode\_fld**,  subjdesc: **element.subjdesc\_fld**,  activity: **this.getClassActivity**(element.classcode\_fld, sActivity)  })  })  **return** todoObj  }  getClassActivity(classcode, actobj){  **let** actList: **any**    **let** arr: **any** = actobj.filter(item =>  (classcode == item.classcode\_fld)  )  arr.forEach( (element, i) => {  **let** submit = **this**.subObj  .find( (item: **any** ) => {  **return** item.actcode\_fld == element.actcode\_fld && item.classcode\_fld == classcode  })  **if**(arr[i]!=**undefined**){  **if**(submit!=**undefined**){  arr[i] = {  ...arr[i], ...submit  }    }  }  });          **return** arr  }  } |

**OTHER DIALOGS (STUDENT)**

**GLOBAL DIALOG**

|  |
| --- |
| **export** **class** DialogComponent **implements** OnInit {  textcontent: **any**;  newpass = '';  link:**string** = '';  **constructor**(**private** \_ds: **DataService**, **public** \_user:**UserService**, @Inject(MAT\_DIALOG\_DATA) **public** data: **any**, **private** dialogReg: **MatDialogRef**<DialogComponent>) { }  ngOnInit(): **void** {  **if** (**this**.data.newpw == **undefined** || **this**.data.newpw == **null**) {  **return**  }**else**{  **this**.newpass = **this**.data.newpw;  }  **if** (**this**.data.data == **undefined** || **this**.data.data.content\_fld == **undefined** || **this**.data.data.content\_fld == **null**) {  **return**;  }**else**{  **this**.textcontent = **this**.data.data.content\_fld;  }  }    condel(v: **number**) { **this**.dialogReg.close({ data: **v** }); }  editcomment(v: **number**) { **this**.dialogReg.close({ data: **this.textcontent** }); }  turnin(v: **number**) { **this**.dialogReg.close({ data: **v** }); }  unsubmit(v: **number**) { **this**.dialogReg.close({ data: **v** }); }  getFilename(data) { **return** data.replace(/^.\*[\\\/]/, ''); }  returnFileUrl(path) { **return** **this**.\_ds.baseURL + path }  logout(v: **number**){  **this**.dialogReg.close({data:**v**});  }  quizSubmit(v:**number**){  **this**.dialogReg.close({data:**v**});  }  confirmRedeem(){  **this**.dialogReg.close({data:**true**})  }  myreg = /(^|\s)((https?:\/\/)?[\w-]+(\.[\w-]+)+\.?(:\d+)?(\/\S\*)?)/gi  linkFormControl = **new** FormControl('',[Validators.required,Validators.pattern(**this**.myreg)])  addLink(){  **if** (**this**.linkFormControl.hasError('required')) {  **return** 'Link is required.'  }**else** **if**(**this**.linkFormControl.hasError('pattern'))  {  **return** 'Invalid URL.'  }  **this**.dialogReg.close({data:**this.link**})  }  } |

**IMAGE PREVIEW**

|  |
| --- |
| **export** **class** ImagePreviewComponent **implements** OnInit {  **constructor**(  @Inject(MAT\_DIALOG\_DATA) **public** data: **any**,  **private** dialogReg: **MatDialogRef**<ImagePreviewComponent>  ) {}  ngOnInit(): **void** {}  } |

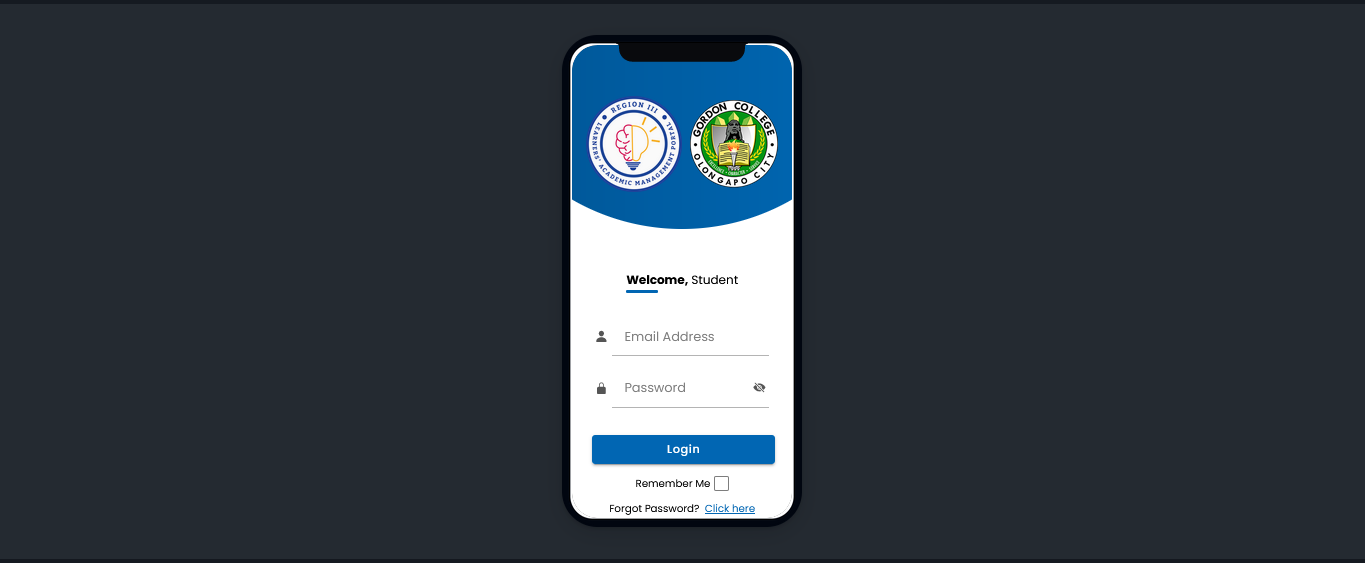
**EDIT POST**

|  |
| --- |
| **export** **class** PostsEditComponent **implements** OnInit {  datatoedit: **any**;  textcontent = **this**.data.data['content\_fld'];  files: **any** = [];  fileButton: **boolean**;  photoButton: **boolean**;  videoButton: **boolean**;  removeFiles: **boolean**;  withfile: **number** = **0**;  filepathtoupload: **any**;  isUploading: **boolean**;  filepath: **FormData**;  disableButton:**number**;  submittedFiles: **any** = [];  filesToBeUpdated: **any** = [];  existingDir: **any**;  split: **any**;  combined: **any**;  editFiles: **any** = [];  **constructor**(**public** \_upload:**UploadsService**,  **private** \_ds: **DataService**,  **public** dialog: **MatDialog**,  **private** \_snackbar:**MatSnackBar**,  **public** \_user:**UserService**,  @Inject(MAT\_DIALOG\_DATA) **public** data: **any**, **private** dialogReg: **MatDialogRef**<DialogComponent>) { }  ngOnInit(): **void** {  **this**.buttondisablerpost();  }    async editpost(v: **number**) {  **this**.\_user.setLoading(**true**);  **if** (**this**.files.length != **0**) {  await **this**.\_upload.Filepath(**this**.files).then((filepath:**any**)=>{  *// console.log(filepath);*  **if** (**this**.existingDir === **null** || **this**.existingDir === **undefined** ) {  **this**.filepathtoupload = filepath;  **this**.\_user.setLoading(**false**);  *// console.log(this.existingDir);*    }**else**{  **this**.split = **this**.existingDir.split(':');  **this**.split.push(filepath);  **this**.combined = **this**.split.join(':');  *// console.log(this.combined);*    **this**.filepathtoupload = **this**.combined;  **this**.\_user.setLoading(**false**);    }    }).**catch**(e=>{  **return**;    });  }**else**{  **this**.filepathtoupload = **this**.existingDir;  **this**.\_user.setLoading(**false**);  *// console.log(this.filepathtoupload);*    }      **this**.dialogReg.close({  data : {  data:{  classcode\_fld:**this.\_user.getSelectedClass**().classcode\_fld,  authorid\_fld:**this.\_user.getUserID**(),  content\_fld:**this.textcontent**,  withfile\_fld:**this.withfile**,  dir\_fld:**this.filepathtoupload**  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message: **this.\_user.getFullname**()+' edited a post on '+**this**.\_user.getSelectedClass().subjdesc\_fld,  module: 'classroom'  }  }  });  }  buttondisablerpost(){  *// console.log(this.data);*    **if** (**this**.data.data['dir\_fld'] != '') {  **this**.existingDir = **this**.data.data['dir\_fld'];  **this**.submittedFiles = **this**.\_upload.splitFilestring(**this**.data.data['dir\_fld']);  *// console.log(this.submittedFiles);*  **for** (**let** i = **0**; i < **this**.submittedFiles.length; i++) {  **var** ext = **this**.getext(**this**.submittedFiles[i].name);  **switch** (ext) {  **case** 'txt':  **case** 'doc':  **case** 'docx':  **case** 'html':  **case** 'htm':  **case** 'odt':  **case** 'pdf':  **case** 'xls':  **case** 'xlsx':  **case** 'ods':  **case** 'pdf':  **case** 'ppt':  **case** 'pptx':  **this**.fileButton = **false**;  **this**.photoButton = **true**;  **this**.videoButton = **true**;  **this**.withfile = **5**;  **break**;  **case** 'm4v':  **case** 'avi':  **case** 'mpg':  **case** 'mp4':  **this**.fileButton = **true**;  **this**.photoButton = **true**;  **this**.videoButton = **false**;  **this**.withfile = **2**;  **break**;  **case** 'jpg':  **case** 'gif':  **case** 'bmp':  **case** 'png':  **case** 'jfif':  **this**.fileButton = **true**;  **this**.photoButton = **false**;  **this**.videoButton = **true**;  **this**.withfile = **1**;  **break**;  }    }  }**else**{  **this**.fileButton = **false**;  **this**.photoButton = **false**;  **this**.videoButton = **false**;  }    }  async getFile(event) {  **let** sum: **number** = **0**;  **for** (**var** i = **0**; i < event.target.files.length; i++) {  sum += event.target.files[i].size;  **this**.files.push(event.target.files[i]);  **var** ext = **this**.getext(**this**.files[i].name);  **switch** (ext) {  **case** 'txt':  **case** 'doc':  **case** 'docx':  **case** 'html':  **case** 'htm':  **case** 'odt':  **case** 'pdf':  **case** 'xls':  **case** 'xlsx':  **case** 'ods':  **case** 'pdf':  **case** 'ppt':  **case** 'pptx':  **this**.fileButton = **false**;  **this**.photoButton = **true**;  **this**.videoButton = **true**;  **this**.withfile = **5**;  **break**;  **case** 'm4v':  **case** 'avi':  **case** 'mpg':  **case** 'mp4':  **this**.fileButton = **true**;  **this**.photoButton = **true**;  **this**.videoButton = **false**;  **this**.withfile = **2**;  **break**;  **case** 'jpg':  **case** 'gif':  **case** 'bmp':  **case** 'png':  **case** 'jfif':  **this**.fileButton = **true**;  **this**.photoButton = **false**;  **this**.videoButton = **true**;  **this**.withfile = **1**;  **break**;  }  **this**.\_upload.getFile(**this**.files)  }  **if** (sum > **30000000**) {  **this**.\_snackbar.open((sum / **1048576**).toFixed(**2**).toString() + "File to big error",**null**,{duration:**1500**})  *// console.log((sum / 1048576).toFixed(2).toString() + "File to big error");*  **this**.files = [];  (<HTMLInputElement>document.getElementById('files')).value = '';  }**else**{  (<HTMLInputElement>document.getElementById('files')).value = '';  }  }  async deletefile(path){  **let** load = {  dir\_fld:**path**  }  **return** **new** Promise((resolve,rejects)=>{  **this**.\_ds.\_httpRequest('deletefile', load, **1**).subscribe((data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a);  resolve(data.status.remarks);  },er=>{  er = **this**.\_user.\_decrypt(er.error.a);  **return** **new** Promise(()=>{  rejects (er.status.remarks)  })  })  })  }  deletefileconfirmation(i,option){  **let** condel  **const** dialogRef = **this**.dialog.open(DialogComponent, {  data: {  option: **option**,  isConfirmed: **condel**  },  });  dialogRef.afterClosed().subscribe(result => {    **if** (result.data) {  **this**.unlinkedFile(i);  }  });  }  async unlinkedFile(i){  **let** newFiles:**any** = [];  **this**.\_user.setLoading(**true**)  await **this**.deletefile(**this**.submittedFiles[i].path).then(()=>{  **this**.submittedFiles.splice(i, **1**);  **this**.submittedFiles.map((e)=>{  newFiles.push(e.name+'?'+e.path);  });  newFiles = newFiles.join(':');  *// console.log(newFiles);*  **let** load = {  data:{  withfile\_fld: **newFiles** == '' ? 0 : **this.withfile**,  dir\_fld: **newFiles**  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message: **this.\_user.getFullname**()+' remove a post on '+**this**.\_user.getSelectedClass().subjdesc\_fld,  module: 'classroom'  }  };  **if** (newFiles == '') {  **this**.fileButton = **false**;  **this**.photoButton = **false**;  **this**.videoButton = **false**;  }  **this**.\_ds.\_httpRequest('editpost/'+**this**.data.data['postcode\_fld'], load, **1**).subscribe((data:**any**)=>{  data = **this**.\_user.\_decrypt(data.a);  *// console.log(data);*  **if** (data.payload[**0**].dir\_fld == '') {  **this**.existingDir = **null**;  }**else**{  **this**.existingDir = data.payload[**0**].dir\_fld;  }      **this**.withfile = data.payload[**0**].withfile\_fld;  **this**.\_user.setLoading(**false**);    *// console.log(data);*    },er=>{  er = **this**.\_user.\_decrypt(er.error.a);  *// console.log(er);*    })    });    }  async removeFile(i) {  **this**.files.splice(i, **1**);  **if** (**this**.files.length == **0**) {  **this**.fileButton = **false**;  **this**.photoButton = **false**;  **this**.videoButton = **false**;  }  *// console.log("remove");*    }  getext(data) {  **return** data.split('.').pop();  }  getFileExtension(filename) {  **if** (filename == 'docx') { **return** '#004db3'; }  **if** (filename == 'pptx') { **return** '#c34524'; }  **if** (filename == 'txt') { **return** '#95a5a6'; }  **if** (filename == 'pdf') { **return** '#ad0b00'; }  **else** {  **return** '#1f6499';  }  }  getfileExt(filename) {  **return** filename.split('.').pop();  }  } |

**RESOURCES PREVIEW**

|  |
| --- |
| **export** **class** ResourcePreviewComponent **implements** OnInit {  linkPath: **any**;  name: **string**;  **constructor**(**public** \_upload:**UploadsService**,**private** \_ds: **DataService**, **public** \_user:**UserService**, @Inject(MAT\_DIALOG\_DATA) **public** data: **any**, **private** dialogReg: **MatDialogRef**<DialogComponent>,**private** sanitizer: **DomSanitizer**) { }  ngOnInit(): **void** {  **this**.getLink(**this**.data.resourceString);  }  getLink(link){  **let** embbed:**string**;  **this**.name = **this**.\_upload.getfileExt(link);  **if** (**this**.name.includes('docx') || **this**.name.includes('xlsx') || **this**.name.includes('pptx')) {  embbed = `https:*//view.officeapps.live.com/op/embed.aspx?src=${link}&embedded=true`*  }  **if** (**this**.name.includes('txt') || **this**.name.includes('pdf') || **this**.name.includes('csv')) {  embbed = link;  }  **if** (**this**.name.includes('jpg') || **this**.name.includes('jpeg') || **this**.name.includes('png')) {  embbed = link;  }  *// console.log(embbed);*    **if** (embbed != **undefined**) {  **this**.linkPath = **this**.sanitizer.bypassSecurityTrustResourceUrl(embbed);  }**else**{  **this**.linkPath = **null**  }    *// this.linkPath = this.sanitizer.bypassSecurityTrustResourceUrl(embbed);*  }  } |

**STUDENT MOBILE VERSION**

****

**Login.ts**

|  |
| --- |
| networkStatus: **any**;  networkListener: **PluginListenerHandle**;  loginCredentials: **any** = {  param1: '',  param2: ''  };  passwordType: **string** = 'password';  passwordIcon: **string** = 'eye-off';  autoInput=**false**;  ionViewWillEnter(){  **this**.getNetWorkStatus()  *// console.clear()*  **let** auth :**boolean** = **false**  **try** {  **const** autologin = **this**.\_storage.get('autologin').then(async (res)=>{  **if**(res != **null**){ auth = await res.isAuth }  });  } **catch** (error) {  console.log(error);  }  **if** (**this**.loginCredentials.param1 == "" || **this**.loginCredentials.param1 == "" || auth) {  **return**  }  **this**.OnSubmit("")  }  async getNetWorkStatus() {  **this**.networkStatus = await Network.getStatus();  *// console.log(this.networkStatus);*  }  async ngOnInit() {  await **this**.\_storage.create();  **try** {  **const** autologin = **this**.\_storage.get('autologin').then((res)=>{  **if**(res !=**null**){  **if**(res.isAuth ==**true**){  **this**.loginCredentials.param1 = res.us;  **this**.loginCredentials.param2= res.ps;  }**else**{ console.log('error') }  }**else**{ console.log('error') } });  } **catch** (error) {  **this**.\_user.presentToast('Remember Password failed!','danger');  } **if**(**this**.\_user.isLoggedIn()){**this**.\_router.navigate(['/main']);} }  hideShowPassword() {  **this**.passwordType = **this**.passwordType === 'password' ? 'text' : 'password';  **this**.passwordIcon = **this**.passwordIcon === 'eye' ? 'eye-off' : 'eye';  }  *//getAcadyear*  async getSettings(){  **const** loading = await **this**.\_loading.create({  message: 'Please wait...',  });  loading.present();  **this**.\_ds.\_httpRequest('getsettings',**null**,**2**).subscribe((res:**any**)=>{  loading.dismiss();  res = **this**.\_user.\_decrypt(res.a);  **this**.\_user.setSettings(res.payload[**0**]);  },er=>{ loading.dismiss(); }); }  *//end of Acadyear*  *//remember me the last user*  async remember(e): Promise<any> {  **try** {  **const** result = await **this**.\_storage.set('isStay', e.target.checked);  **this**.autoInput = result;  } **catch** (reason) {console.log(reason);}  }  *//end of remember me*  async passwordChange(){  **const** modal = await **this**.\_modal.create({  component: **PasswordChangePage**,  });  await modal.present();  }  *//submit*  async OnSubmit(e){  e.preventDefault();  await **this**.getSettings();  **if**(**this**.loginCredentials.param1 == '' || **this**.loginCredentials.param2 ==''){  **this**.\_user.presentToast('Invalid Credentials!',"danger");  **return**;}  **let** param1 = **this**.loginCredentials.param1;  **let** param2 = **this**.loginCredentials.param2;  **let** param3: **number** = **1**;  **let** param4: **string** = **this**.\_user.isMobile()?'Web Access using Mobile phone':'Web Access using Desktop/Laptop';  **this**.\_ds.\_httpRequest('studentlogin', { param1 , param2, param3, param4}, **2**).subscribe((res: **any**)=>{  res = **this**.\_user.\_decrypt(res.a);  **this**.\_storage.set('autologin',{ us: **this.loginCredentials.param1**, ps:**this.loginCredentials.param2**,isAuth: **this.autoInput**} );  **this**.\_user.setData('user', res.payload);  **this**.\_user.setLoginState();  **this**.\_user.presentToast("Welcome " + **this**.\_user.getFullname(), "dark");  **if**(res.payload['reset'] == **1**){  **this**.passwordChange();  }  **this**.loginCredentials.param1 = ""  **this**.loginCredentials.param2 = ""  **this**.\_router.navigate(['main/tabs/classes']);  }, er => {  **this**.\_user.presentToast('The username/password you entered is invalid!',"danger");  **let** data = **this**.\_user.\_decrypt(er.error.a); }); }  *//end of submit }* |

**Classes**

|  |
| --- |
| color='';  allClasses=[];  classes: **any**[] = [];  mon = []; tue = []; wed = []; thu = []; fri = []; sat = []; sun = [];  sched =[];  searchItem = '';  questProgress :**any** =[];  ngOnInit() {  **this**.color = '#07b559';  **this**.getclasslist();  **this**.getGameProgress();  }  *//get ClassList*  async getclasslist(){  **const** loading = await **this**.\_loading.create({  message: 'Please wait...',  });  loading.present();  **this**.\_ds.\_httpRequest('getclasslist', { userid: **this.\_user.getUserID**(), acadyear:**this.\_user.getAcadYear**(), semester:**this.\_user.getSemester**()}, **1**).subscribe((res: **any**)=>{  loading.dismiss();  res = **this**.\_user.\_decrypt(res.a);  **this**.allClasses = **this**.sortClass(res.payload);  **this**.classes = **this**.allClasses;  **this**.\_user.setStudentSchedule(**this**.allClasses);  }, er => {  loading.dismiss();  **this**.allClasses = [];  **this**.\_user.presentToast("No Classes Yet!","danger");  });}  *//end of ClassList*  *//open Open Educational Resources*  async openOER(e) {  await Browser.open({ url: 'https://chedro3.ched.gov.ph/open-educational-resources/' });  }  *//profile pic*  facultyProfilePic(data){  **if**(data ==" " || data == **null** || data ==""){  **return** "assets/img/profile.jpg";  }**else**{  **return** **this**.\_ds.fileUrl + data;  }}  *//end of profile pic*  *//faculty Name*  facultyName(data){  **if**(data.includes("null")){  **return** "TBA";  }**else**{ **return** data; }  }  *//end of faculty name*  filterClass(ev) {  **let** searchItem = ev.target.value;  **this**.classes = **this**.allClasses.filter((x)=>{  **return** (  x.classcode\_fld.toUpperCase().includes(searchItem.toUpperCase()) || x.subjcode\_fld.toUpperCase().includes(searchItem.toUpperCase()) ||  x.subjdesc\_fld.toUpperCase().includes(searchItem.toUpperCase()) ); });}  sortClass(payload): any {  **let** arrDays: **string**[] = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'];  **let** tmp: **any** = payload;  **let** items: **any**[]=[];  arrDays.forEach(x => {  tmp.forEach(y => {  **let** found=**0**;  **if**(y.day\_fld.includes(x)) {  items.forEach(z=>{  **if**(y.classcode\_fld==z.classcode\_fld) found = **1**;  });  **if**(found!=**1**) items.push(y);  }  });});**return** items}  async getGameProgress(){  **let** load = **this**.\_user.getUserID();  **this**.\_ds.\_httpRequest('gameprogress', load ,**1** ).subscribe(async ( data: **any**) => {  data = await **this**.\_user.\_decrypt(data.a);  **this**.questProgress = data.payload[**0**];  **this**.\_user.setQuestData(**this**.questProgress);  }, er => {  **this**.presentToast("No Classes Yet!","danger");  });}  viewClass(data){  **this**.\_user.setSelectedClass(data);  **this**.\_router.navigate(['/sub/tabs/classfeed']);  }  async viewProfile(e){  **this**.\_router.navigate(['main/tabs/profile',{ id:'classes' }]);  }  viewSchedule(){  **this**.\_router.navigate(['main/tabs/schedule']);  }  async viewMessage(e){  **const** modal = await **this**.\_modal.create({  component: **MessagingPage**,  });  await modal.present();  }  async viewNotification(e){  **const** modal = await **this**.\_modal.create({  component: **NotificationPage**,  componentProps: { value: **123** }  });  await modal.present();  } |

**Activities**

|  |
| --- |
| classobject = []; resourceObject = [];limitResources = [];  activities = []; topicObj = [];  disabledBtn = **false**;  showResource = **false**;  datenow = **new** Date;  refresh = **0**;  isResources = **false**;  isActivities = **true**;  selectedSlide: **any**;  segment = **0**;  sliderOption = {  initialSlide : **0**,  slidesPerView: **1**,  speed: **400**  }  ngOnInit() {  **this**.classobject =**this**.\_user.getSelectedClass();  }  ionViewWillEnter(){  **this**.topicObj = []; **this**.resourceObject = [];  **this**.limitResources = []; **this**.activities = [];  **this**.getResource();  **this**.getActivity();  **this**.getTopic();}  getResource(){  **this**.\_ds.\_httpRequest('getreslist',{ classcode:**this.classobject**['classcode\_fld'] },**1**).subscribe((res:**any**) =>{  res = **this**.\_user.\_decrypt(res.a);  **for** (**let** index = **0**; index < res.payload.length; index++) {  **this**.refresh = **0**;  **if**(res.payload[index].datesched\_fld <= **this**.\_user.getDateToday() || res.payload[index].datesched\_fld == **null** ){  **this**.resourceObject.push({data:**res.payload**[index], show:**false** });}}  },er=>{  **this**.\_user.presentToast('No Upload Resources!','danger'); **this**.limitResources = [];});}  async getActivity(){  **const** loading = await **this**.\_loading.create({  message: 'Please wait...',  });  loading.present();  **this**.\_ds.\_httpRequest('getcpost',{ classcode:**this.classobject**['classcode\_fld'] ,type:''},**1**).subscribe((res:**any**)=>{  res = **this**.\_user.\_decrypt(res.a);  loading.dismiss();  **for** (**let** index = **0**; index < res.payload.length; index++) {  **if**(res.payload[index].datesched\_fld == **null**){  res.payload[index].datesched\_fld = **this**.\_user.getDateToday();  }  **if**(res.payload[index].recipient\_fld.includes(**this**.\_user.getUserID()) && (res.payload[index].datesched\_fld <= **this**.\_user.getDateToday()) ){  **this**.activities.push(res.payload[index]) }}  },er=>{ loading.dismiss(); **this**.activities = []; })}  getTopic(){  **let** load = {  classcode:**this.\_user.getSelectedClass**().classcode\_fld  }  **this**.\_ds.\_httpRequest('gettopic',load,**1**).subscribe((res:**any**)=>{  res = **this**.\_user.\_decrypt(res.a)  **for** (**let** index = **0**; index < res.payload.length; index++) {  **this**.topicObj.push({data: **res.payload**[index], show:**false**}) ;  }  **this**.\_user.setTopic(res.payload);  },er=>{  er = **this**.\_user.\_decrypt(er.error.a);  **this**.topicObj = []; }) }  async resTab(e){  **this**.isResources = **true**;  **this**.isActivities = **false**;  }  async actTab(e){  **this**.isResources = **false**;  **this**.isActivities = **true**;  }  doRefresh(event) {  **if**(**this**.refresh == **0**){  **this**.ionViewWillEnter();  }  setTimeout(() => {  event.target.complete();  }, **2000**);  }  async preview(e,data){  **const** modal = await **this**.\_modal.create({  component: **PreviewPage**,  componentProps: { preview: **data** }  });  await modal.present();}  splitFilestring(filestring: **string**) {  **let** arr1 = filestring.split(':');  **let** filearray: { name: **string**; link: **string** }[] = [];  **for** (**let** i = **0**; i < arr1.length; i++) {  **let** arr2 = arr1[i].split('?');  filearray.push({ name: **arr2**[**0**], link: **this.\_ds.fileUrl** + arr2[**1**] }); } **return** filearray; }  *//get profile pic*  getProfilePic(data){  **if**(data ==" " || data == **null** || data ==""){  **return** "assets/img/profile.jpg";  }**else**{  **return** data;} }  *//end of get profile pic*  back(){  **this**.\_router.navigate(['main/tabs/classes']);  }  async viewActivities(e,a){  **const** modal = await **this**.\_modal.create({  component: **ViewActivitiesPage**,  componentProps: { activity: **a** }  });  await modal.present();  }  async viewQuiz(e){  **const** modal = await **this**.\_modal.create({  component: **ViewQuizzesPage**,  });  await modal.present();  }  async segmentChange(ev){  await **this**.selectedSlide.slideTo(**this**.segment);  }  slideShange(slides: **IonSlides**){  **this**.selectedSlide = slides;  slides.getActiveIndex().then(index =>{  **this**.segment = index; }) }  async classinfo(){  **const** alert = await **this**.alertControl.create({  subHeader: **this.classobject**['subjdesc\_fld'],  message: 'Classcode : ' + **this**.classobject['classcode\_fld'] + '<br>' +  **this**.classobject['day\_fld'] +'<br>'+ **this**.classobject['starttime\_fld'] + ' - ' + **this**.classobject['endtime\_fld'],  buttons: [{text : 'OK', cssClass: 'defaultbtn', }] });  await alert.present();  await alert.onDidDismiss();  }  getFileExtension(filename) {  **if** (filename == 'pdf') {  **return** '#EA462E';  } **else** **if** (filename == 'docx') {  **return** '#2D5292'; }  **else** **if** (filename == 'ppt') {  **return** '#CA4223'; }  **else** **if** (filename == 'zip') {  **return** '#B23333'; }  **else** **if** (filename == 'txt') {  **return** '#546A7A';  } **else** { **return** '#222'; } }  getfileExt(filename) {  **return** filename.split('.').pop();  } }  Classlist.page.ts  classobject=[]; instructor = []; classList = [];  ngOnInit() {  **this**.classobject =**this**.\_user.getSelectedClass();  **this**.getMembers();  }  getMembers(){  **let** load = {  data:{  classcode:**this.\_user.getSelectedClass**().classcode\_fld,  acadyear:**this.\_user.getAcadYear**(),  semester:**this.\_user.getSemester**()  }}  **this**.\_ds.\_httpRequest('getmembers',load,**1**).subscribe((res:**any**)=>{  res = **this**.\_user.\_decrypt(res.a);  **this**.instructor = res.payload.instructor;  **this**.classList = res.payload.student;  **this**.classList.sort((a, b) => a.lname\_fld.localeCompare(b.lname\_fld))  }); }  *//get profile pic*  getProfilePic(data){  **if**(data ==" " || data == **null** || data ==""){  **return** "assets/img/profile.jpg";  }**else**{  **return** **this**.\_ds.fileUrl + data;  } }  *//end of get profile pic*  *//faculty Name*  facultyName(data){  **if**(data.includes("null")){  **return** "TBA";  }**else**{**return** data; }}  *//end of faculty name*  back(){  **this**.\_router.navigate(['main/tabs/classes']);  }  async classinfo(){  **const** alert = await **this**.alertControl.create({  subHeader: **this.classobject**['subjdesc\_fld'],  message: 'Classcode : ' + **this**.classobject['classcode\_fld'] + '<br>' +  **this**.classobject['day\_fld'] +'<br>'+ **this**.classobject['starttime\_fld'] + ' - ' + **this**.classobject['endtime\_fld'],  buttons: [{text : 'OK', cssClass: 'defaultbtn', }]  }); await alert.present(); await alert.onDidDismiss();} } |

**Classfeed**

|  |
| --- |
| classobject = []; classFeed= []; classpost = [];  textLimit = **180**;  img = ['jpg','gif','bmp','png','jpeg'];  video = ['avi','mpg','m4v','mp4','wmv'];  filext = ['txt', 'doc' , 'docx' ,'html' ,'htm' ,'odt' ,'pdf' ,'xls', 'xlsx' ,'ods', 'pdf' ,'pptx', 'ppt','sql'];  index = **0**; refresh = **0**;  ngOnInit() {  **this**.classobject =**this**.\_user.getSelectedClass();  }  ionViewWillEnter(){  **this**.classFeed = [];  **this**.getClassFeed();  }  async getClassFeed(){  **const** loading = await **this**.\_loading.create({  message: 'Please wait...',  });  loading.present();  **this**.\_ds.\_httpRequest('getcpost',{classcode:**this.classobject**['classcode\_fld'],type: 'post'},**1**).subscribe((res)=>{  res = **this**.\_user.\_decrypt(res.a);  loading.dismiss();  **this**.classFeed = **this**.\_user.\_convert(res.payload);  **this**.refresh = **0**;  **this**.classpost = **this**.addfiles(**this**.classFeed);  **for** (**let** index = **0**; index < **this**.classpost.length; index++) {  **if**(**this**.classpost[index].poll\_fld.length != **0**){  **let** load ={  filepath: **this.classpost**[index].dir\_fld,  postcode: **this.classpost**[index].postcode\_fld  }  **this**.\_ds.\_httpRequest('getpoll',load,**1**).subscribe((res:**any**)=>{  res = **this**.\_user.\_decrypt(res.a);  **this**.classpost[index].options = res.payload.options;  **this**.classpost[index].respcode = res.payload.respcode;  },er=>{ console.log(er) })}}  },er=>{  loading.dismiss();  **this**.classpost = []; }); }  disable(){  **this**.\_user.presentToast('Disabled Poll!','danger');  }  trimString(**string**, length) {  **return** **string**.length > length ?  **string**.substring(**0**, length) + '...' :  **string**;  }  addfiles(array) {  array.forEach(e => {  e.images\_fld = [];  e.video\_fld = [];  e.attachment\_fld = [];  e.poll\_fld = [];  e.pollchoices\_fld = [];  e.pollresponse\_fld = [];  e.respcode\_fld = '';  e.showMore = **false**;  e.youtube\_fld = [];  e.content\_fld = e.content\_fld;  e.authorid\_fld == **this**.\_user.getUserID() ? e.author\_owner = **true** : e.author\_owner = **false**;  **if** (e.withfile\_fld >= **1**) {  **let** files = [];  **if**(e.dir\_fld != **null**){  files = **this**.splitFilestring(e.dir\_fld);  }  files.forEach(f => {  *// console.log(f)*  **if** (**this**.img.includes(**this**.getext(f.name))) {  e.images\_fld.push(f);  }  **if** (**this**.video.includes(**this**.getext(f.name))) {  e.video\_fld.push(f);  }  **if** (**this**.filext.includes(**this**.getext(f.name))) {  e.attachment\_fld.push(f);  }  **if** (**this**.getext(f.name) == 'json') {  e.poll\_fld.push(f);  }});}  **if**(e.content\_fld == **null**){ e.content\_fld = ''; }  **if**(e.content\_fld.includes('youtube.com') || e.content\_fld.includes('google.com/file/d/')){  e.youtube\_fld = **this**.getVideoEmbedURL(e.content\_fld);  }**else** {  e.youtube\_fld = **null**;  }  **if** ((e.withfile\_fld == **3** || e.withfile\_fld == **4**)) {  **let** pollObj = **this**.\_upload.splitPollContent(e);  e.content\_fld = pollObj.content\_fld;  e.pollchoices\_fld = pollObj.pollchoices\_fld;  **let** load = {  postcode: **e.postcode\_fld**,  filepath: {  options: [] } }  e.pollchoices\_fld.map((chc) => {  load.filepath.options.push({ "choice": chc, "votes": **0**, "respondents": [] });  });  **this**.\_commonservice.commonSubscribe('getpoll', load, **1**)  .then(async (dt: **any**) => {  e.pollresponse\_fld = await dt.payload.options  e.respcode\_fld = dt.payload.respcode;  e.pollresponse\_fld.map(poll => {  poll.respondents.map(e => {  **if** (e.studnum\_fld === **this**.\_user.getUserID()) {  poll.userpollvoted = **true**;  } }) }) }); } }); **return** array;  }  *// Addedd by Melner*  getVideoEmbedURL(content: **string**): any {  **let** replacePattern1 = /(\b(https?|ftp):\/\/[-A-Z0-9+&@#\/%?=~\_|!:,.;]\*[-A-Z0-9+&@#\/%=~\_|])/gim;  **let** replacedText = content.replace(replacePattern1, '<\*>$1<\*>');  **let** content\_arr: **string**[] = replacedText.split('<\*>');  **let** a: **any** = content\_arr.filter((x) => {  **if** (x.includes('http')) {  **return** x}; });  **for** (**let** i = **0**; i < a.length; i++) {  **if** (a[i].includes('youtube.com')) {  a[i] = a[i].replace('watch?v=', 'embed/');;  }  **else** {  a[i] = a[i].replace('view?usp=sharing', 'preview');  }  a[i] = **this**.\_sanitizer.bypassSecurityTrustResourceUrl(a[i]);  } **return** a; }  async setAnswers(e,index,post, postIndex){  **let** load = {  data:{  postcode\_fld: **post**['postcode\_fld'],  studnum\_fld: **this.\_user.getUserID**(),  response\_fld: **index**,  respcode\_fld: **post**['respcode\_fld']  },  filepath: {  options: **post.pollresponse\_fld**,  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message: **this.\_user.getFullname**()+' reponded in a poll on '+**this**.\_user.getSelectedClass().subjdesc\_fld,  module: 'classroom'  } }  **this**.\_ds.\_httpRequest('answerpoll', load, **1**).subscribe((data: **any**) => {  data = **this**.\_user.\_decrypt(data.a);  **this**.classpost[postIndex].pollresponse\_fld.map( (poll: **any**) => {  poll.userpollvoted = **false**;  poll.respondents.map( (e, indx) =>{  **if** (e.studnum\_fld === **this**.\_user.getUserID()) {  poll.respondents.splice(indx, **1**)  poll.votes -= **1** } }) })  **this**.classpost[postIndex].pollresponse\_fld[index].userpollvoted = **true**  **this**.classpost[postIndex].pollresponse\_fld[index].votes += **1**  **this**.classpost[postIndex].pollresponse\_fld[index].respondents.push({  fullname: **this.\_user.getFullname**(),  profilepic\_fld: **this.\_user.getProfilePic**(),  studnum\_fld: **this.\_user.getUserID**()  }) });  sanitize(data){  **return** **this**.\_sanitizer.bypassSecurityTrustResourceUrl(data);  }  async addpost(e){  **let** modal = await **this**.\_modal.create({  component: **AddPostPage**,  });  modal.onDidDismiss().then((dataReturned) => {  **if**(dataReturned.data == 'exit'){**return**;}  **let** addpost = **this**.addfiles(dataReturned.data)  **this**.classpost.unshift(**this**.\_user.\_convert(addpost[**0**]));  });  **return** await modal.present();  }  *//get profile pic*  getProfilePic(data){  **if**(data ==" " || data == **null** || data ==""){  **return** "assets/img/profile.jpg";  }**else**{  **return** **this**.\_ds.fileUrl + data; } }  *//end of get profile pic*  doRefresh(event) {  **if**(**this**.refresh == **0**){  **this**.getClassFeed();  **this**.refresh++;  }  setTimeout(() => {  event.target.complete();  }, **2000**);  }  async preview(e,data){  **const** modal = await **this**.\_modal.create({  component: **PreviewPage**,  componentProps: { preview: **data** } });  await modal.present(); }  async viewComments(e,data,i, module){  **let** modal = await **this**.\_modal.create({  component: **CommentsPage**,  componentProps:{actionCode: **data**,  module: **module**},  cssClass: ''  });  modal.onDidDismiss().then((dataReturned) => {  **this**.classpost[i].commentcount = dataReturned.data;  }); **return** await modal.present(); }  async presentPopover(ev,f,i,option){  **const** popover = await **this**.\_popover.create({  component: **PopoverPage**,  componentProps: { classfeed: **f** ,  option: **option**,  index: **i**  },  cssClass: 'my-popover-class',  mode:'ios',  event: **ev**,  translucent: **true**  });  await popover.present();  popover.onDidDismiss().then((dataReturned) => {  **if**(dataReturned.role == 'backdrop' || dataReturned.data == 'exit'){**return**;}  **if**(dataReturned.role == **null**){  **this**.classpost.splice(dataReturned.data,**1**);  **return**; }  **let** post = **this**.addfiles(dataReturned.data);  **for** (**let** index = **0**; index < **this**.classpost.length; index++) {  **if**(**this**.classpost[index].postcode\_fld == dataReturned.data[**0**].postcode\_fld){  **this**.classpost[index] = post[**0**]; } } });  **return** await popover.present(); }  splitFilestring(filestring: **string**) {  **let** arr1 = filestring.split(':');  **let** filearray: { name: **string**; link: **string**; }[] = [];  **for** (**let** i = **0**; i < arr1.length; i++) {  **let** arr2 = arr1[i].split('?');  filearray.push({ name: **arr2**[**0**], link: **this.\_ds.fileUrl** + arr2[**1**]}); } **return** filearray;  }  getext(data) { **return** data.split('.').pop(); }  back(){  **this**.\_router.navigate(['main/tabs/classes']);  }  getFileExtension(filename) {  **if** (filename == 'pdf') {  **return** '#EA462E';  } **else** **if** (filename == 'docx') {  **return** '#2D5292'; }  **else** **if** (filename == 'ppt') {  **return** '#CA4223'; }  **else** **if** (filename == 'zip') {  **return** '#B23333'; }  **else** **if** (filename == 'txt') {  **return** '#546A7A';  } **else** {**return** '#222'; } }  getfileExt(filename) {  **return** filename.split('.').pop();  }  async classinfo(){  **const** alert = await **this**.alertControl.create({  subHeader: **this.classobject**['subjdesc\_fld'],  message: 'Classcode : ' + **this**.classobject['classcode\_fld'] + '<br>' +  **this**.classobject['day\_fld'] +'<br>'+ **this**.classobject['starttime\_fld'] + ' - ' + **this**.classobject['endtime\_fld'],  buttons: [{text : 'OK', cssClass: 'defaultbtn', }]});  await alert.present();  **let** result = await alert.onDidDismiss(); } } |

**Comments**

|  |
| --- |
| @ViewChild('myInput', {**static**: **true**}) myInput: **ElementRef**;  @Input() actionCode = ""; @Input() module ='';  postcomments = []; editComment = []; reply = [];  classobject = [];  textLimit = **100**;  comments = ''; recipientcode = ''; edit = '';  replyComment=''; editPost = ''; editClasscode = '';  stats = 'add';  editIndex = **0**;  replystatus = **false**;  replies = **false**;  index = **0**;  editreplyIndex = **0**;  ngOnInit() {  **this**.classobject =**this**.\_user.getSelectedClass();  **this**.getcomments();  }  trimString(**string**, length) {  **return** **string**.length > length ?  **string**.substring(**0**, length) + '...' :  **string**;  }  *//comments*  async getcomments(){  **const** loading = await **this**.\_loading.create({  message: 'Please wait...',  });  loading.present();  **this**.\_ds.\_httpRequest('getccomment',{ acode:**this.actionCode** },**1**).subscribe((res:**any**)=>{  loading.dismiss();  res = **this**.\_user.\_decrypt(res.a);  **this**.postcomments = **this**.\_user.\_convert(res.payload);  **for** (**let** index = **0**; index < **this**.postcomments.length; index++) {  **this**.postcomments[index].showMore = **false**;  **this**.postcomments[index].replies = **false**;  **this**.postcomments[index].writeReply = **false**;  }  },er=>{loading.dismiss()  **this**.postcomments = []; }); }  *//end of comments*  *//add and edit comments*  async addComments(e,d){  *//add*  **if**(d == 'add'){ **this**.addComment();  *//edit*  }**else** **if**(d == 'edit'){ **this**.editComments();  }}  *//end add and edit*  async addComment(){  **if**(**this**.comments.trim() == ''){  **this**.\_user.presentToast('Please Write Something!','danger');  **return**; }  **const** loading = await **this**.\_loading.create({  message: 'Please wait...',  });  loading.present();  **let** load = {  data:{  classcode\_fld:**this.\_user.getSelectedClass**().classcode\_fld,  authorid\_fld : **this.\_user.getUserID**(),  actioncode\_fld: **this.actionCode**,  content\_fld : **this.comments.trim**(),  withfile\_fld: **0**,  dir\_fld : ''  },notif:{  id: **this.\_user.getUserID**(),  recipient: **this.classobject**['empcode\_fld'],  message: **this.\_user.getFullname**() + ' created a comment on ' + **this**.classobject['subjdesc\_fld'] ,  module: **this.module**  }}  **this**.\_ds.\_httpRequest('addcomment',load,**1**).subscribe((res:**any**)=>{  loading.dismiss();  res = **this**.\_user.\_decrypt(res.a);  res.payload[**0**].showMore = **false**;  res.payload[**0**].replies = **false**;  res.payload[**0**].reply = [];  res.payload[**0**].replycount = **0**;  **this**.postcomments.unshift(res.payload[**0**]);  **this**.comments = '';  **this**.myInput.nativeElement.style.height = '40px';  **this**.\_user.presentToast('Successfully Posted!','dark');  **if**(res.payload[**0**].actioncode\_fld.includes("CP")){  **this**.\_quest.UpdateCountAchievement(Crud.ADD,fldNames.ciccnt\_fld);  }},er=>{  loading.dismiss();  **this**.\_user.presentToast('Invalid Post!','danger');  er = **this**.\_user.\_decrypt(er.error.a);  });}  async editComments(){  **if**(**this**.comments.trim() == ''){  **this**.\_user.presentToast('Please Write Something!','danger');  **return**; }  **const** loading = await **this**.\_loading.create({  message: 'Please wait...',  });  loading.present();  **let** load = {  data:{  classcode\_fld:**this.editComment**['classcode\_fld'],  authorid\_fld:**this.\_user.getUserID**(),  content\_fld:**this.comments.trim**(),  },notif:{  id: **this.\_user.getUserID**(),  recipient: **this.classobject**['empcode\_fld'],  message: **this.\_user.getFullname**() + ' edited a comment on ' + **this**.classobject['subjdesc\_fld'] ,  module: **this.module**} }  **this**.\_ds.\_httpRequest('editcomm/'+ **this**.editComment['commentcode\_fld'] , load , **1**).subscribe((res) => {  loading.dismiss();  res = **this**.\_user.\_decrypt(res.a);  **if**(**this**.edit == 'comment'){  **this**.postcomments[**this**.editIndex].content\_fld = res.payload[**0**].content\_fld;  }**else**{  **for** (**let** index = **0**; index < **this**.postcomments.length; index++) {  **if**(**this**.postcomments[index].commentcode\_fld == res.payload[**0**].actioncode\_fld){  **this**.postcomments[index].reply[**this**.editIndex].content\_fld = **this**.\_user.\_convert(res.payload[**0**].content\_fld);  } } }  **this**.stats = 'add';  **this**.comments = '';  **this**.\_user.presentToast('Successfully Updated!','dark');  },er=>{  loading.dismiss();  **this**.\_user.presentToast('Invalid Post!','danger');  er = **this**.\_user.\_decrypt(er.error.a); }); }  async editReply(e){  e.preventDefault();  **if**(e.target[**0**].value.trim() == ''){  **this**.\_user.presentToast('Please Write Something!','danger');  **return**; }  **let** load = {  data:{  classcode\_fld:**this.classobject**['classcode\_fld'],  authorid\_fld:**this.\_user.getUserID**(),  content\_fld:**e.target**[**0**].value.trim(),  },notif:{  id: **this.\_user.getUserID**(),  recipient: **this.classobject**['empcode\_fld'],  message: **this.\_user.getFullname**() + ' edited a comment on ' + **this**.classobject['subjdesc\_fld'] ,  module: **this.module**  }}  **this**.\_ds.\_httpRequest('editcomm/'+ **this**.editClasscode , load , **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a);  **this**.postcomments[**this**.editreplyIndex].reply[**this**.editIndex].content\_fld = res.payload[**0**].content\_fld; **this**.postcomments[**this**.editreplyIndex].reply[**this**.editIndex].showEdit = **false**;  **this**.stats = 'add';  **this**.comments = '';  **this**.\_user.presentToast('Successfully Updated!','dark');  },er=>{  **this**.\_user.presentToast('Invalid Post!','danger');  er = **this**.\_user.\_decrypt(er.error.a);  }); }  async addreply(e,comment,i){  **if**(**this**.replyComment.trim() == ''){  **this**.\_user.presentToast('Please Write Something!','danger');  **return**; }  **let** load = {  data:{  classcode\_fld:**this.\_user.getSelectedClass**().classcode\_fld,  authorid\_fld : **this.\_user.getUserID**(),  actioncode\_fld: **comment.commentcode\_fld**,  content\_fld : **this.replyComment.trim**(),  withfile\_fld: **0**,  dir\_fld : ''  },notif:{  id: **this.\_user.getUserID**(),  recipient: **this.classobject**['empcode\_fld'],  message: **this.\_user.getFullname**() + ' replied a comment on ' + **this**.classobject['subjdesc\_fld'] ,  module: **this.module**  } } **this**.\_ds.\_httpRequest('addcomment',load,**1**).subscribe((res:**any**)=>{  res = **this**.\_user.\_decrypt(res.a);  **this**.postcomments[i].replycount +=**1**;  **if** (**this**.postcomments[i].reply == **null**) {  **this**.postcomments[i].reply = [];  **this**.postcomments[i].reply.push(res.payload[**0**]);  } **else** {  **this**.postcomments[i].reply.push(res.payload[**0**]); }  **this**.replyComment = '';  **this**.replystatus = **false**;  **this**.\_user.presentToast('Successfully Posted!','dark');  },er=>{  **this**.\_user.presentToast('Invalid Post!','danger');  er = **this**.\_user.\_decrypt(er.error.a); })}  removeReply(){  **this**.replystatus = **false**;  **this**.stats = 'add';  }  async presentPopover(ev,c,j,i,option){  **const** popover = await **this**.\_popover.create({  component: **PopoverPage**,  componentProps: { classcomment: **c** ,  option: **option**, index: **j**,  postIndex: **I** },  cssClass: 'my-popover-class',  mode:'ios',  event: **ev**,  translucent: **true**  });  await popover.present();  popover.onDidDismiss().then((dataReturned) => {  **if**(dataReturned.role == 'backdrop'){**return**;}  **if**(dataReturned.data.option == 'comment'){  *//edit comment*  **this**.editIndex = dataReturned.data.postIndex;  **this**.stats = 'edit';  **this**.edit = dataReturned.data.option;  **this**.editComment = dataReturned.data.data;  **this**.comments = **this**.editComment['content\_fld'];  }**else** **if**(dataReturned.data.option == 'reply comment'){    *//edit reply*  **this**.editIndex = dataReturned.data.index;  **this**.editreplyIndex = dataReturned.data.postIndex;  **this**.stats = 'edit';  **this**.edit = dataReturned.data.option;  **this**.editPost = **this**.postcomments[dataReturned.data.postIndex].reply[dataReturned.data.index].content\_fld;  **this**.editClasscode = **this**.postcomments[dataReturned.data.postIndex].reply[dataReturned.data.index].commentcode\_fld;  **this**.postcomments[dataReturned.data.postIndex].reply[dataReturned.data.index].showEdit = **true**;  }**else** **if**(dataReturned.data.option == 'delete comment'){  *//delete comment*  **this**.postcomments.splice(dataReturned.data.postIndex,**1**);  }**else** **if**(dataReturned.data.option == 'reply delete'){  *//delete reply*  **this**.postcomments[dataReturned.data.postIndex] = dataReturned.data.data;  }});  **return** await popover.present();  }  resize() {  **if**(**this**.comments.length > **20** || **this**.comments.length < **20**){ **this**.myInput.nativeElement.style.height = '40px'; }  **this**.myInput.nativeElement.style.height = **this**.myInput.nativeElement.scrollHeight + 'px';  }  *//get profile pic*  getProfilePic(data){  **if**(data ==" " || data == **null** || data ==""){  **return** "assets/img/profile.jpg";  }**else**{ **return** **this**.\_ds.fileUrl + data; }  }  *//end of get profile pic*  back(){  **if**(**this**.postcomments.length == **0**){  **this**.\_modal.dismiss(**0**);  }**else**{  **this**.\_modal.dismiss(**this**.postcomments.length); }}}  Studentworks.page.ts  classobject = []; studWorks = []; activities = [];  index = **0**;  ngOnInit() {  **this**.classobject =**this**.\_user.getSelectedClass();  **this**.getActivity();  }  ionViewWillEnter(){  **this**.studWorks = [];  **this**.getStudentWorks();  }  async getStudentWorks(){  **const** loading = await **this**.\_loading.create({  message: 'Please wait...',  });  loading.present();  **this**.\_ds.\_httpRequest('getstudworks',{userid: **this.\_user.getUserID**(),classcode:**this.classobject**['classcode\_fld'],actcode:'',type:""},**1**).subscribe((res:**any**)=>{  res = **this**.\_user.\_decrypt(res.a);  loading.dismiss();  **this**.studWorks = res.payload;  *// console.log(this.studWorks);*  },er=>{  loading.dismiss();  **this**.studWorks = [];  });  }  getActivity(){  **this**.\_ds.\_httpRequest('getcpost',{classcode:**this.classobject**['classcode\_fld'] ,type:''},**1**).subscribe((res:**any**)=>{  res = **this**.\_user.\_decrypt(res.a);  **this**.activities = res.payload;  },er=>{  **this**.activities = [];  }) }  async viewActivity(e,actcode){  **let** act = []  **for** (**let** index = **0**; index < **this**.activities.length; index++) {  **if**(**this**.activities[index].actcode\_fld == actcode){  act = **this**.activities[index]; } }  **const** modal = await **this**.\_modal.create({  component: **ViewActivitiesPage**,  componentProps: {activity : **act** }  });  await modal.present();  }  *//get profile pic*  getProfilePic(data){  **if**(data ==" " || data == **null** || data ==""){  **return** "assets/img/profile.jpg";  }**else**{  **return** data; } }  *//end of get profile pic*  back(){  **this**.\_router.navigate(['main/tabs/classes']);  }  async classinfo(){  **const** alert = await **this**.alertControl.create({  subHeader: **this.classobject**['subjdesc\_fld'],  message: 'Classcode : ' + **this**.classobject['classcode\_fld'] + '<br>' +  **this**.classobject['day\_fld'] +'<br>'+ **this**.classobject['starttime\_fld'] + ' - ' + **this**.classobject['endtime\_fld'],  buttons: [{text : 'OK',cssClass: 'defaultbtn',}] });  await alert.present();  **let** result = await alert.onDidDismiss();} }  ViewActivies.page.ts  @Input() activity = [];  works= []; files=[]; dupefile = []; alreadyuploaded = [];  issubmitted = **0**;  isscored = **0**;  withfile = **0**;  dueDate:**any**;  displayCloseIcon : **boolean** = **true**;  saveBtn: **boolean**;  index = **0**;  filepathtoupload = **undefined**;  downloadfiles = []; updateFile =[]; viewQuiz = [];  classobject= [];  score = ''; quiztotal = '';  ngOnInit() {  **this**.classobject =**this**.\_user.getSelectedClass();  **this**.downloadfiles = **this**.splitFilestring(**this**.activity['filedir\_fld']);  **if**(**this**.activity['type\_fld'] == **1**){  **this**.downloadfiles = [];}  **this**.dupefile = [];}  ionViewWillEnter(){  **this**.works = [];  **this**.alreadyuploaded = [];  **this**.issubmitted = **0**;  **this**.isscored = **0**;  **this**.getWorks();  }  async getWorks(){  **let** load = {  userid:**this.\_user.getUserID**(),  classcode:**this.\_user.getSelectedClass**().classcode\_fld,  actcode:**this.activity**['actcode\_fld'],  type:'act' }  **this**.\_ds.\_httpRequest('getstudworks',load,**1**).subscribe((res:**any**)=>{  res = **this**.\_user.\_decrypt(res.a);  **this**.works = res.payload;  **if**(**this**.activity['type\_fld'] == **0**){  **this**.alreadyuploaded = **this**.splitFilestring(**this**.works[**0**].dir\_fld);  **this**.issubmitted = **this**.works[**0**].issubmitted\_fld;  **this**.isscored = **this**.works[**0**].isscored\_fld;  }**else** **if**(**this**.activity['type\_fld'] == **1**){  **this**.issubmitted = **this**.works[**0**].issubmitted\_fld;  **this**.isscored = **this**.works[**0**].isscored\_fld;  **if**(**this**.isscored == **1**){  **this**.getQuizResult(**this**.works[**0**].dir\_fld); }}  },(er) => {  **this**.works = [];  **let** err = **this**.\_user.\_decrypt(er.error.a);  }) }  async getQuizResult(data){  **const** loading = await **this**.\_loading.create({  message: 'Please wait...',  });  loading.present();  **this**.\_ds.\_httpRequest('getresult', { filepath: **data** }, **1**).subscribe((res: **any**) => {  res = **this**.\_user.\_decrypt(res.a);  loading.dismiss();  **this**.viewQuiz = res.payload;  **this**.score = **this**.viewQuiz['totalscore'];  **this**.quiztotal = **this**.viewQuiz['quiztotal'];  },er=>{  loading.dismiss();})}  *//getFile in the storage*  getFile(event) {  **let** sum: **number** = **0**;  **for** (**var** i = **0**; i < event.target.files.length; i++) {  sum += event.target.files[i].size;  **if**(**this**.dupefile.includes(event.target.files[i].name)){  **this**.presentToast('Files already exist!','danger');  **return**;  }**else**{  **this**.files.push(event.target.files[i]);  **this**.dupefile.push(event.target.files[i].name); } }  **if** (sum > **10000000**) {  **this**.presentToast((sum / **1048576**).toFixed(**2**).toString() + "File to big error",'danger');  **this**.files = [];  }**else**{  **if** (**this**.files.length > **0**) {  **if** (**this**.dueDate < **this**.\_user.getDateToday()) {  **this**.withfile = **2**;  }**else**{**this**.withfile = **1**; }} }  **this**.displayCloseIcon = **false**; }  *//end of getFile*  async Filepath(){  **let** formdata: **FormData**;  formdata = **this**.uploadFile(**this**.files);  **return** **new** Promise((resolve, rejects) => {  **this**.\_ds.\_httpRequest('upload/' + **this**.\_user.getUserID(), formdata, **3**).subscribe((data: **any**) => {  data = **this**.\_user.\_decrypt(data.a);  resolve(data.payload.filepath);  }, er => {  er = **this**.\_user.\_decrypt(er.error.a);  rejects(er.error.a)  });})}  uploadFile(payload) {  **const** formData = **new** FormData();  **for** (**var** i = **0**; i < **this**.files.length; i++) {  formData.append('file[]', **this**.files[i]);  }  formData.append('payload', JSON.stringify(payload));  **return** formData;  }  *//submit work*  async addWork(e,i){  **if**(i == 'Submit'){  **const** loading = await **this**.\_loading.create({  message: 'Please wait...',  });  loading.present();  **if**(**this**.files.length == **0**){  **this**.presentToast("Please add file/s first!",'danger');  **return**;  }  await **this**.Filepath().then((filepath:**any**)=>{  **new** Promise<**void**>(() => {  **let** load = {  data:{  classcode\_fld:**this.\_user.getSelectedClass**().classcode\_fld,  actcode\_fld: **this.activity**['actcode\_fld'],  studnum\_fld:**this.\_user.getUserID**(),  type\_fld:**this.\_user.getSelectedClass**().type\_fld,  dir\_fld:**filepath**,  issubmitted\_fld: **this.withfile**  },notif:{  id: **this.\_user.getUserID**(),  recipient: **this.classobject**['empcode\_fld'],  message: **this.\_user.getFullname**()+' submitted a file on ' + **this**.classobject['subjdesc\_fld'] ,  module: 'activities'  }}  **this**.\_ds.\_httpRequest('addwork',load,**1**).subscribe((res:**any**)=>{  res = **this**.\_user.\_decrypt(res.a);  loading.dismiss();  **this**.works = **this**.\_user.\_convert(res.payload);  **this**.files = [];  **this**.alreadyuploaded = **this**.splitFilestring(**this**.works[**0**].dir\_fld);  **this**.issubmitted = **this**.works[**0**].issubmitted\_fld;  **this**.isscored = **this**.works[**0**].isscored\_fld;  **this**.displayCloseIcon = **true**;  **this**.getWorks();  },er => {  er = **this**.\_user.\_decrypt(er.error.a);  })});}).**catch**(e=>{});  }**else**{  **this**.displayCloseIcon = **false**;  **this**.saveBtn = **true**;  } }  *//end of submit work*  async save(e){  **const** loading = await **this**.\_loading.create({  message: 'Please wait...',  });  loading.present();  **if**(**this**.files.length == **0** && **this**.alreadyuploaded.join()== ""){  **this**.presentToast("Please add file/s first!",'danger');  **return**; }  **if**(!(**this**.files.length == **0**)) {  await **this**.Filepath().then((filepath:**any**)=>{  **new** Promise<**void**>(() => {  **this**.filepathtoupload = filepath;  }) }); }  **for** (**let** index = **0**; index < **this**.alreadyuploaded.length; index++) {  **if**(**this**.alreadyuploaded[index].name == ''){  **this**.alreadyuploaded = [];  }**else**{  **let** name = **this**.alreadyuploaded[index].name;  **let** link = **this**.removeBaseUrl(**this**.alreadyuploaded[index].link);  **this**.alreadyuploaded[index] = name +"?"+ link;  } }  **if**(**this**.filepathtoupload == **undefined**){  **this**.filepathtoupload = '';  **this**.filepathtoupload = **this**.alreadyuploaded.join(":");  }**else** **if**(**this**.alreadyuploaded.join(':') == ''){  **this**.alreadyuploaded = [];  **this**.filepathtoupload = **this**.filepathtoupload;  }**else**{  **this**.filepathtoupload = **this**.filepathtoupload+ ":" + **this**.alreadyuploaded.join(":");  }  **if**(**this**.updateFile.length != **0**){  **for** (**let** index = **0**; index < **this**.updateFile.length; index++)  { **this**.deleteFile(**this**.updateFile[index]); }  **this**.updateFile = []; }  **let** load = {  data:{*// isdeleted\_fld: 1,*  dir\_fld: **this.filepathtoupload**,  isscored\_fld: **this.works**[**0**].isscored\_fld  }  ,notif:{  id: **this.\_user.getUserID**(),  recipient: **this.classobject**['empcode\_fld'],  message: **this.\_user.getFullname**() + ' edited an activity on ' + **this**.classobject['subjdesc\_fld'] ,  module: 'activities' } }  **this**.\_ds.\_httpRequest('editsubmit/'+**this**.works[**0**].submitcode\_fld, load ,**1**).subscribe((res:**any**)=>{  res = **this**.\_user.\_decrypt(res.a);  loading.dismiss();  **this**.dupefile =[];  **this**.alreadyuploaded = **this**.splitFilestring(res.payload[**0**].dir\_fld)  **this**.files = [];  **this**.filepathtoupload = **undefined**;  **this**.displayCloseIcon = **true**;  **this**.saveBtn = **false**;  },er=>{  er = **this**.\_user.\_decrypt(er.error.a);  loading.dismiss() }) }  async deleteFile(path){  **let** load = { dir\_fld:**path** }  **return** **new** Promise((resolve,rejects)=>{  **this**.\_ds.\_httpRequest('deletefile',load,**1**).subscribe((res:**any**)=>{  res = **this**.\_user.\_decrypt(res.a);  resolve(res.status.remarks);  },er=>{  er = **this**.\_user.\_decrypt(er.error.a);  **return** **new** Promise(()=>{  rejects(er.status.remarks)  }) }) }) }  async removeFile(i,data) {  **if**(data == 'add'){  **this**.files.splice(i, **1**);  **this**.dupefile.splice(i,**1**);  }**else**{  **this**.dupefile.splice(i,**1**);  **this**.updateFile.push(**this**.removeBaseUrl(**this**.alreadyuploaded[i].link));  **this**.alreadyuploaded.splice(i,**1**); }  **if** (**this**.files.length == **0** && **this**.alreadyuploaded.join() == '' ){  **this**.files = [];  **this**.alreadyuploaded = []; } }  removeBaseUrl(data){  **let** substring = data.replace(**this**.\_ds.fileUrl, "");  **return** substring;  }  async preview(e,data){  **const** modal = await **this**.\_modal.create({  component: **PreviewPage**,  componentProps: { preview: **data** }  });  await modal.present();  }  splitFilestring(filestring: **string**) {  **let** arr1 = filestring.split(':');  **let** filearray: { name: **string**; link: **string** }[] = [];  **for** (**let** i = **0**; i < arr1.length; i++) {  **let** arr2 = arr1[i].split('?');  **this**.dupefile.push(arr2[**0**]);  filearray.push({ name: **arr2**[**0**], link: **this.\_ds.fileUrl** + arr2[**1**] });  } **return** filearray; }  back(){ **this**.\_modal.dismiss(); }  async ViewSubmit() {  **const** modal = await **this**.\_modal.create({  component: **DrawerPage**,  componentProps: { activity: **this.activity** },  cssClass: 'Submissions',  swipeToClose: **true**,  presentingElement: **await** **this**.\_modal.getTop()  });  modal.onDidDismiss().then((dataReturned) => {  **if**(dataReturned){ **this**.getWorks();} });  **return** await modal.present(); }  async viewComments(e,data,module){  **const** modal = await **this**.\_modal.create({  component: **CommentsPage**,  componentProps: { actionCode: **data**,  module: **module** } });  modal.onDidDismiss().then((dataReturned) => {  **this**.activity['commentcount'] = dataReturned.data;  }); **return** await modal.present(); }  async startQuiz(e,data){  **if**(**this**.activity['withfile\_fld'] == **3** ){  **if**(data == 'start'){  **const** modal = await **this**.\_modal.create({  component: **SlidePage**,  componentProps: { quiz: **this.activity** ,page: 'quiz' }  });  modal.onDidDismiss().then((dataReturned) => {  **this**.quiztotal = dataReturned.data.quiztotal;  **this**.score = dataReturned.data.score  **this**.getWorks();  }); **return** await modal.present(); }  }**else** **if**( **this**.activity['withfile\_fld'] == **2**){  **this**.quizpreview();}}  async quizpreview(){  **const** modal = await **this**.\_modal.create({  component: **ViewQuizzesPage**,  componentProps: { quiz: **this.viewQuiz**, status: **this.activity**['withfile\_fld'],works: **this.works** }  }); await modal.present(); }  getFileExtension(filename) {  **if** (filename == 'pdf') {  **return** '#EA462E';  } **else** **if** (filename == 'docx') {  **return** '#2D5292'; }  **else** **if** (filename == 'ppt') {  **return** '#CA4223'; }  **else** **if** (filename == 'zip') {  **return** '#B23333'; }  **else** **if** (filename == 'txt') {  **return** '#546A7A';  } **else** {**return** '#222'; } }  getfileExt(filename) {  **return** filename.split('.').pop(); } } |

**Forums**

|  |
| --- |
| forums=[]; allForums =[];  search = ''; searchItem = '';  textLimit = **130**;  ngOnInit() {  **this**.getForums();  }  async getForums(){  **const** loading = await **this**.\_loading.create({  message:'Please Wait...' });  loading.present();  **this**.\_ds.\_httpRequest('getforumtopic',{},**1**).subscribe((res:**any**)=>{  res = **this**.\_user.\_decrypt(res.a);  **this**.allForums = res.payload;  **this**.forums = **this**.allForums;  **for** (**let** index = **0**; index < **this**.forums.length; index++) {  **this**.forums[index].showMore = **false**;  }loading.dismiss();  }, er => { loading.dismiss();  **let** data = **this**.\_user.\_decrypt(er.error.a);  **this**.\_user.presentToast("No Forum Posts! ","danger");  }); }  async openOER(e) {  await Browser.open({ url: 'https://chedro3.ched.gov.ph/open-educational-resources/' });  }  trimString(**string**, length) {  **return** **string**.length > length ?  **string**.substring(**0**, length) + '...' :  **string**;  }  filterForums(ev) {  **let** searchItem = ev.target.value;  **this**.forums = **this**.allForums.filter((x)=>{  **return** ( x.forumdesc\_fld.toUpperCase().includes(searchItem.toUpperCase()) ||  x.forumtitle\_fld.toUpperCase().includes(searchItem.toUpperCase())  ); }); }  async viewProfile(e){  **this**.\_router.navigate(['main/tabs/profile',{ id:'forum' }]); }  async gotoClasses(e){  **this**.\_router.navigate(['main/tabs/classes']); }  *//profile pic*  facultyProfilePic(data){  **if**(data ==" " || data == **null** || data ==""){  **return** "assets/img/profile.jpg";  }**else**{ **return** **this**.\_ds.fileUrl + data; } }  async viewMessage(e){  **const** modal = await **this**.\_modal.create({  component: **MessagingPage**,  });await modal.present();}  async viewNotification(e){  **const** modal = await **this**.\_modal.create({  component: **NotificationPage**,  }); await modal.present(); }  async viewSubForum(e, data){  **const** modal = await **this**.\_modal.create({  component: **SubforumsPage**,  componentProps: { forum: **data** }  });  await modal.present(); } }  Subforums.page.ts  @Input() forum = [];  searchItem = ''; search ='';  subForum = []; searchForums=[]; allSub = [];  index = **0**;  textLimit = **150**;  ngOnInit() {  **this**.getSubforum();  }  async getSubforum(){  **const** loading = await **this**.\_loading.create({  message:'Please Wait...'  });  loading.present();  **this**.\_ds.\_httpRequest('getsubforum',{forumcode: **this.forum**['forumcode\_fld']},**1**).subscribe((res:**any**)=>{  res = **this**.\_user.\_decrypt(res.a);  loading.dismiss();  **this**.allSub = res.payload;  **this**.subForum = **this**.allSub;  **for** (**let** index = **0**; index < **this**.subForum.length; index++) { **this**.subForum[index].showMore = **false**; }  }, er => { loading.dismiss();  **let** data = **this**.\_user.\_decrypt(er.error.a);  **this**.subForum = [];  **this**.\_user.presentToast("No Sub-Forum Post!","danger");}); }  filterSubForums(ev) {  **let** searchItem = ev.target.value;  **this**.subForum = **this**.allSub.filter((x)=>{  **return** (  x.subdesc\_fld.toUpperCase().includes(searchItem.toUpperCase()) ||  x.subtitle\_fld.toUpperCase().includes(searchItem.toUpperCase()) ); }); }  trimString(**string**, length) {  **return** **string**.length > length ?  **string**.substring(**0**, length) + '...' :  **string**;  }  back(){ **this**.\_modal.dismiss(); }  *//get profile pic*  getProfilePic(data){  **if**(data ==" " || data == **null** || data ==""){  **return** "assets/img/profile.jpg";  }**else**{ **return** **this**.\_ds.fileUrl + data; } }  *//end of get profile pic*  async viewAddSubForum(e){  **const** modal = await **this**.\_modal.create({  component: **SubforumAddPage**,  componentProps: { forum: **this.forum**}  });  modal.onDidDismiss().then((dataReturned) => {  **if**(dataReturned.role == 'backdrop'){ **return**; }  **this**.getSubforum();  }); **return** await modal.present(); }  async viewContent(e,data,i){  **const** modal = await **this**.\_modal.create({  component: **SubforumContentsPage**,  componentProps: { subForum: **data**,index : **i** }  });  modal.onDidDismiss().then((dataReturned) => {  **this**.subForum[dataReturned.data.index].replies = dataReturned.data.count;  }); **return** await modal.present();}} |

**Subforums Content**

|  |
| --- |
| @ViewChild('myInput') myInput: **ElementRef**;  @Input() subForum = [];  @Input() index :**number**;  subForumcontents = [];  reply = ''; content = ''; editSubComment='';  commentCount = **0**; refresh = **0**; postIndex=**0**;  stats='add';  replystatus = **false**;  recipientcode = ''; postId = '';  ngOnInit() {  **this**.getContent();  }  async getContent(){  **const** loading = await **this**.\_loading.create({  message:'Please Wait...'  }); loading.present();  **this**.\_ds.\_httpRequest('getsubcontent',{ subcode: **this.subForum**['subcode\_fld'] },**1**).subscribe((res:**any**)=>{  loading.dismiss();  res = **this**.\_user.\_decrypt(res.a);  **this**.refresh = **0**;  **this**.subForumcontents = res.payload;  **for** (**let** index = **0**; index < **this**.subForumcontents.length; index++) {  **this**.subForumcontents[index].editReply = **false**; }  **this**.commentCount = **this**.subForumcontents.length;  }, er => {loading.dismiss();  **let** data = **this**.\_user.\_decrypt(er.error.a);  **this**.subForumcontents = [];  **this**.\_user.presentToast(data.status.message,"danger"); }); }  async ForumComment(e,stats){  **if**(**this**.content == ''){  **this**.\_user.presentToast('Please Enter Something!','danger');  **return**; }  **if**(stats =='add'){  **this**.addcomment();  }**else** **if**(stats =='edit'){  }**else** **if**(stats =='reply'){  **this**.addreply(); } }  addcomment(){  **let** load = {  data:{  authorid\_fld:**this.\_user.getUserID**(),  content\_fld: **this.content.trim**(),  subcode\_fld: **this.subForum**['subcode\_fld']  },notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getUserID**(),  message: **this.\_user.getFullname**()+' added a comment in a thread.',  module: 'forums'}}  **this**.\_ds.\_httpRequest('addreply',load, **1**).subscribe((res:**any**)=>{  res = **this**.\_user.\_decrypt(res.a);  **this**.content = '';  **this**.getContent();  **this**.myInput.nativeElement.style.height = '40px';  **this**.\_user.presentToast('Add Comment Successfully!',"dark");  **this**.\_quest.UpdateCountAchievement(Crud.ADD,fldNames.cifcnt\_fld);  }, er => { er = **this**.\_user.\_decrypt(er.error.a);  **this**.\_user.presentToast("Invalid Comment","danger");}); }  editcomment(e){  **let** load = {  data:{  authorid\_fld:**this.\_user.getUserID**(),  content\_fld: **e.target**[**0**].value,  isdeleted\_fld:**0**  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getUserID**(),  message: **this.\_user.getFullname**()+' edited a comment in a thread.',  module: 'forums' } }  **this**.\_ds.\_httpRequest('editreply/'+ **this**.postId , load , **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a);  **this**.getContent();  **this**.stats = 'add';  **this**.subForumcontents[**this**.postIndex].editReply = **false**;  **this**.subForumcontents[**this**.postIndex].content\_fld = e.target[**0**].value;  **this**.\_user.presentToast('Successfully Updated!',"dark");  },er=>{ er = **this**.\_user.\_decrypt(er.error.a);  **this**.\_user.presentToast(er.status.message,"danger");}); }  addreply(){  **if**(**this**.content.trim() == ''){  **this**.\_user.presentToast('Please Write Something!','danger');  **return**; }  **let** load = {  data:{  actioncode\_fld:**this.recipientcode**,  authorid\_fld:**this.\_user.getUserID**(),  content\_fld: **this.content**,  isdeleted\_fld:**0**  },  notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getUserID**(),  message: **this.\_user.getFullname**()+' added a reply in a thread.',  module: 'forums' }}}  resize() {  **if**(**this**.content.length > **20** || **this**.content.length < **20**){  **this**.myInput.nativeElement.style.height = '40px';  }  **this**.myInput.nativeElement.style.height = **this**.myInput.nativeElement.scrollHeight + 'px';  }  *//get profile pic*  getProfilePic(data){  **if**(data ==" " || data == **null** || data ==""){  **return** "assets/img/profile.jpg";  }**else**{ **return** **this**.\_ds.fileUrl + data; }}  *//end of get profile pic*  async presentPopover(ev,c,option,i){  **const** popover = await **this**.\_popover.create({  component: **PopoverPage**,  componentProps: { forumContent: **c** ,  option: **option**, index: **I** },  cssClass: 'my-popover-class',  mode:'ios', event: **ev**,  translucent: **true**  }); await popover.present();  popover.onDidDismiss().then((dataReturned) => {  **if**(dataReturned.role == 'backdrop'){**return**;}  **if**(dataReturned.data.option == 'edit'){  **this**.postId = dataReturned.data.data.contentcode\_fld;  **this**.postIndex = dataReturned.data.postIndex;  **this**.subForumcontents[**this**.postIndex].editReply = **true**;  }**else** **if**(dataReturned.data.option == 'delete'){  **this**.replystatus = **false**;  **this**.content = '';  **this**.getContent();  **this**.commentCount--;  }**else** **if**(dataReturned.data.option == 'reply'){  **this**.content = '';  **this**.reply = dataReturned.data.data  **this**.recipientcode = **this**.reply['contentcode\_fld'];  **this**.replystatus = **true**;  **this**.stats = 'reply'; }});  **return** await popover.present(); }  removeReply(){  **this**.replystatus = **false**;  **this**.stats = 'add'; }  back(){  **this**.\_modal.dismiss({count: **this.commentCount**, index: **this.index**}); }  doRefresh(event) {  **if**(**this**.refresh == **0**){  **this**.getContent();  **this**.refresh++;  }  setTimeout(() => {  event.target.complete();  }, **2000**); }} |

**Add Subforums**

|  |
| --- |
| @Input() forum = [];  title=''; desc = '';  async addSubForum(){  **if**(**this**.title == '' || **this**.desc == ''){  **this**.\_user.presentToast('Please Enter Something!','danger');  **return**; }  **const** loading = await **this**.\_loading.create({  message:'Please Wait...'  }); loading.present();  **let** load = {  data:{  authorid\_fld: **this.\_user.getUserID**(),  forumcode\_fld:**this.forum**['forumcode\_fld'],  isapproved\_fld: **0**,  subdesc\_fld: **this.desc**,  subtitle\_fld: **this.title**  },notif:{  id:**this.\_user.getUserID**(),  recipient: **this.\_user.getUserID**(),  message: **this.\_user.getFullname**()+' requested to open a subforum on' + **this**.forum['forumtitle\_fld'],  module: 'forums'}}  **this**.\_ds.\_httpRequest('addsubforum',load,**1**).subscribe((res:**any**)=>{  loading.dismiss();  res = **this**.\_user.\_decrypt(res.a);  **this**.\_modal.dismiss(res.payload);  **this**.\_user.presentToast('Successfully Submitted but need to Approved!','dark');  },er=>{  **this**.\_modal.dismiss();  loading.dismiss();  er = **this**.\_user.\_decrypt(er.error.a);  **this**.\_user.presentToast('Invalid Post','danger'); }) }  trimString(**string**, length) {  **return** **string**.length > length ?  **string**.substring(**0**, length) + '...' :  **string**;  }  *//get profile pic*  getProfilePic(data){  **if**(data ==" " || data == **null** || data ==""){  **return** "assets/img/profile.jpg";  }**else**{ **return** **this**.\_ds.fileUrl + data; } }  *//end of get profile pic*  back(){ **this**.\_modal.dismiss('','backdrop'); }} |

**ELibrary**

|  |
| --- |
| isactive: **boolean** = **false**;  dupefile = []; bookObject: **any** = [];  isTiled: **boolean** = **true**;  bookTitle: **string**;  bookObjectIndividual;  searchList;  ngOnInit() { **this**.getBooks(); }  toggleview() { **this**.isTiled = !**this**.isTiled; }  Search() {  **let** book: **any**;  **let** author: **any**;  book = **this**.searchList.filter((res: **any**) =>  res.title\_fld  .toLocaleLowerCase()  .match(**this**.bookTitle.toLocaleLowerCase())  );  author = **this**.searchList.filter((res: **any**) =>  res.author\_fld  .toLocaleLowerCase()  .match(**this**.bookTitle.toLocaleLowerCase())  );  **if** (**this**.bookTitle == '') {  **this**.getBooks();  } **else** **if** (**this**.bookTitle != '') {  **if** (book == '') {  **this**.bookObject = author;  } **else** { **this**.bookObject = book; }}}  getBooks() {  **this**.\_ds.\_httpRequest('getbooks', '', **1**).subscribe(  (dt: **any**) => {  dt = **this**.\_user.\_decrypt(dt.a);  **this**.bookObject = dt.payload;  **this**.searchList = **this**.bookObject; },  (er) => { console.clear();  **this**.bookObject = **null**; } ); }  async viewBook(data) {  **let** array = [];  array = **this**.splitFilestring(data.file);  **const** modal = await **this**.\_modal.create({  component: **PreviewPage**,  componentProps: { preview: **array**[**0**] }, });  await modal.present(); }  splitFilestring(filestring: **string**) {  **let** arr1 = filestring.split(':');  **let** filearray: { name: **string**; link: **string** }[] = [];  **for** (**let** i = **0**; i < arr1.length; i++) {  **let** arr2 = arr1[i].split('?');  **this**.dupefile.push(arr2[**0**]);  filearray.push({  name: **arr2**[**0**],  link: **this.\_ds.booksURL** + arr2[**0**], });}  **return** filearray; }  messaging.page.ts  @ViewChild('myInput') myInput: **ElementRef**;  @ViewChild('scrollMe') **private** myScrollContainer: **ElementRef**;  @ViewChild(IonContent, { **static**: **false** }) content: **IonContent**;  **private** unSubscribe = **new** Subject()  ScrollToBottom() {  **this**.content.scrollToBottom(**1500**);  }  data = []; rooms = []; classobject = [];  message =''; selectedRoom = '';  roomMessages: **any**;  webSocket: **WebSocket**;  openMessage = **false**;  sender = []; unreadMessages = [] openChat = [];  refresh = **0**;  messageForms: **FormGroup**;  ngOnInit() {  **this**.classobject = **this**.\_user.getSelectedClass();  **this**.messageForms = **this**.fb.group({  usermassage: ['', Validators.required], });}  ionViewWillEnter(){  **this**.rooms = [];  **this**.unreadMessages  **this**.getUnread(); }  ngOnDestroy(): **void** {  **this**.unSubscribe.unsubscribe()  }  doRefresh(event) {  **if**(**this**.refresh == **0**){  **this**.getRooms();  **this**.getMessage(event);  **this**.refresh++;  }  setTimeout(() => {  event.target.complete();  }, **2000**);  }  getRooms(){  **this**.\_ds.\_httpRequest('getmembers', {data:{classcode: **this.classobject**['classcode\_fld'], acadyear: **this.\_user.getSettings**().acadyear\_fld, semester: **this.\_user.getSettings**().sem\_fld}}, **1**).subscribe( async (res: **any**)=>{  **const** result = await **this**.\_user.\_decrypt(res.a)  **this**.rooms = await **this**.setRoom(result.payload);  },er=>{ }) }  setRoom(data: **any**) {  **const** student = data.student;  **const** instructor = data.instructor;  **const** socketid = [];  **const** tempRoom = [];  instructor.forEach(element => {  tempRoom.push({  id: **element.empcode\_fld**,  fullname:` ${element.lname\_fld}, ${element.fname\_fld}`,  image: **element.profilepic\_fld**,  unread: +**this**.getUnreadCount(element.empcode\_fld)  }) });  student.forEach(element => {  **if**(element.studnum\_fld != **this**.\_user.getUserID()){  tempRoom.push({  id: **element.studnum\_fld**,  fullname:` ${element.lname\_fld}, ${element.fname\_fld}`,  image: **element.profilepic\_fld**,  unread: +**this**.getUnreadCount(element.studnum\_fld)  } ) } });  **function** compare( a, b ) {  **if** ( a.unread > b.unread ){  **return** -**1**; }  **if** ( a.unread < b.unread ){  **return** **1**; }  **return** **0**; }  tempRoom.sort(compare)  **return** tempRoom; }  getUnreadCount(id){  **let** count = **0**  **this**.unreadMessages.forEach(element => {  **if**(element.authorid\_fld == id){ count += **1**}  }); **return** count }  getUnread() {  **this**.\_ds.\_httpRequest('getunread', { roomcode: **this.classobject**['classcode\_fld'], recipient: **this.\_user.getUserID**() }, **1**).subscribe(async(res:**any**) => {  **const** data = await **this**.\_user.\_decrypt(res.a);  **this**.unreadMessages = await data.payload;  **this**.getRooms();  }, async er => {  er = await **this**.\_user.\_decrypt(er.error.a)  **this**.getRooms(); }) }  isMessageMine(dt){  **if**(dt.receiver == **this**.\_user.getUserID() && dt.classcode==**this**.classobject['classcode\_fld'] && dt.sender == **this**.selectedRoom['id']){  **this**.roomMessages.push(dt['data'])  setTimeout(() =>{  **this**.myScrollContainer.nativeElement.scroll({  top: **this.myScrollContainer.nativeElement.scrollHeight**,  left: **0**,  behavior: 'smooth'});  }, **200**); }  **else**{console.log('different room') }  }  async chatBody(e,data){  **this**.roomMessages = [];  **this**.selectedRoom = data;  **this**.getMessage(data);  **this**.openMessage = **true**;  **this**.openChat = data;  **this**.unreadMessages.forEach(message => {  **if**(message.roommember\_fld.toString().includes(data['id'].toString())){  **this**.\_ds.\_httpRequest("readmsg", { roomcode: **this.classobject**['classcode\_fld'], recipient: **this.\_user.getUserID**(), authorid: **data**['id'] }, **1**).subscribe(async (res:**any**) => {  **const** data = await **this**.\_user.\_decrypt(res.a);  **this**.unreadMessages = data.payload  **this**.\_user.setMessageBadge(**this**.unreadMessages.length)  },async (er:**any**) => {  **const** err = await **this**.\_user.\_decrypt(er.error.a);  **this**.\_user.setMessageBadge('')  **if**(err.status.message == "No data found") {  **this**.unreadMessages = []; }})}})}  sendChat(e){  **if**(**this**.message == "" || **this**.message.length == **0**){  **this**.\_user.presentToast("Please Enter Something!","danger");  **return**; }  **const** load = {  authorid\_fld: **this.\_user.getUserID**(),  content\_fld: **this.message**,  roomcode\_fld: **this.classobject**['classcode\_fld'],  roommember\_fld: `${**this**.\_user.getUserID()} - ${**this**.selectedRoom['id']}`  }  **this**.\_ds.\_httpRequest('addmsg', load, **1**).subscribe(async (res:**any**)=>{  **const** result = await **this**.\_user.\_decrypt(res.a)  **this**.roomMessages.push(result.payload[**0**]);  **this**.message = "";  **this**.ScrollToBottom();  }),er=>{ console.log(er); } }  getMessage(e){  **this**.\_ds.\_httpRequest('getmsg', {roomcode: **this.classobject**['classcode\_fld'],sender: **this.\_user.getUserID**(), receiver: **e.id**}, **1**).subscribe(async (res: **any**)=>{  **const** result = **this**.\_user.\_decrypt(res.a);  **this**.roomMessages = await result.payload  **this**.refresh = **0**;  **this**.ScrollToBottom();  },er=>{ console.log(er); }) }  resize() {  **if**(**this**.message.length > **20** || **this**.message.length < **20**){  **this**.myInput.nativeElement.style.height = '40px';  }  **this**.myInput.nativeElement.style.height = **this**.myInput.nativeElement.scrollHeight + 'px';  }  *//get profile pic*  getProfilePic(data){  **if**(data ==" " || data == **null** || data ==""){  **return** "assets/img/profile.jpg";  }**else**{ **return** **this**.\_ds.fileUrl + data; }}  *//end of get profile pic*  back(data){  **if**(data == 'back'){  **this**.openMessage = **false**;  **this**.ionViewWillEnter();  }**else** **if**(data == 'exit'){  **this**.\_router.navigate(['main/tabs/classes']); } }  async classinfo(){  **const** alert = await **this**.alertControl.create({  subHeader: **this.classobject**['subjdesc\_fld'],  message: 'Classcode : ' + **this**.classobject['classcode\_fld'] + '<br>' +  **this**.classobject['day\_fld'] +'<br>'+ **this**.classobject['starttime\_fld'] + ' - ' + **this**.classobject['endtime\_fld'],  buttons: [{ text : 'OK', cssClass: 'defaultbtn', }] });  await alert.present();  **let** result = await alert.onDidDismiss(); } } |

**News**

|  |
| --- |
| searchItem = ''; search = '';  news = []; latest = []; searchNews= []; allNews = [];  index = **0**;  isSearch = **false**;  ngOnInit() {  **this**.getNews();  }  async getNews(){  **const** loading = await **this**.\_loading.create({  message: 'Please wait...',  });  loading.present().then(()=>{  **this**.\_ds.\_httpRequest('getannounce',{dept:**this.\_user.getDepartment**()},**1**).subscribe((res:**any**)=>{  loading.dismiss();  res = **this**.\_user.\_decrypt(res.a);  **this**.allNews = res.payload;  **this**.news = **this**.allNews;  }, er => { loading.dismiss();  **this**.latest = [];  **let** data = **this**.\_user.\_decrypt(er.error.a); **this**.\_user.presentToast(data.status.message,"danger");});});}  async openOER(e) {  await Browser.open({ url: 'https://chedro3.ched.gov.ph/open-educational-resources/' });  }  filterNews(ev) {  **let** searchItem = ev.target.value;  **this**.news = **this**.allNews.filter((x)=>{  **return** (  x.content\_fld.toUpperCase().includes(searchItem.toUpperCase()) || x.title\_fld.toUpperCase().includes(searchItem.toUpperCase()) ); }); }  noPicture(data){  **if** (data == '' || data == **null** || data == ' ') {  **return** 'assets/img/noimage.png';  } **else** { **return** **this**.\_ds.fileUrl + data; } }  *//profile pic*  facultyProfilePic(data){  **if**(data ==" " || data == **null** || data ==""){  **return** "assets/img/profile.jpg";  }**else**{ **return** **this**.\_ds.fileUrl + data; } }  errorImage(index){  **this**.news[index].imgdir\_fld = '';  }  async viewProfile(e){  **this**.\_router.navigate(['main/tabs/profile',{ id:'news' }]);  }  async gotoClasses(e){  **this**.\_router.navigate(['main/tabs/classes']);  }  async viewNews(e, data) {  **const** modal = await **this**.\_modal.create({  component: **ViewNewsPage**,  componentProps: { news: **data** }  });await modal.present(); }  async viewMessage(e){  **const** modal = await **this**.\_modal.create({  component: **MessagingPage**,  componentProps: { value: **123** }  }); await modal.present(); }  async viewNotification(e){  **const** modal = await **this**.\_modal.create({  component: **NotificationPage**,  }); await modal.present(); } } |

**View News**

|  |
| --- |
| @Input() news = [];  textLimit= **230**;  showMore =**false**;  ngOnInit() {  }  trimString(**string**, length) {  **return** **string**.length > length ?  **string**.substring(**0**, length) + '...' :  **string**;  }  noPicture(data){  **if** (data == '' || data == **null** || data == ' ') {  **return** 'assets/img/noimage.png';  } **else** { **return** **this**.\_ds.fileUrl + data; } }  back(){ **this**.\_modal.dismiss(); }  async viewImage(data){  **if**(data['imgdir\_fld'] == ""){  **this**.\_user.presentToast('No Image','danger');  **return**; }  **const** modal = await **this**.\_modal.create({  component: **PreviewPage**,  componentProps: { newsPreview: **data** }  }); await modal.present(); }  viewProfile(){  **this**.\_router.navigate(['main/tabs/profile']);  }  viewSchedule(){  **this**.\_router.navigate(['main/tabs/schedule']);  }  async viewMessage(e){  **const** modal = await **this**.\_modal.create({  component: **MessagingPage**,  }); await modal.present(); }  async viewNotification(e){  **const** modal = await **this**.\_modal.create({  component: **NotificationPage**,  }); await modal.present(); }  async viewComments(e,data){  **const** modal = await **this**.\_modal.create({  component: **NewsCommentsPage**,  componentProps: { news: **data** }  });  modal.onDidDismiss().then((dataReturned) => {  **this**.news['commentcount'] = dataReturned.data;  }); await modal.present(); } }  Profile.page.ts  student = [];  from = '';  ngOnInit() {**this**.getStudentInfo(); }  ionViewWillEnter(){  **this**.from = **this**.\_route.snapshot.paramMap.get('id');  }  getStudentInfo() {  **this**.\_ds.\_httpRequest('studentprofile', {userid:**this.\_user.getUserID**()}, **1**).subscribe((res: **any**) => {  res = **this**.\_user.\_decrypt(res.a);  **this**.student = res.payload;  },er=>{ **let** data = **this**.\_user.\_decrypt(er.error.a);});}  logout(){ **this**.\_user.setLogOut();  **this**.\_router.navigateByUrl('login'); }  *//get profile pic*  getProfilePic(data){  **if**(data ==" " || data == **null** || data ==""){  **return** "assets/img/profile.jpg";  }**else**{ **return** **this**.\_ds.fileUrl + data; } }  *//end of get profile pic*  async back(e){  **this**.\_router.navigate(['main/tabs/' + **this**.from]); }  async editProfile(){  **const** modal = await **this**.\_modal.create({  component: **ProfileUpdatePage**,  componentProps: {student :**this.student**} });  modal.onDidDismiss().then((data) => {  **this**.getStudentInfo(); });  **return** await modal.present(); }  async changeProfilePic(e){  **const** modal = await **this**.\_modal.create({  component: **ChangeProfilePicturePage**,  }); **return** await modal.present(); }  async viewChangePassword(e){  **const** modal = await **this**.\_modal.create({  component: **PasswordChangePage**, });  modal.onDidDismiss().then((data) => {  **if**(data.data=='logout'){ **this**.\_user.setLogOut();  **this**.\_router.navigateByUrl('login'); } });  **return** await modal.present(); } }  profile-update.page.ts  @Input() student = [];  region: **any** = ""; province: **any** = ""; city: **any** = ""; brgy: **any**;  gender :**number**; civilstatus: **number**;  selectedRegion : **string**= "";  selectedProvince : **string** = "";  selectedCity : **string** = ""; selectedBrgy : **string** = "";  frmGroup:**FormGroup**;  ngOnInit() {  **this**.initializeAddress( 'r', 'getRegion', **null**);  **this**.frmGroup = **this**.\_fb.group({  fname\_fld: [**this**.student[**0**].fname\_fld],  mname\_fld: [**this**.student[**0**].mname\_fld],  lname\_fld: [**this**.student[**0**].lname\_fld],  extname\_fld: [**this**.student[**0**].extname\_fld],  sex\_fld: [**this**.student[**0**].sex\_fld.toString(), Validators.required],  civilstatus\_fld: [**this**.student[**0**].civilstatus\_fld.toString(), Validators.required],  religion\_fld: [**this**.student[**0**].religion\_fld, Validators.required],  nationality\_fld: [**this**.student[**0**].nationality\_fld, Validators.required],  birthdate\_fld:[**this**.student[**0**].birthdate\_fld, Validators.required],  birthplace\_fld: [**this**.student[**0**].birthplace\_fld, Validators.required],  contactnum\_fld: [**this**.student[**0**].contactnum\_fld, Validators.required],  region\_fld: [**this**.student[**0**].region\_fld, Validators.required],  province\_fld: [**this**.student[**0**].province\_fld, Validators.required],  city\_fld: [**this**.student[**0**].city\_fld, Validators.required],  brgy\_fld: [**this**.student[**0**].brgy\_fld, Validators.required],  house\_fld: [**this**.student[**0**].house\_fld,Validators.required],  zipcode\_fld: [**this**.student[**0**].zipcode\_fld, Validators.required],  }) }  regions: **any**[] = []; provinces: **any** = []; cities: **any** = [];  barangays: **any** = [];  initializeAddress(t: **string**, api: **string**, value: **string**): **void** {  **if**(api != ''){ **switch** (t) { **case** 'r':  **this**.\_ds.\_httpRequest(api, {}, **1**).subscribe((res: **any**)=> {  res = **this**.\_user.\_decrypt(res.a);  **this**.regions = res.payload;  **this**.getActiveAddress('r');  },er=>{console.log(er); }); **break**;  **case** 'p':  **this**.\_ds.\_httpRequest(api, { region\_id:**value** }, **1**).subscribe((res: **any**)=> {  res = **this**.\_user.\_decrypt(res.a);  **this**.provinces = res.payload;  **this**.getActiveAddress('p');  },er=>{ console.log(er); }); **break**;  **case** 'c':  **this**.\_ds.\_httpRequest(api, { province\_id:**value** }, **1**).subscribe((res: **any**)=> {  res = **this**.\_user.\_decrypt(res.a);  **this**.cities = res.payload;  **this**.getActiveAddress('c');  },er=>{console.log(er); }); **break**;  **case** 'b':  **this**.\_ds.\_httpRequest(api, { municipality\_id:**value** }, **1**).subscribe((res: **any**)=> {  res = **this**.\_user.\_decrypt(res.a);  **this**.barangays = res.payload;  **this**.barangays.filter((x: **any**, i: **number**)=>{  **if**(x.brgyCode == **this**.student[**0**].brgy\_fld) {  **this**.brgy = x.brgyDesc; } });  },er=>{ console.log(er); }); **break**;  **default**: **break**; } } }  selectNewAddress(t: **string**, api: **string**, value: **any**){  **switch** (t) {**case** 'p':  **this**.\_ds.\_httpRequest(api, { region\_id:**value**}, **1**).subscribe((res: **any**)=> {  res = **this**.\_user.\_decrypt(res.a);  **this**.provinces = res.payload;  **for**(**let** i = **0**; i < **this**.provinces.length; i++) {  **if**(**this**.provinces[i].regCode != value.toString()){  **this**.provinces.splice(i,**1**); i--; } }  **this**.provinces.sort(**function**(a, b){  **return** a.provDesc.localeCompare(b.provDesc })  },er=>{console.log(er); }); **break**;    **case** 'c':  **this**.\_ds.\_httpRequest(api, { province\_id:**value** }, **1**).subscribe((res: **any**)=> {  res = **this**.\_user.\_decrypt(res.a);  **this**.cities = res.payload;  **for**(**let** i = **0**; i < **this**.cities.length; i++) {  **if**(**this**.cities[i].provCode != value.toString()){  **this**.cities.splice(i,**1**); i--; } }  **this**.cities.sort(**function**(a,b){ **return** a.citymunDesc.localeCompare(b.citymunDesc); })  },er=>{console.log(er); }); **break**;  **case** 'b':  **this**.\_ds.\_httpRequest(api, { municipality\_id:**value** }, **1**).subscribe((res: **any**)=> {  res = **this**.\_user.\_decrypt(res.a);  **this**.barangays = res.payload;  **for**(**let** i = **0**; i < **this**.barangays.length; i++) {  **if**(**this**.barangays[i].citymunCode != value.toString()){  **this**.barangays.splice(i,**1**); i--; } }  **this**.barangays.sort(**function**(a,b){  **return** a.brgyDesc.localeCompare(b.brgyDesc); })  },er=>{console.log(er); });**break**;  **default**: **break**; } }  getActiveAddress(a: **string**): **void** {  **switch**(a) {**case** 'r':  **this**.regions.filter((x: **any**, i: **number**)=>{  **if**(x.regCode == **this**.student[**0**].region\_fld) {  **this**.initializeAddress('p', 'getProvince', x.regCode);  **this**.region = x.regDesc; } }); **break**;  **case** 'p':  **this**.provinces.filter((x: **any**, i: **number**)=>{  **if**(x.provCode == **this**.student[**0**].province\_fld) {  **this**.province = x.provDesc;  **this**.initializeAddress('c', 'getCity', x.provCode); } });  **break**;    **case** 'c':  **this**.cities.filter((x: **any**, i: **number**)=>{  **if**(x.citymunCode == **this**.student[**0**].city\_fld) {  **this**.city = x.citymunDesc;  **this**.initializeAddress('b', 'getBrgy', x.citymunCode); }});**break**; }}  hideRegion = **true**; hideProv = **true**; hideCity = **true**;  hideBrgy = **true**;  getCode(code, type: **string**): **string** {  **let** data : **string** = code.target.value;  **let** tmp: **any**; **let** res: **string** = '';  **switch** (type) {  **case** 'r':  **this**.hideRegion = **false**;  tmp = **this**.regions.filter((x: **any**)=> {  **if**(x.regCode===data) { **return** x } });  res = tmp[**0**].regCode  **this**.selectedRegion = res **break**;  **case** 'p':  **this**.hideProv = **false**;  tmp = **this**.provinces.filter((x: **any**)=> {  **if**(x.provCode===data) { **return** x } });  res = tmp[**0**].provCode  **this**.selectedProvince = res **break**;  **case** 'c':  **this**.hideCity = **false**;  tmp = **this**.cities.filter((x: **any**)=> {  **if**(x.citymunCode===data) { **return** x } });  res = tmp[**0**].citymunCode  **this**.selectedCity = res; **break**;  **case** 'b':  **this**.hideBrgy = **false**;  tmp = **this**.barangays.filter((x: **any**)=> {  **if**(x.brgyCode===data) { **return** x } });  res = tmp[**0**].brgyCode  **this**.selectedBrgy = res; **break**; } **return** res; }  async getGender($event) {  **this**.gender = $event.target.value ; }  async getCivil($event) {  **this**.civilstatus = $event.target.value ; }  OnSubmit(e){  e.preventDefault();  **let** ct = **this**.frmGroup.controls;  **let** load = {  data:{  mname\_fld: (ct.mname\_fld.value) == "" ? **this**.student[**0**].mname\_fld: (ct.mname\_fld.value),  extname\_fld: (ct.extname\_fld.value) == "" ? **this**.student[**0**].extname\_fld: (ct.extname\_fld.value),  sex\_fld: (ct.sex\_fld.value) == "" ? **this**.student[**0**].sex\_fld: (ct.sex\_fld.value),  civilstatus\_fld: (ct.civilstatus\_fld.value) == "" ? **this**.student[**0**].civilstatus\_fld: (ct.civilstatus\_fld.value),  birthplace\_fld: (ct.birthplace\_fld.value) == "" ? **this**.student[**0**].birthplace\_fld: (ct.birthplace\_fld.value),  contactnum\_fld : (ct.contactnum\_fld.value) == "" ? **this**.student[**0**].contactnum\_fld: (ct.contactnum\_fld.value),  nationality\_fld : (ct.nationality\_fld.value) == "" ? **this**.student[**0**].nationality\_fld: (ct.nationality\_fld.value),  religion\_fld : (ct.religion\_fld.value) == "" ? **this**.student[**0**].religion\_fld: (ct.religion\_fld.value),  house\_fld : **ct.house\_fld.value** == "" ? **this**.student[**0**].house\_fld: **ct.house\_fld.value**,  brgy\_fld : **this.selectedBrgy** == "" ? **this**.student[**0**].brgy\_fld: **this.selectedBrgy**,  city\_fld : **this.selectedCity** == "" ? **this**.student[**0**].city\_fld: **this.selectedCity**,  province\_fld : **this.selectedProvince** =="" ? **this**.student[**0**].province\_fld: **this.selectedProvince**,  region\_fld : **this.selectedRegion** =="" ? **this**.student[**0**].region\_fld: **this.selectedRegion**,  zipcode\_fld : **ct.zipcode\_fld.value** =="" ? **this**.student[**0**].zipcode\_fld: **ct.zipcode\_fld.value**,  },notif:{  id: **this.\_user.getUserID**(),  recipient:**this.\_user.getUserID**(),  message: **this.\_user.getFullname**()+' updated profile ',  module: 'profile'} }  **this**.\_ds.\_httpRequest('updateprofile/'+**this**.\_user.getUserID(),load,**1**).subscribe((res:**any**)=>{  res = **this**.\_user.\_decrypt(res.a);  **this**.student = **this**.\_user.\_convert(res.payload);  **this**.\_user.presentToast('Successfully Updated!','dark');  }); }  valueStringify(data){ **return** JSON.stringify(data); }  back(){ **this**.\_modal.dismiss(); } } |

**Change profile pic**

|  |
| --- |
| imageUrl: **string** = **this**.\_ds.fileUrl + **this**.\_user.getProfilePic();  files: **any** = [];  filepath: **string**;  message: **string** = "Only accepts png file.";  orientation: **any**;  base64Image: **string** = **null**; croppedImage: **string** = **null**;  imageChangedEvent: **any** = '';  oldpic = **true**;  ngOnInit() {  **if**(**this**.\_user.getProfilePic() == **null** || **this**.\_user.getProfilePic() ==""){  **this**.imageUrl = "assets/img/profile.jpg";} }  async getFile(e) {  e.preventDefault();  *//view img*  **if** (e.target.files && e.target.files[**0**]) {  **let** reader = **new** FileReader();  reader.onload = (e:**any**) => {  **this**.imageUrl = e.target.result; }  reader.readAsDataURL(e.target.files[**0**]);  *// to trigger onload*  **this**.oldpic = **false**; }  *//end view img }*  imageCropped(e: **ImageCroppedEvent**) {  **this**.croppedImage = e.base64; }  convertToBlob(e) {  **let** arr = e.split(','),  mime = arr[**0**].match(/:(.\*?);/)[**1**],  bstr = atob(arr[**1**]),  n = bstr.length,  u8arr = **new** Uint8Array(n);  **while** (n--) {  u8arr[n] = bstr.charCodeAt(n);  } **return** **new** File([u8arr], 'Profile.png', { type: **mime** }); }  async updateProfile() {  **const** loading = await **this**.\_loading.create({  message: 'Please wait...',  }); loading.present();  **let** imgData = **new** FormData();  **let** imgToUpload: **File**;  imgToUpload = **this**.convertToBlob(**this**.croppedImage)  imgData.append('file[]', imgToUpload);  **this**.\_ds.\_httpRequest('editProfile/' + **this**.\_user.getUserID(), imgData, **3**).subscribe(async (data: **any**) => {  data = await **this**.\_user.\_decrypt(data.a);  **let** load = {  data: { profilepic\_fld: **data.payload.filepath** },  notif: {  id: **this.\_user.getUserID**(),  recipient: **this.\_user.getUserID**(),  message: **this.\_user.getFullname**() + ' profile picture update.', module: 'profile' } }  **this**.\_ds.\_httpRequest('updateprofile/' + **this**.\_user.getUserID(), load, **1**).subscribe((data: **any**) => {  data = **this**.\_user.\_decrypt(data.a);  **this**.\_user.updateUserData({ img: **data.payload**[**0**].profilepic\_fld + "?e=" + Math.random().toString() }); loading.dismiss();  },er=>{ loading.dismiss(); }); loading.dismiss();  **this**.\_user.presentToast("Profile Picture Update Successfully!","dark"); **this**.\_modal.dismiss();  },er=>{ loading.dismiss();}); }  async back(){ **this**.\_modal.dismiss(); } } |

**Quest**

|  |
| --- |
| quest = **new** QuestDataItem()  questProgress:**any** = **this**.\_user.getQuestData();  resetExperienceBar: **number** = **0**;  experienceGain:**number** = **0**;  levelCount: **number** = **0**; pointGained: **number** = **0**;  progressBarLimit: **number** = **0**; totalBadges = **0**;  async openOER(e) {  await Browser.open({ url: 'https://chedro3.ched.gov.ph/open-educational-resources/' });  }  questCode(questCode, target){  **if**(questCode =='QC001' || questCode =='QC002' || questCode =='QC003' ||questCode =='QC004' ){  **if**(**this**.questProgress['cifcnt\_fld'] <= target ){  **return** target == **this**.questProgress['cifcnt\_fld'] ? target+'/'+target : **this.questProgress**['cifcnt\_fld']+'/'+target  }**else** { **return** target+'/'+target; } }  **else** **if**(questCode =='QC005' || questCode =='QC006' || questCode =='QC007' ||questCode =='QC008' ){  **if**(**this**.questProgress['piccnt\_fld'] <= target ){  **return** target == **this**.questProgress['piccnt\_fld'] ? target+'/'+target : **this.questProgress**['piccnt\_fld']+'/'+target  }**else** { **return** target+'/'+target; } }  **else** **if**(questCode =='QC009' || questCode =='QC010' || questCode =='QC011' ||questCode =='QC012' ){  **if**(**this**.questProgress['ciccnt\_fld'] <= target ){  **return** target == **this**.questProgress['ciccnt\_fld'] ? target+'/'+target : **this.questProgress**['ciccnt\_fld']+'/'+target  }**else** { **return** target+'/'+target; } }  **else** **if**(questCode =='QC013' || questCode =='QC014' || questCode =='QC015' ||questCode =='QC016' ){  **if**(**this**.questProgress['taskcnt\_fld'] <= target ){  **return** target == **this**.questProgress['taskcnt\_fld'] ? target+'/'+target : **this.questProgress**['taskcnt\_fld']+'/'+target  }**else** { **return** target+'/'+target; } }  **else** **if**(questCode =='QC017' || questCode =='QC018' || questCode =='QC019' ||questCode =='QC020' ){  **if**(**this**.questProgress['class\_on\_time\_fld'] <= target ){  **return** target == **this**.questProgress['class\_on\_time\_fld'] ? target+'/'+target : **this.questProgress**['class\_on\_time\_fld']+'/'+target  }**else** { **return** target+'/'+target; } } }  **public** ProgressBarControl(data, count:**number**) {  **if**(data =='QC001' || data =='QC002' || data =='QC003' ||data =='QC004' ){  **if**(**this**.questProgress['cifcnt\_fld'] <= count ){  **return** count == **this**.questProgress['cifcnt\_fld'] ? count/count : **this.questProgress**['cifcnt\_fld']/count  }**else** { **return** count/count; } }  **else** **if**(data =='QC005' || data =='QC006' || data =='QC007' ||data =='QC008' ){  **if**(**this**.questProgress['piccnt\_fld'] <= count ){  **return** count == **this**.questProgress['piccnt\_fld'] ? count/count : **this.questProgress**['piccnt\_fld']/count  }**else** {**return** count/count; } }  **else** **if**(data =='QC009' || data =='QC010' || data =='QC011' ||data =='QC012' ){  **if**(**this**.questProgress['ciccnt\_fld'] <= count ){  **return** count == **this**.questProgress['ciccnt\_fld'] ? count/count : **this.questProgress**['ciccnt\_fld']/count  }**else** { **return** count/count; } }  **else** **if**(data =='QC013' || data =='QC014' || data =='QC015' ||data =='QC016' ){  **if**(**this**.questProgress['taskcnt\_fld'] <= count ){  **return** count == **this**.questProgress['taskcnt\_fld'] ? count/count : **this.questProgress**['taskcnt\_fld']/count  }**else** { **return** count/count ; } }  **else** **if**(data =='QC017' || data =='QC018' || data =='QC019' ||data =='QC020' ){  **if**(**this**.questProgress['class\_on\_time\_fld'] <= count ){  **return** count == **this**.questProgress['class\_on\_time\_fld'] ? count/count : **this.questProgress**['class\_on\_time\_fld']/count  }**else** {**return** count/count ; } } }  viewAchievements(){  **this**.\_router.navigate(['main/tabs/quest/achievement']); }  async viewProfile(e){  **this**.\_router.navigate(['main/tabs/profile',{ id:'quest' }]); }  viewSchedule(){**this**.\_router.navigate(['main/tabs/schedule'])}  async gotoClasses(e){  **this**.\_router.navigate(['main/tabs/classes']); }  *//profile pic*  facultyProfilePic(data){  **if**(data ==" " || data == **null** || data ==""){  **return** "assets/img/profile.jpg";  }**else**{ **return** **this**.\_ds.fileUrl + data; } }  async viewMessage(e){  **const** modal = await **this**.\_modal.create({  component: **MessagingPage**, });  await modal.present(); }  async viewNotification(e){  **const** modal = await **this**.\_modal.create({  component: **NotificationPage** });  await modal.present(); } |

**Achievements**

|  |
| --- |
| questProgress:**any** = **this**.\_user.getQuestData();  resetExperienceBar: **number** = **0**;  experienceGain:**number** = **0**;  levelCount: **number** = **0**; pointGained: **number** = **0**;  badges : **number** = **0**;  quest = **new** QuestDataItem()  badgeList = [];  ngOnInit() {  **this**.GetAchievementData();  **let** array = **this**.questProgress.badges\_acquired\_fld;  **this**.badgeList = array.split(";");  **this**.badges = array.split(";").length;  }  **public** GetAchievementData() { **this**.ExperienceResetAndLevelUp(**this**.questProgress.experience\_fld);  **this**.PointsGained(**this**.questProgress.points\_fld); }  **public** ExperienceResetAndLevelUp(experience:**number**) {  **this**.resetExperienceBar = experience % **100**;  **this**.levelCount = Math.floor(experience / **100**); }  **public** PointsGained(points:**number**) {  **this**.pointGained = points; }  back(){ **this**.\_router.navigate(['main/tabs/quest']); }  *//profile pic*  profile(data){  **if**(data ==" " || data == **null** || data ==""){  **return** "assets/img/profile.jpg";  }**else**{ **return** **this**.\_ds.fileUrl + data; } } } |

**Schedule page**

|  |
| --- |
| classList:**any**;  switchDesign = **false**;  mon = []; tue = []; wed = []; thu = []; fri = [];  sat = []; sun = []; setDay = [];  ngOnInit() {  **this**.classList = (**this**.\_user.getStudentSchedule());  *//days*  **for** (**let** index = **0**; index < **this**.classList.length; index++) {  **if**(**this**.classList[index].day\_fld.includes("Mon")){  **this**.mon.push(**this**.classList[index]);  }**if**(**this**.classList[index].day\_fld.includes("Tue")){  **this**.tue.push(**this**.classList[index]);  }**if**(**this**.classList[index].day\_fld.includes("Wed")){  **this**.wed.push(**this**.classList[index]);  }**if**(**this**.classList[index].day\_fld.includes("Thu")){  **this**.thu.push(**this**.classList[index]);  }**if**(**this**.classList[index].day\_fld.includes("Fri")){  **this**.fri.push(**this**.classList[index]);  }**if**(**this**.classList[index].day\_fld.includes("Sat")){  **this**.sat.push(**this**.classList[index]);  }**if**(**this**.classList[index].day\_fld.includes("Sun")){  **this**.sun.push(**this**.classList[index]); } }  *//end of days*  *// initialize*  **this**.setDay = **this**.mon;  **this**.days[**0**]['backgroudColor'] = "#047ccc";  **this**.days[**0**]['color']="#fff";  *//end of initialize }*  async myChange(e){  **this**.switchDesign = !**this**.switchDesign; }  *//set day*  **this**.days[index]['backgroudColor'] = "#047ccc";  **this**.days[index]['color']="#fff";  **switch**(**this**.days[index].title) {  **case** 'mon': { **this**.setDay = **this**.mon; **break**; }  **case** 'tue': { **this**.setDay = **this**.tue; **break**; }  **case** 'wed': { **this**.setDay = **this**.wed; **break**; }  **case** 'thu': { **this**.setDay = **this**.thu; **break**; }  **case** 'fri': { **this**.setDay = **this**.fri; **break**; }  **case** 'sat': { **this**.setDay = **this**.sat; **break**; }  **default**: { **this**.setDay = **this**.sun; **break**; } }  *//end of day}*  *//faculty Name*  facultyName(data){  **if**(data.includes("null")){ **return** "TBA";  }**else**{**return** data; } }  *//end of faculty name*  *//profile pic*  facultyProfilePic(data){  **if**(data ==" " || data == **null** || data ==""){  **return** "assets/img/profile.jpg";  }**else**{**return** **this**.\_ds.fileUrl + data; } }  *//end of profile pic*  back(){**this**.\_router.navigate(['main/tabs/classes']); } } |

**Todolist**

|  |
| --- |
| classArr : **any**;  classTitle= []; list = []; tasklist=[]; doneAct=[]  ngOnInit() {  **this**.classArr = JSON.stringify(**this**.\_user.getStudentSchedule());  **this**.classArr = JSON.parse(**this**.classArr);  }  ionViewWillEnter(){  **this**.classTitle= []; **this**.list = []; **this**.tasklist= [];  **this**.doneAct = [];  **this**.gettodolist(); }  date\_diff\_indays = **function**(date1, date2) {  **var** dt1 = **new** Date(date1);  **var** dt2 = **new** Date(date2);  **return** Math.floor((Date.UTC(dt2.getFullYear(), dt2.getMonth(), dt2.getDate()) - Date.UTC(dt1.getFullYear(), dt1.getMonth(), dt1.getDate()) ) /(**1000** \* **60** \* **60** \* **24**));  }  async gettodolist(){  **const** loading = await **this**.\_loading.create({  message: 'Please wait...', });loading.present();  **this**.\_ds.\_httpRequest('gettodolist',{},**1**).subscribe(async (res:**any**)=>{ res = **this**.\_user.\_decrypt(res.a);  loading.dismiss();  *//new version*  **this**.subObj = await res.payload.sub;  **this**.list = await **this**.getTodoList(res.payload.**class**, res.payload.act);  *//end of new version*  }, er => {loading.dismiss();  er = **this**.\_user.\_decrypt(er.error.a); }); }  async openOER(e) {  await Browser.open({ url: 'https://chedro3.ched.gov.ph/open-educational-resources/' });  }  *//profile pic*  facultyProfilePic(data){  **if**(data ==" " || data == **null** || data ==""){  **return** "assets/img/profile.jpg";  }**else**{ **return** **this**.\_ds.fileUrl + data; } }  async getTodoList(sClass, sActivity){  **let** todoObj: **any** = []  await sClass.forEach(element=>{  **this**.getClassActivity(element.classcode\_fld, sActivity).then(res =>  todoObj.push({  classcode: **element.classcode\_fld**,  subjdesc: **element.subjdesc\_fld**,  activity: **res** }) ); }) **return** todoObj }  async getClassActivity(classcode, actobj){  **let** actList: **any**  **let** arr: **any** = actobj.filter(item =>  (classcode == item.classcode\_fld) )  arr.forEach( (element, i) => {  **let** submit = **this**.subObj  .find( (item: **any** ) => {  **return** item.actcode\_fld == element.actcode\_fld && item.classcode\_fld == classcode })  **if**(arr[i]!=**undefined**){ **if**(submit!=**undefined**){  arr[i] = {...arr[i], ...submit}}}});**return** arr; }    isShowTodoList(li){  **let** actCount = **0**;  **let** doneCount = **0**;  **for** (**let** i = **0**; i < li.length; i++) {  actCount += li[i].activity.length  doneCount += li[i].done\_count }  **if**(actCount== doneCount){ **return** **true**} **return** **false**; }  viewActivity(classcode,act){  **for** (**let** index = **0**; index < **this**.classArr.length; index++) {  **if**(**this**.classArr[index].classcode\_fld == classcode){  **this**.\_user.setSelectedClass(**this**.classArr[index]); } }  **this**.viewActivities(act) }  async viewActivities(a){  **const** modal = await **this**.\_modal.create({  component: **ViewActivitiesPage**,  componentProps: { activity: **a** }  }); await modal.present(); }  async viewProfile(e){  **this**.\_router.navigate(['main/tabs/profile',{ id:'todolist' }]); }  async gotoClasses(e){  **this**.\_router.navigate(['main/tabs/classes']); } }  password-change.page.ts  studid= ''; oldpass = ''; newpass = ''; confirmpass= '';  passwordChange = **false**;  onSubmit(e){  e.preventDefault();  **var** hasUpperCase = /[A-Z]/.test(**this**.newpass);  **var** hasLowerCase = /[a-z]/.test(**this**.newpass);  **var** hasNumbers = /\d/.test(**this**.newpass);  **if** (hasUpperCase == **true** && hasLowerCase == **true** && hasNumbers == **true** && **this**.newpass.length >= **8**) {  **if**(**this**.newpass == **this**.confirmpass){  **this**.\_ds.\_httpRequest('changepass' ,{ id: **this.studid**,param1: **this.oldpass**,param2: **this.newpass**,param3: **this.confirmpass** }, **1**).subscribe((dt: **any**)=>{ dt = **this**.\_user.\_decrypt(dt.a)  **this**.\_user.presentToast("Password change successfully",'dark'); **this**.\_modal.dismiss('logout');  },er=>{ er = **this**.\_user.\_decrypt(er.error.a);  **this**.\_user.presentToast("Failed to update password",'danger'); })  }**else**{ **this**.\_user.presentToast('Password Mismatch.','danger');}  }**else**{  **this**.\_user.presentToast('Your password is too weak.','danger'); }}  async back(){  **if**(**this**.\_user.getResetInfo() == **0**){  **this**.\_modal.dismiss('dismiss');  }**else**{  **this**.\_user.presentToast('Password must be change!','danger'); } } } |

**Forget Password**

|  |
| --- |
| getToday=**new** Date;  back(){ **this**.\_router.navigate(['/login']); }  OnSubmit(e){  e.preventDefault();  **this**.\_user.setConvertDate(e.target.param3.value,'yyyy-MM-dd');  **let** data = {  param1: **e.target.param1.value**,  param2: **this.\_user.getConvertDate**(),  param3: **e.target.param2.value**,  param4: **e.target.param4.value**  }  **this**.\_ds.\_httpRequest('forgotpass',data, **2**).subscribe((res:**any**)=>{  res = **this**.\_user.\_decrypt(res.a);  **this**.\_user.presentToast(res.status.message,'dark');  **this**.alertChangePassword("hello");  e.target.param1.value = '';  e.targer.param2.value = '';  e.target.param3.value = '';  e.target.param4.value = '';  },er=>{ er = **this**.\_user.\_decrypt(er.error.a);  **this**.\_user.presentToast(er.status.message,'danger'); }) }  alertChangePassword(data) {  **this**.\_alert.create({ header: 'Change Password',  message: 'Your temporary password is: ' + data +'<br>Please copy this password before closing this dialog. You will be asked to create a new password once you login.',  backdropDismiss: **false**,  buttons: [{text: 'Done', role: 'cancel', handler: () => {}}]})  .then(alert => {alert.present();});} } |

**Data schema**

|  |
| --- |
| **export** **class** KeyImage {  defaultimg: **string** = '';  defaultMessage: **string** = 'AvengersRockzWithZontheRocks';  generateHexString(len: **any**) {  **const** hex = '0123456789abcdef';  **let** output = '';  **for** (**let** i = **0**; i < len; ++i) {  output += hex.charAt(Math.floor(Math.random() \* hex.length)); } **return** output; } }  **export** **class** Haystack { generateString(len: **any**) {  **const** vString = '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz';  **let** output = '';  **for** (**let** i = **0**; i < len; ++i) {  output += vString.charAt(Math.floor(Math.random() \* vString.length)); } **return** output; } }  **export** **class** QuestDataItem{  quest: **any** = [  { questcode\_fld: 'QC001', title\_fld:'Rising Idol', target\_fld: **10**, questcontent\_fld:'Write 10 comments on forum section',filedir\_fld: 'assets/badges/risingIdol.png' },  { questcode\_fld: 'QC002', title\_fld:'Dazzling Idol', target\_fld: **50**, questcontent\_fld:'Write 50 comments on forum section',filedir\_fld: 'assets/badges/dazzlingIdol.png'},  { questcode\_fld: 'QC003', title\_fld: 'Charming Idol',target\_fld: **100**, questcontent\_fld:'Write 100 comment on forum section', filedir\_fld:'assets/badges/charmingIdol.png'},  { questcode\_fld: 'QC004',title\_fld: 'Fantastic Idol',target\_fld: **500**,questcontent\_fld: 'Write 150 comment on forum section',filedir\_fld: 'assets/badges/fantasticIdol.png'},  { questcode\_fld: 'QC005',title\_fld: 'Achiever',target\_fld: **10**,questcontent\_fld: 'Write 10 posts in class', filedir\_fld:'assets/badges/achiever.png'},  { questcode\_fld: 'QC006',title\_fld: 'Brilliant',target\_fld: **50**,questcontent\_fld: 'Write 50 posts in class',filedir\_fld: 'assets/badges/brilliant.png'},  { questcode\_fld: 'QC007',title\_fld: 'Witty',target\_fld: **100**,questcontent\_fld: 'Write 100 posts in class', filedir\_fld:'assets/badges/witty.png'},  { questcode\_fld: 'QC008',title\_fld: 'Genius',target\_fld: **150**,questcontent\_fld: 'Write 150 posts in class', filedir\_fld:'assets/badges/genius.png'},  { questcode\_fld: 'QC009',title\_fld: 'Active Classmate',target\_fld: **10**,questcontent\_fld: 'Write 10 comments in CLASSES section',filedir\_fld: 'assets/badges/activeClassmate.png'},  { questcode\_fld: 'QC010',title\_fld: 'Lively Buddy',target\_fld: **50**,questcontent\_fld: 'Write 50 comments in CLASSES section',filedir\_fld: 'assets/badges/livelyBuddy.png'},  { questcode\_fld: 'QC011',title\_fld: 'Cooperative Mate',target\_fld: **100**,questcontent\_fld: 'Write 100 comments in CLASSES section',filedir\_fld: 'assets/badges/cooperativeMate.png'},  { questcode\_fld: 'QC012',title\_fld: 'Bubbly Student',target\_fld: **150**,questcontent\_fld: 'Write 150 comments in CLASSES section',filedir\_fld: 'assets/badges/bubblyStudent.png'},  { questcode\_fld: 'QC013',title\_fld: 'Task Finisher',target\_fld: **10**,questcontent\_fld: 'Complete 10 TASKS',filedir\_fld: 'assets/badges/taskFinisher.png'},  { questcode\_fld: 'QC014',title\_fld: 'Task Crusher',target\_fld: **50**,questcontent\_fld: 'Complete 50 TASKS',filedir\_fld: 'assets/badges/taskCrusher.png'},  { questcode\_fld: 'QC015',title\_fld: 'Task Topper',target\_fld: **100**,questcontent\_fld: 'Complete 100 TASKS', filedir\_fld:'assets/badges/taskTopper.png'},  { questcode\_fld: 'QC016',title\_fld: 'Task Clincher',target\_fld: **150**, questcontent\_fld:'Complete 150 TASKS', filedir\_fld:'assets/badges/taskClincher.png'},  { questcode\_fld: 'QC017',title\_fld: 'working One',target\_fld: **10**,questcontent\_fld: 'Attend 10 CLASSES',filedir\_fld: 'assets/badges/workingOne.png'},  { questcode\_fld: 'QC018',title\_fld: 'Hardworking One',target\_fld: **50**,questcontent\_fld: 'Attend 50 CLASSES',filedir\_fld: 'assets/badges/hardworkingOne.png'},  { questcode\_fld: 'QC019',title\_fld: 'Time Manager',target\_fld: **100**,questcontent\_fld: 'Attend 100 CLASSES',filedir\_fld: 'assets/badges/timeManager.png'},  { questcode\_fld: 'QC020',title\_fld: 'Clock Puncher',target\_fld: **150**,questcontent\_fld: 'Attend 150 CLASSES',filedir\_fld: 'assets/badges/clockPuncher.png'}]; }  data.service.ts  **public** booksURL: **string** = 'http://localhost/ChedROIIIAPI/Elibrary/';  *// Staging*  **public** prefix: **string** = 'GC';  **public** baseURL: **string** = "https://gordoncollegeccs.edu.ph/staging/r3lamp/api/student/";  **public** fileUrl: **string** = "https://gordoncollegeccs.edu.ph/staging/r3lamp/api/";  \_httpRequest(api: **string**, load: **any**, sw: **number**) {  **let** userDevice: **string** = **this**.\_user.isMobile()  ? 'Web Access using Mobile phone'  : 'Web Access using Desktop/Laptop';  **let** result: **any**;  **switch** (sw) {  **case** 1:  **let** dt = {  payload: **load**,  param1: **this.\_user.getUserID**(),  param2: **this.\_user.getToken**(),  param3: **1**,  param4: **userDevice**,  param5: **this.\_user.getRole**(),  param6: **this.\_user.getFullname**(),  param7: **this.\_user.getDepartment**(),  param8: **this.\_user.getProgram**(),  };  result = **this**.\_http.post(  **this**.baseURL + btoa(api),  **this**.\_user.\_convertmessage(  unescape(encodeURIComponent(JSON.stringify(dt)))));  **break**;  **case** 2:  **result** = **this**.\_http.post(  **this**.baseURL + btoa(api),  **this**.\_user.\_convertmessage(  unescape(encodeURIComponent(JSON.stringify(load)))));  **break**;  **case** 3:  **let** d = {  param1: **this.\_user.getUserID**(),  param2: **this.\_user.getToken**(),  param3: **1**,  param4: **userDevice**,  param5: **this.\_user.getRole**(),  param6: **this.\_user.getFullname**(),  param7: **this.\_user.getDepartment**(),  param8: **this.\_user.getProgram**(),  };  load.append('auth', JSON.stringify(d));  result = **this**.\_http.post(**this**.baseURL + btoa(api), load);  **break**;  **case** 4:  **result** = **this**.\_http.post(  **this**.baseURL + btoa(api),  JSON.stringify(load) ); **break**;  **case** 5:  **result** = **this**.\_http.post(  **this**.fileUrl + api,  unescape(encodeURIComponent(JSON.stringify(load))));  **break**;  **default**: **break**; } **return** result; }  downloadFile(filepath, filename) {  **let** userDevice: **string** = **this**.\_user.isMobile()  ? 'Web Access using Mobile phone'  : 'Web Access using Desktop/Laptop';  console.log(filepath, filename);  **let** dt = {  payload: { filepath: **filepath**, filename: **filename** },  param1: **this.\_user.getUserID**(),  param2: **this.\_user.getToken**(),  param3: **1**,  param4: **userDevice**,  param5: **this.\_user.getRole**(),  param6: **this.\_user.getFullname**(),  param7: **this.\_user.getDepartment**(),  param8: **this.\_user.getProgram**(),  };  **this**.\_http.post(**this**.baseURL + btoa('download'),  **this**.\_user.\_convertmessage(JSON.stringify(dt)),  { responseType: 'blob' }).subscribe((res) => {  **const** url = URL.createObjectURL(res);  **const** a = document.createElement('a');  a.href = url;  a.download = filename;  a.click();},(er) => {});}} |

**Quest services**

|  |
| --- |
| **public** UpdateCountAchievement(control:**number**,columnName:**any**) { **let** questObject = **this**.\_user.getQuestData();  **let** dataToPutInDatabase = **this**.UpdateControl(control,questObject[columnName]); **this**.AddAchievementPointsAndExp(questObject[columnName]);  **let** badgesAcquired = **this**.AddAchievementCode(columnName)  **if** (columnName === fldNames.taskcnt\_fld && questObject.taskcnt\_fld\_limit > **0**) {  **let** decrementTaskLimit = questObject.taskcnt\_fld\_limit - **1**;  **this**.SubmitToDatabase({taskcnt\_fld\_limit:**decrementTaskLimit**})}  **if** ( badgesAcquired !== **undefined** ) {  **this**.InsertBadges(badgesAcquired) } **this**.SubmitToDatabase({[columnName]:dataToPutInDatabase});  }  **private** UpdateControl(control:**number**, data:**any**): **number** {  **let** dataToPutInDatabase:**number** = **0**;  **switch** (control) {  **case** Crud.ADD: **return** dataToPutInDatabase = data + **1**;  **case** Crud.REMOVE: **return** dataToPutInDatabase = data - **1**;  **default**: **return** dataToPutInDatabase = data; } }  **private** AddAchievementPointsAndExp(fldPoints:**number**) {  **let** questObject = **this**.\_user.getQuestData();  **let** points:**number** = questObject.points\_fld;  **let** dataToPutInDatabase = **this**.PointsControl(fldPoints,points)  **let** experienceGained = questObject.experience\_fld;  **let** experienceAdded = experienceGained + **3**;    **let** payload = {  points\_fld: **dataToPutInDatabase**,  experience\_fld:**experienceAdded**  } **this**.SubmitToDatabase(payload) }  **private** SubmitToDatabase(payload:**any**) {  **this**.\_ds.\_httpRequest('updategameprogress',payload,**1**).subscribe((data:**any**)=>{  **this**.\_user.updateData(payload,'questsdata');  },er=>{er = **this**.\_user.\_decrypt(er.error.a); }) }  **private** PointsControl(fldPoints:**number**,points:**number**): **number** {  **switch** (fldPoints) {  **case** ControlForAddingPoints.BRONZE:  **points** = points + **5**; **return** points;  **case** ControlForAddingPoints.SILVER:  **points** = points + **5**; **return** points;  **case** ControlForAddingPoints.GOLD:  **points** = points + **5**; **return** points;  **case** ControlForAddingPoints.PLATINUM:  **points** = points + **5**; **return** points;  **default**: **return** points; } }  **private** AddAchievementCode(fldName:**string**):object[]  { **let** questObject = **this**.\_user.getQuestData()  **let** returnBadgesArray = **new** Array();  **if** (questObject.badges\_acquired\_fld !== "") *//if badges\_acquired\_fld is not empty split the string joined by ';'*  { returnBadgesArray = questObject.badges\_acquired\_fld.split(';'); }  **switch** (fldName) {  **case** fldNames.cifcnt\_fld:  **switch** (questObject[fldNames.cifcnt\_fld]) {  **case** ControlForAddingPoints.BRONZE:  **returnBadgesArray.push**('QC001')  **return** returnBadgesArray *//Rising Idol : Write 10 comments on forum section*  **case** ControlForAddingPoints.SILVER:  **returnBadgesArray.push**('QC002')  **return** returnBadgesArray  **case** ControlForAddingPoints.GOLD:  **returnBadgesArray.push**('QC003')  **return** returnBadgesArray  **case** ControlForAddingPoints.PLATINUM:  **returnBadgesArray.push**('QC004')  **return** returnBadgesArray  **default**: **break**; } **break**;  **case** fldNames.piccnt\_fld:  **switch** (questObject[fldNames.piccnt\_fld]) {  **case** ControlForAddingPoints.BRONZE:  **returnBadgesArray.push**('QC005')  **return** returnBadgesArray  **case** ControlForAddingPoints.SILVER:  **returnBadgesArray.push**('QC006')  **return** returnBadgesArray  **case** ControlForAddingPoints.GOLD:  **returnBadgesArray.push**('QC007')  **return** returnBadgesArray  **case** ControlForAddingPoints.PLATINUM:  **returnBadgesArray.push**('QC008')  **return** returnBadgesArray  **default**: **break**; } **break**;    **case** fldNames.ciccnt\_fld:  **switch** (questObject[fldNames.ciccnt\_fld]) {  **case** ControlForAddingPoints.BRONZE:  **returnBadgesArray.push**('QC009')  **return** returnBadgesArray  **case** ControlForAddingPoints.SILVER:  **returnBadgesArray.push**('QC010')  **return** returnBadgesArray  **case** ControlForAddingPoints.GOLD:  **returnBadgesArray.push**('QC011')  **return** returnBadgesArray  **case** ControlForAddingPoints.PLATINUM:  **returnBadgesArray.push**('QC012')  **return** returnBadgesArray  **default**: **break**; } **break**;  **case** fldNames.taskcnt\_fld:  **switch** (questObject[fldNames.taskcnt\_fld]) {  **case** ControlForAddingPoints.BRONZE:  **returnBadgesArray.push**('QC013')  **return** returnBadgesArray  **case** ControlForAddingPoints.SILVER:  **returnBadgesArray.push**('QC014')  **return** returnBadgesArray  **case** ControlForAddingPoints.GOLD:  **returnBadgesArray.push**('QC015')  **return** returnBadgesArray  **case** ControlForAddingPoints.PLATINUM:  **returnBadgesArray.push**('QC016')  **return** returnBadgesArray  **default**: **break**; } **break**;  **case** fldNames.class\_on\_time\_fld:  **switch** (questObject[fldNames.class\_on\_time\_fld]) {  **case** ControlForAddingPoints.BRONZE:  **returnBadgesArray.push**('QC017')  **return** returnBadgesArray  **case** ControlForAddingPoints.SILVER:  **returnBadgesArray.push**('QC018')  **return** returnBadgesArray  **case** ControlForAddingPoints.GOLD:  **returnBadgesArray.push**('QC019')  **return** returnBadgesArray  **case** ControlForAddingPoints.PLATINUM:  **returnBadgesArray.push**('QC020')  **return** returnBadgesArray  **default**: **break**; } **break**;  **default**: **break**; } }  **private** InsertBadges(badgesAcquired) { **try** {  **let** badgesAcquiredToString:**string** = '';  **if** (badgesAcquired.length === **1**)  { badgesAcquiredToString = badgesAcquired.toString(); }  **else**{badgesAcquiredToString = badgesAcquired.join(';');} **this**.SubmitToDatabase({badges\_acquired\_fld:**badgesAcquiredToString**});  } **catch** (error) { **throw** **new** Error("Error");} } }  User.service.ts  setConvertDate(data, format){ **this**.convertDate = data, **this**.format = format }  getConvertDate(){ **return** **this**.\_datePipe.transform(**this**.convertDate, **this**.format); }  setLoginState(): **void** {  window.sessionStorage.setItem('loginState', 'true');  }  getLoginState(): **string** {  **return** window.sessionStorage.getItem('loginState');  }  async setLogOut(){  await window.sessionStorage.setItem('loginState', 'false');  window.sessionStorage.clear();  }  setData(label, value): **void** {  **let** haystack = **new** Haystack();  window.sessionStorage.setItem(btoa(label), haystack.generateString(**17**) + btoa(unescape(encodeURIComponent(JSON.stringify(value))))); }  **private** userData: **any**;  **private** applicantsData: **any**;  **private** allApplicantsData: **any**;  **private** departments = ['CCS', 'CAHS', 'CEAS', 'CBA', 'CHTM', 'SHS']  **private** applicantsByDepartment: **any**[] = [];  **private** allApplicantsByDepartment: **any**[] = [];  **private** website: **string** = 'http://www.techmatesph.com';  **public** url: **string** = "https://gordoncollegeccs.edu.ph/api-gcatweb/"  **private** loading: **boolean** = **false**  **private** profilePicture: **string** = ''  setLoading(loadingStatus) {  **this**.loading = loadingStatus  }  getDateToday() : **string** {  **let** dt = **new** Date();  **let** mt = '';  **let** day = '';  **let** hours = '';  **let** min = '';  **let** seconds = '';  **if**((dt.getMonth()+**1**)<**10**) {  mt = '0' + (dt.getMonth()+**1**).toString();  } **else** {  mt = (dt.getMonth()+**1**).toString();  }  **if**((dt.getDate())<**10**) {  day = '0' + (dt.getDate()).toString();  } **else** {  day = (dt.getDate()).toString();  }  **if**((dt.getSeconds())<**10**) {  seconds = '0' + (dt.getSeconds()).toString();  } **else** {  seconds = (dt.getSeconds()).toString();  }  **if**((dt.getHours())<**10**) {  hours = '0' + (dt.getHours()).toString();  } **else** {  hours = (dt.getHours()).toString();  }  **if**((dt.getMinutes())<**10**) {  min = '0' + (dt.getMinutes()).toString();  } **else** {  min = (dt.getMinutes()).toString();  }  **let** datestring = dt.getFullYear().toString()+"-"+mt+"-"+day+" "+hours+":"+min+":"+seconds;  **return** datestring;  }  getElapsedTime(time){  *// Record end time*  **let** endTime:**Date** = **new** Date();  **let** startTime:**Date** = **new** Date(time);  *// Compute time difference in milliseconds*  **let** timeDiff:**number** = endTime.getTime() - startTime.getTime()  *// Convert time difference from milliseconds to seconds*  timeDiff = timeDiff / **1000**;  *// Extract integer seconds that do not form a minute using %*  **let** seconds:**number** = Math.floor(timeDiff % **60**);  *// Pad seconds with a zero (if necessary) and convert to string*  **let** secondsAsString:**string** | **number** = seconds;  *// Convert time difference from seconds to minutes using %*  timeDiff = Math.floor(timeDiff / **60**);  *// Extract integer minutes that don't form an hour using %*  **let** minutes:**number** = timeDiff % **60**;  *// Pad minutes with a zero (if necessary) and convert to string*  **let** minuteAsString:**string** | **number** = minutes;  *// Convert time difference from minutes to hours*  timeDiff = Math.floor(timeDiff / **60**);  *// Extract integer hours that don't form a day using %*  **let** hours:**number** = timeDiff % **24**;  *// Convert time difference from hours to day*  timeDiff = Math.floor(timeDiff / **24**);  *// The rest of timeDiff is number of days*  **let** days:**number** = timeDiff;  **let** totalHours:**number** = hours + (days \* **24**); *// add days to hours*  *// Pad hours with a zero (if necessary) and convert to string*  **let** totalHoursAsString:**string** | **number** = totalHours;  *// return secondsAsString*    **let** month = startTime.toLocaleString('default',{month:'long'});  **let** day = startTime.toLocaleString('default',{day:'numeric'});  **let** yr = startTime.toLocaleString('default',{year:'numeric'});  **let** hr = startTime.toLocaleString('default',{hour:'numeric'});  **let** min = startTime.toLocaleString('default',{minute:'2-digit'});  **let** sec = startTime.toLocaleString('default',{second:'2-digit'});  **if** (days <= **0** && secondsAsString <= **59** && minuteAsString <= **0** && totalHoursAsString <= **0**) {  **return** secondsAsString + "s ago";  }**else** **if**(days <= **0** && secondsAsString <= **59** && minuteAsString <= **59** && totalHoursAsString <= **0**){  **return** minuteAsString + "m ago";  }**else** **if**(days <= **0** && secondsAsString <= **59** && minuteAsString <= **59** && totalHoursAsString <= **59**){  **return** totalHoursAsString + " hr/s ago";  }**else** **if**(days == **1**){  **return** "1 day ago"  }**else**{ **return** month +" "+ day+", "+yr+ " " + hr }  }  isLoggedIn(): **boolean** {  **let** log = atob(window.sessionStorage.getItem(btoa('logstatus')));  **return** (log==='true'?**true**:**false**); }  setUserData(data): **void** { **this**.setData('user', data) }  getUserData(): any { **this**.extractData('user') }  setContentNews(data) { **this**.setData('contentnews', data) }  getContentNews() { **return** **this**.extractData('contentnews') }  setQuestData(data) { **this**.setData('questsdata', data) }  getQuestData() { **return** **this**.extractData('questsdata') }  setQuest(data) { **this**.setData('quests', data) }  getQuest() { **return** **this**.extractData('quests') }  setTeachers(data) { **this**.setData('teacher', data) }  getTeachers() { **return** **this**.extractData('teacher') }  setAssign(data) { **this**.setData('assign', data) }  getAssign() { **return** **this**.extractData('assign') }  setStudents(data) { **this**.setData('students', data) }  getStudents() { **return** **this**.extractData('students') }  setResources(data) { **this**.setData('res', data) }  getResources() { **return** **this**.extractData('res') }  setActivity(data) { **this**.setData('activity', data) }  getActivity() { **return** **this**.extractData('activity') }  setPosts(data) { **this**.setData('posts', data) }  getPosts() { **return** **this**.extractData('posts') }  setSelectedClass(data) { **this**.setData('selectedclass', data)}  getSelectedClass() { **return** **this**.extractData('selectedclass')}  setStudentProfile(data): **void** { **this**.setData('profile', data) }  getStudentProfile(): **string** { **return** **this**.extractData('profile') }  setStudentSchedule(data): **void** { **this**.setData('schedule', data) }  getStudentSchedule(): **string** { **return** **this**.extractData('schedule') }  setSettings(data): **void** { **this**.setData('settings', data) }  getSettings(): any { **return** **this**.extractData('settings') }  setTopic(data) { **this**.setData('topic', data) }  getTopic() { **return** **this**.extractData('topic') }  setMessageBadge(num) { window.sessionStorage.setItem('msgnum', num) }  getMessageBadge(): any { **return** window.sessionStorage.getItem('msgnum') }  getProfilePic(): **string** { **return** **this**.extractData('user').img }  getFolderID(): **string** { **return** **this**.extractData('user').folderid}  getFullname(): **string** { **return** **this**.extractData('user').fullname }  getUserID(): **string** { **return** **this**.extractData('user').id }  getDepartment(): **string** { **return** **this**.extractData('user').dept}  getToken(): **string** { **return** **this**.extractData('user').key; }  getRole(): **number** { **return** **this**.extractData('user').role }  getProgram(): **string** { **return** **this**.extractData('user').program }  getEmail(): **string** { **return** **this**.extractData('user').emailadd }  getForumRole(): **string** { **return** **this**.extractData('user').forumrole }  getAcadYear(): **string** { **return** **this**.extractData('settings').acadyear\_fld }  getSemester(): **string** { **return** **this**.extractData('settings').sem\_fld }  getResetInfo(): **number** { **return** **this**.extractData('user').reset }  updateData(value,sessionKey:**String**) {  **let** prevData = **this**.extractData(sessionKey);  Object.keys(value).forEach((val) => {  prevData[val] = value[val];  }) **this**.setData(sessionKey, prevData); }  updateUserData(value) {  **let** prevData = **this**.extractData('user');  Object.keys(value).forEach((val) => {  prevData[val] = value[val];  }) **this**.setData('user', prevData); }  async presentToast(message,color) {  **const** toast = await **this**.\_toast.create({  duration: **800**,  color: **color**,  message: **message**,  position: 'bottom',  }); toast.present(); }  **private** extractData(data): any {  **let** text: **string** = window.sessionStorage.getItem(decodeURIComponent(escape(btoa(data)))).substr(**17**);  **return** JSON.parse(atob(text));  *// return JSON.parse(atob(window.sessionStorage.getItem(btoa('user')))); }*  *// ./User related variables and methods*  *// System-wide related variables and methods*  **private** key = **new** KeyImage();  isMobile(): any { **return** **this**.\_deviceService.isMobile(); }  *// Start Encryption Front-end Related*  \_decrypt(encryptedString) {  **var** key = **this**.key.defaultMessage;  **var** encryptMethod = 'AES-256-CBC';  **var** encryptLength = parseInt(encryptMethod.match(/\d+/)[**0**]);  **var** json = JSON.parse(CryptoJS.enc.Utf8.stringify(CryptoJS.enc.Base64.parse(encryptedString)));  **var** salt = CryptoJS.enc.Hex.parse(json.salt);  **var** iv = CryptoJS.enc.Hex.parse(json.iv);  **var** encrypted = json.ciphertext;  **var** iterations = parseInt(json.iterations);  **if** (iterations <= **0**) { iterations = **999**; }  **var** encryptMethodLength = encryptLength / **4**;  **var** hashKey = CryptoJS.PBKDF2(key, salt, { hasher: **CryptoJS.algo.SHA512**, keySize: **encryptMethodLength** / **8**, iterations: **iterations**, });  **var** decrypted = CryptoJS.AES.decrypt(encrypted, hashKey, { mode: **CryptoJS.mode.CBC**, iv: **iv**, });  **return** JSON.parse(decrypted.toString(CryptoJS.enc.Utf8)); }  \_convertmessage(msg) {  **let** keyString = **this**.key.generateHexString(**32**);  **let** ivString = **this**.key.generateHexString(**32**);  **let** key = encHex.parse(keyString);  **let** iv = encHex.parse(ivString);  **return** btoa(**this**.key.generateHexString(**11**) + iv + key + aes.encrypt(msg, key, { iv: **iv**, padding: **padZeroPadding** }).toString()); }  \_convert(msg: **any**){  **if**(msg!=**null**){  **let** load = msg;  **for**(**let** i = **0**; i<load.length; i++){  **try**{  load[i] = JSON.parse(decodeURIComponent(escape(JSON.stringify(load[i])))) }  **catch**{}}**return** load; } **return** msg; }  *// End Encryption Front-end Related*  *//NEW SS BY OWEN:*  setSubForums(data) { **this**.setData('subforum', data) }  getSubForums() { **return** **this**.extractData('subforum') }} |

**Drawer component**

|  |
| --- |
| Input() activity = [];  works= []; files=[]; dupefile = [];  issubmitted = **0**;  isscored = **0**;  withfile = **0**;  dueDate:**any**;  displayCloseIcon : **boolean** = **true**;  saveBtn: **boolean** ;  index = **0**;  alreadyuploaded = []; downloadfiles = [];updateFile =[];  viewQuiz = []; classobject= [];  filepathtoupload = **undefined**;  type = **0**;  ngOnInit() {  **this**.classobject =**this**.\_user.getSelectedClass();  **this**.downloadfiles = **this**.splitFilestring(**this**.activity['filedir\_fld']);  **this**.dupefile = [];  **this**.getWorks();  }  ionViewWillEnter(){  **this**.works = [];  **this**.alreadyuploaded = [];  **this**.issubmitted = **0**;  **this**.isscored = **0**;  }  async getWorks(){  **const** loading = await **this**.\_loading.create({  message: 'Please wait...', });  loading.present();  **let** load = {  userid:**this.\_user.getUserID**(),  classcode:**this.\_user.getSelectedClass**().classcode\_fld,  actcode:**this.activity**['actcode\_fld'],  type:'act' }  **this**.\_ds.\_httpRequest('getstudworks',load,**1**).subscribe(async (res:**any**)=>{  res = **this**.\_user.\_decrypt(await res.a);  loading.dismiss();  **this**.works = **this**.\_user.\_convert(res.payload);  **if**(**this**.activity['type\_fld'] == **0**){  **this**.alreadyuploaded = **this**.splitFilestring(**this**.works[**0**].dir\_fld);  **this**.issubmitted = **this**.works[**0**].issubmitted\_fld;  **this**.isscored = **this**.works[**0**].isscored\_fld;  **this**.type = **this**.works[**0**].type\_fld  **if**(**this**.works[**0**].type\_fld == **2**){  **this**.alreadyuploaded = **this**.works[**0**].dir\_fld.split(',');  **this**.type = **this**.works[**0**].type\_fld  }  }**else** **if**(**this**.activity['type\_fld'] == **1**){  **this**.issubmitted = **this**.works[**0**].issubmitted\_fld;  **this**.isscored = **this**.works[**0**].isscored\_fld;  **if**(**this**.isscored == **1**){  **this**.getQuizResult(**this**.works[**0**].dir\_fld); }}  },(er) => { loading.dismiss();  **let** err = **this**.\_user.\_decrypt(er.error.a); }) }  async presentAlertPrompt(e) {  **const** alert = await **this**.alertController.create({  cssClass: 'my-custom-class',  inputs: [{name: 'link', type: 'url', placeholder: 'Add Link'},],  buttons: [{text: 'Cancel',role: 'cancel',cssClass:**secondary**', }, {text: 'Add',  handler: (data) => {  if(data.link.includes("https")){  this.type = 2;  this.files.push(data.link);  }else{  this.\_user.presentToast('Invalid Link!','danger');}}}]});  await alert.present();}  async getQuizResult(data){  const loading = await this.\_loading.create({  message: 'Please wait...',  });loading.present();  this.\_ds.\_httpRequest('getresult', { filepath: data }, 1).subscribe((res: any) => {  res = this.\_user.\_decrypt(res.a);  this.viewQuiz = res.payload; loading.dismiss();  },er=>{loading.dismiss();})}  //getFile in the storage  getFile(event) {  this.type = 1  let sum: number = 0;  for (var i = 0; i < event.target.files.length; i++) {  sum += event.target.files[i].size;  if(this.dupefile.includes(event.target.files[i].name)){  this.\_user.presentToast('Files already exist!','danger');  return;  }else{ this.files.push(event.target.files[i]);  this.dupefile.push(event.target.files[i].name); }}  if (sum > 10000000) {  this.\_user.presentToast((sum / 1048576).toFixed(2).toString() + "File to big error",'danger');  this.files = [];  }else{if (this.files.length > 0) {  if (this.dueDate < this.\_user.getDateToday()) {  this.withfile = 2;  }else{this.withfile = 1; }} }  this.displayCloseIcon = false; }  //end of getFile  async Filepath(){  let formdata: FormData;  formdata = this.uploadFile(this.files);  return new Promise((resolve, rejects) => {  this.\_ds.\_httpRequest('upload/' + this.\_user.getUserID(), formdata, 3).subscribe((data: any) => {  data = this.\_user.\_decrypt(data.a);  resolve(data.payload.filepath);  }, er => {er = this.\_user.\_decrypt(er.error.a);  rejects(er.error.a) });}) }  uploadFile(payload) {  const formData = new FormData();  for (var i = 0; i < this.files.length; i++) {  formData.append('file[]', this.files[i]); }  formData.append('payload', JSON.stringify(payload));  return formData; }  //submit work  async addWork(e,i){  let load = {};  if(i == 'Submit'){  const loading = await this.\_loading.create({  message: 'Please wait...', });  loading.present();  if(this.files.length == 0){  this.\_user.presentToast("Please add file/s first!",'danger'); return; }  if(this.type == 1){  await this.Filepath().then((filepath:any)=>{  new Promise<void>(() => {  load = { data:{  classcode\_fld:this.\_user.getSelectedClass().classcode\_fld,  actcode\_fld: this.activity['actcode\_fld'],  studnum\_fld:this.\_user.getUserID(),  type\_fld: 1,  dir\_fld:filepath,  issubmitted\_fld: this.withfile  },notif:{  id: this.\_user.getUserID(),  recipient: this.classobject['empcode\_fld'],  message: this.\_user.getFullname()+' submitted a file on ' + this.classobject['subjdesc\_fld'] ,  module: 'activities'} } });  }).catch(e=>{});  }else if( this.type == 2){  load = {  data:{  classcode\_fld:this.\_user.getSelectedClass().classcode\_fld,  actcode\_fld: this.activity['actcode\_fld'],  studnum\_fld:this.\_user.getUserID(),  type\_fld: 2,  dir\_fld:this.files.join(','),  issubmitted\_fld: 1  },notif:{  id: this.\_user.getUserID(),  recipient: this.classobject['empcode\_fld'],  message: this.\_user.getFullname()+' submitted a file on ' + this.classobject['subjdesc\_fld'] ,  module: 'activities'} } }  this.\_ds.\_httpRequest('addwork',load,1).subscribe((res:any)=>{ res = this.\_user.\_decrypt(res.a); loading.dismiss();  this.works = this.\_user.\_convert(res.payload);  this.files = [];  this.alreadyuploaded = this.splitFilestring(this.works[0].dir\_fld);  this.type = this.works[0].type\_fld  if(this.works[0].type\_fld == 2){  this.alreadyuploaded = this.works[0].dir\_fld.split(',');  this.type = this.works[0].type\_fld  }  this.issubmitted = this.works[0].issubmitted\_fld;  this.isscored = this.works[0].isscored\_fld;  this.displayCloseIcon = true;  this.getWorks();  },er => { er = this.\_user.\_decrypt(er.error.a); })  } else{ this.displayCloseIcon = false; this.saveBtn = true; } }  async save(e){  const loading = await this.\_loading.create({  message: 'Please wait...',  }); loading.present();  if(this.files.length == 0 && this.alreadyuploaded.join()== ""){  this.\_user.presentToast("Please add file/s first!",'danger');  return; }  if(this.type == 1){  if(!(this.files.length == 0)) {  await this.Filepath().then((filepath:any)=>{  new Promise<void>(() => {  this.filepathtoupload = filepath; })});} }  for (let index = 0; index < this.alreadyuploaded.length; index++) {  if(this.works[0].type\_fld != 2){  if(this.alreadyuploaded[index].name == ''){  this.alreadyuploaded = [];  }else{  let name = this.alreadyuploaded[index].name;  let link = this.removeBaseUrl(this.alreadyuploaded[index].link);  this.alreadyuploaded[index] = name +"?"+ link; }  }else{ if(this.alreadyuploaded.join(',') == ''){  this.filepathtoupload = this.files.join(',') }  if(this.files.length == 0){  this.filepathtoupload = this.alreadyuploaded.join(',') }  if(this.files.length != 0 && this.alreadyuploaded.length != 0){  this.filepathtoupload = this.files.join(',') + ',' + this.alreadyuploaded.join(',') } } }  if(this.type == 1 ){  if(this.filepathtoupload == undefined){  this.filepathtoupload = '';  this.filepathtoupload = this.alreadyuploaded.join(":");  }else if(this.alreadyuploaded.join(':') == ''){  this.alreadyuploaded = [];  this.filepathtoupload = this.filepathtoupload;  }else{  this.filepathtoupload = this.filepathtoupload+ ":" + this.alreadyuploaded.join(":");} }  if(this.updateFile.length != 0){  for (let index = 0; index < this.updateFile.length; index++) {  this.deleteFile(this.updateFile[index]); }  this.updateFile = []; }  let load = {  data:{  dir\_fld: this.filepathtoupload,  isscored\_fld: 0 }  ,notif:{  id: this.\_user.getUserID(),  recipient: this.classobject['empcode\_fld'],  message: this.\_user.getFullname() + ' save a work **in** ' + this.classobject['subjdesc\_fld'] ,  module: 'classroom' } }  let api = `editsubmit/${this.works[0].submitcode\_fld}/${this.works[0].actcode\_fld}`  this.\_ds.\_httpRequest(api, load ,1).subscribe((res:any)=>{  res = this.\_user.\_decrypt(res.a); loading.dismiss();  this.dupefile =[];  this.alreadyuploaded = this.splitFilestring(res.payload[0].dir\_fld)  if(this.works[0].type\_fld == 2){  this.alreadyuploaded = res.payload[0].dir\_fld.split(',')  }  this.files = [];  this.filepathtoupload = undefined;  this.displayCloseIcon = true;  this.saveBtn = false;  },er=>{loading.dismiss()  er = this.\_user.\_decrypt(er.error.a); }) }  async deleteFile(path){  let load = { dir\_fld:path }  return new Promise((resolve,rejects)=>{  this.\_ds.\_httpRequest('deletefile',load,1).subscribe((res:any)=>{ res = this.\_user.\_decrypt(res.a);  resolve(res.status.remarks);  },er=>{ er = this.\_user.\_decrypt(er.error.a);  return new Promise(()=>{  rejects(er.status.remarks) })}) })}  async removeFile(i,data) {  if(data == 'add'){  this.files.splice(i, 1);  this.dupefile.splice(i,1);  }else{  if(this.works[0].type\_fld != 2){  this.dupefile.splice(i,1);  this.updateFile.push(this.removeBaseUrl(this.alreadyuploaded[i].link)); }this.alreadyuploaded.splice(i,1); }  if (this.files.length == 0 && this.alreadyuploaded.join() == '') {  this.files = []; this.alreadyuploaded = []; } }  removeBaseUrl(data){  let substring = data.replace(this.\_ds.fileUrl, "");  return substring;  }  async preview(e,data){  const modal = await this.\_modal.create({  component: PreviewPage,  componentProps: { preview: data }  });await modal.present();}  splitFilestring(filestring: string) {  let arr1 = filestring.split(':');  let filearray: { name: string; link: string }[] = [];  for (let i = 0; i < arr1.length; i++) {  let arr2 = arr1[i].split('?');  this.dupefile.push(arr2[0]);  filearray.push({ name: arr2[0], link: this.\_ds.fileUrl + arr2[1] });}return filearray; }  back(){ this.\_modal.dismiss(); }  async presentPopover(ev,f,i,option){  const popover = await this.\_popover.create({  component: PopoverPage,  componentProps: { classfeed: f ,  option: option, index: i  },cssClass: 'my-popover-**class**',  mode:'ios', event: ev, translucent: true});  await popover.present();  popover.onDidDismiss().then((dataReturned) => {  console.log(dataReturned.data['option']);  if(dataReturned.role == 'backdrop'){ return }  if(dataReturned.data['option'] == 'file'){  this.type = 1;  }else if(dataReturned.data['option'] == 'url'){  this.type = 2; }  }); return await popover.present(); }  async ViewSubmit(e, data) {  const modal = await this.\_modal.create({  component: DrawerPage,  componentProps: { activity: this.activity },  cssClass: 'Submissions', swipeToClose: **true**,  });modal.onDidDismiss().then((dataReturned) => {});  **return** await modal.present(); }  disclose(){ **this**.\_modal.dismiss(); } } |

**Linkify**

|  |
| --- |
| transform(link: **string**): **string** {  **return** **this**.linkify(link);  }  **private** linkify(plainText: **string**){  **let** replacedText;  **let** replacePattern1;  **let** replacePattern2;  **let** replacePattern3;  *//URLs starting with http://, https://, or ftp://*  replacePattern1 = /(\b(https?|ftp):\/\/[-A-Z0-9+&@#\/%?=~\_|!:,.;]\*[-A-Z0-9+&@#\/%=~\_|])/gim;  replacedText = plainText.replace(replacePattern1, '<a href="$1" target="\_blank">$1</a>');  *//URLs starting with "www." (without // before it, or it'd re-link the ones done above).*  replacePattern2 = /(^|[^\/])(www\.[\S]+(\b|$))/gim;  replacedText = replacedText.replace(replacePattern2, '$1<a href="http://$2" target="\_blank">$2</a>');  *//Change email addresses to mailto:: links.*  replacePattern3 = /(([a-zA-Z0-9\-\\_\.])+@[a-zA-Z\\_]+?(\.[a-zA-Z]{2,6})+)/gim;  replacedText = replacedText.replace(replacePattern3, '<a href="mailto:$1">$1</a>');  **return** replacedText; } }  popover.page.ts  @Input() classfeed : **any**;  @Input() option : **string**;  @Input() classcomment: **any**;  @Input() forumContent = [];  @Input() news = [];  @Input() newsObject = [];  @Input() index = **0**;  @Input() postIndex = **0**;  classobject = [];  ngOnInit() {  **if**(**this**.option != 'forums'){  **this**.classobject =**this**.\_user.getSelectedClass();  }}  async editpost(e,option){  e.preventDefault();  **let** modal = await **this**.\_modal.create({  component: **EditPostPage**,  componentProps:{ classfeed: **this.classfeed** }  })  modal.onWillDismiss().then(datareturned =>{  **this**.\_popover.dismiss(datareturned.data,'edit');  }); **return** await modal.present(); }  async editcomment(e, option){  **this**.\_popover.dismiss({option: **option**, data: **this.classcomment**, index: **this.index**, postIndex: **this.postIndex**})  }  async editreply(e, option){  **this**.\_popover.dismiss({data:**this.classcomment**,option:'reply comment',index: **this.index**, postIndex: **this.postIndex**});  }  async editSubForumContent(e, option){  **this**.\_popover.dismiss({option: 'edit', data: **this.forumContent**, postIndex: **this.index**})  }  async editNewsComment(e, option){  **this**.\_popover.dismiss({option: 'edit', data: **this.news**})  }  async deletepost(e, option){  **let** load = {  data:{  classcode\_fld:**this.\_user.getSelectedClass**().classcode\_fld,  authorid\_fld:**this.\_user.getUserID**(),  isdeleted\_fld:**1**  },notif:{  id: **this.\_user.getUserID**(),  recipient: **this.classobject**['empcode\_fld'],  message: **this.\_user.getFullname**() + ' deleted a post on ' + **this**.classobject['subjdesc\_fld'] ,  module: 'classroom' } }  **const** alert = await **this**.\_alert.create({  cssClass: 'my-custom-class',  header: 'Delete Post?',  message: 'Are you sure you want to delete this post?',  buttons: [{text:'Cancel',role:'cancel',cssClass:'secondary',},{  text: 'Delete', handler: () => {  **this**.\_ds.\_httpRequest('editpost/'+ **this**.classfeed['postcode\_fld'] , load , **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a);  },er=>{ er = **this**.\_user.\_decrypt(er.error.a);  **this**.\_user.presentToast('Successfully Deleted!','dark');  **this**.\_popover.dismiss(**this**.index,**null**); });}}]  }); await alert.present();}  async deleteSubForumContent(e, option){  **let** load = {  data:{isdeleted\_fld:**1**},  notif:{  id: **this.\_user.getUserID**(),  recipient: **this.\_user.getUserID**(),  message: **this.\_user.getFullname**() + ' deleted a comment in a thread.' ,  module: 'forums'} }  **const** alert = await **this**.\_alert.create({  cssClass: 'my-custom-class',  header: 'Delete Post?',  message: 'Are you sure you want to delete this post?',  buttons:[{text: 'Cancel',role: 'cancel',cssClass: 'secondary',},{  text: 'Delete',  handler: () => {  **this**.\_ds.\_httpRequest('editreply/'+**this**.forumContent['contentcode\_fld'] , load , **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a);  },er=>{er = **this**.\_user.\_decrypt(er.error.a);  **this**.\_user.presentToast('Successfully Deleted!','dark');  **this**.\_popover.dismiss({data:'',option:'delete'}); });}}] }); await alert.present(); }  async deletecomment(e, option){  **let** load = {  data:{ classcode\_fld:**this.\_user.getSelectedClass**().classcode\_fld,  authorid\_fld:**this.\_user.getUserID**(),  isdeleted\_fld:**1** }  ,notif:{  id: **this.\_user.getUserID**(),  recipient: **this.classobject**['empcode\_fld'],  message: **this.\_user.getFullname**() + ' deleted a comment on ' + **this**.classobject['subjdesc\_fld'] ,  module: 'activities'} }  **const** alert = await **this**.\_alert.create({  cssClass: 'my-custom-class',  header: 'Delete Post?',  message: 'Are you sure you want to delete this post?',  buttons:[{text:'Cancel',role: 'cancel',cssClass:'secondary',}, {  text: 'Delete',  handler: () => { **let** load = { data:{  classcode\_fld:**this.classcomment**['commentcode\_fld'] ,  authorid\_fld:**this.\_user.getUserID**(),  isdeleted\_fld:**1** },  notif:{ id:**this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message: **this.\_user.getFullname**()+' deleted a comment '+ **this**.classcomment['commentcode\_fld'] ,  module: 'classroom'}}  **this**.\_ds.\_httpRequest('editcomm/'+**this**.classcomment['commentcode\_fld'], load , **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a);  }, er => {er = **this**.\_user.\_decrypt(er.error.a);  **this**.\_user.presentToast('Successfully Deleted!','dark'); **this**.\_popover.dismiss({data:**this.classcomment**,option:'delete comment',index: **this.index**, postIndex: **this.postIndex**});})}}]  }); await alert.present();}  async deletereply(e, option){  **const** alert = await **this**.\_alert.create({  cssClass: 'my-custom-class',  header: 'Delete Post?',  message: 'Are you sure you want to delete this post?',  buttons: [{text: 'Cancel', role: 'cancel', cssClass: 'secondary',  }, { text: 'Delete', handler: () => {**let** load = {data:{  classcode\_fld:**this.classcomment.reply**[**this**.index].commentcode\_fld , authorid\_fld:**this.\_user.getUserID**(),isdeleted\_fld:**1**},  notif:{ id:**this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld, message: **this.\_user.getFullname**()+' deleted a comment '+ **this**.classcomment.reply[**this**.index].commentcode\_fld ,  module: 'classroom'}}  **this**.\_ds.\_httpRequest('editcomm/'+**this**.classcomment.reply[**this**.index].commentcode\_fld, load , **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a);  }, er => { er = **this**.\_user.\_decrypt(er.error.a);  **this**.classcomment.reply.splice(**this**.index,**1**);  **this**.classcomment.replycount -= **1**;  **this**.\_user.presentToast('Successfully Deleted!','dark'); **this**.\_popover.dismiss({data:**this.classcomment**,option:'reply delete',index: **this.index**, postIndex: **this.postIndex**}); }) } }]  }); await alert.present();}  async deleteNewsComment(e, option){  **let** load = {data:{  authorid\_fld:**this.\_user.getUserID**(),  isdeleted\_fld:**1**,  },notif:{ id: **this.\_user.getUserID**(),  recipient: **this.newsObject**['recipientcode\_fld'],  message: **this.\_user.getFullname**() + ' deleted a '+ option + ' on ' + **this**.newsObject['title\_fld'] ,  module: 'news'} }  **const** alert = await **this**.\_alert.create({  cssClass: 'my-custom-class',  header: 'Delete Post?',  message: 'Are you sure you want to delete this post?', buttons: [{ text: 'Cancel', role: 'cancel', cssClass: 'secondary',  }, {text: 'Delete', handler: () => {  **this**.\_ds.\_httpRequest('editcomm/'+ **this**.news['commentcode\_fld'] , load , **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a);  },er=>{er = **this**.\_user.\_decrypt(er.error.a);  **this**.\_user.presentToast('Successfully Deleted!','dark');  **this**.\_popover.dismiss({data:'',option:'delete'}); });}}]  });await alert.present(); }  async replySubForumContent(e,option){  **this**.\_popover.dismiss({option: 'reply', data: **this.forumContent**}) }  async replycomment(e,option){  **this**.\_popover.dismiss({option: 'reply', data: **this.classcomment**})}  async replynews(e,option){  **this**.\_popover.dismiss({option: 'reply', data: **this.news**})}} |

**Add post**

|  |
| --- |
| classobject = []; postobject: **any** = []; files: **any** = [];  dupefile = [];  textcontent: **any**;  withfile= **0**;  filepathtoupload = '';  fileButton:**boolean** = **true**;photoButton:**boolean** = **true**;  videoButton:**boolean** = **true**; isUploading: **boolean**;  index = **0**; img1 : **any**;  img = ['jpg','gif','bmp','png'];  video = ['avi','mpg','m4v','mp4'];  filext = ['txt', 'doc' , 'docx' ,'html' ,'htm' ,'odt' ,'pdf' ,'xls', 'xlsx' ,'ods', 'pdf' ,'pptx', 'ppt'];  ngOnInit() {  **this**.classobject =**this**.\_user.getSelectedClass(); }  *//get profile pic*  getProfilePic(data){  **if**(data ==" " || data == **null** || data ==""){  **return** "assets/img/profile.jpg";  }**else**{ **return** **this**.\_ds.fileUrl + data; } }  *//end of get profile pic*  async addpost(e){  e.preventDefault();  **const** loading = await **this**.\_loading.create({  message: 'Please wait...', });  **if**(**this**.withfile > **0**){  await **this**.Filepath().then((filepath:**any**)=>{  **this**.filepathtoupload = filepath; }); }  **if**(**this**.withfile == **0** && **this**.textcontent == **undefined**){  **this**.\_user.presentToast("Enter Something!",'danger');  **return**; }loading.present();  **let** load = { data:{  classcode\_fld: **this.classobject**['classcode\_fld'],  authorid\_fld: **this.\_user.getUserID**(),  content\_fld: **this.textcontent**,  withfile\_fld: **this.withfile**,  dir\_fld: **this.filepathtoupload**  }, notif:{ id: **this.\_user.getUserID**(),  recipient: **this.classobject**['empcode\_fld'],  message: **this.\_user.getFullname**() + ' created a post on ' + **this**.classobject['subjdesc\_fld'] ,  module: 'classroom' } }  **this**.\_ds.\_httpRequest('addpost',load,**1**).subscribe((res:**any**)=>{ res = **this**.\_user.\_decrypt(res.a);  **this**.textcontent = '';  **this**.files = [];  **this**.filepathtoupload='';  **this**.\_modal.dismiss(res.payload);  loading.dismiss();  **this**.\_user.presentToast('Post added.','dark');  **this**.\_quest.UpdateCountAchievement(Crud.ADD,fldNames.piccnt\_fld); },er => {loading.dismiss();  er = **this**.\_user.\_decrypt(er.error.a);  **this**.\_user.presentToast('Invalid Post!','danger'); }) }  *//remove one in the file list*  removeFile(i) {  **this**.files.splice(i, **1**);  **this**.dupefile.splice(i,**1**);  **if** (**this**.files.length == **0**) {  **this**.withfile = **0**;  **this**.fileButton = **true**;  **this**.photoButton = **true**;  **this**.videoButton = **true**; } }  *//end of remove*  async getFile(e) {  e.preventDefault();  **let** sum: **number** = **0**;  **for** (**var** i = **0**; i < e.target.files.length; i++) {  sum += e.target.files[i].size;  **if**(**this**.dupefile.includes(e.target.files[i].name)){  **this**.\_user.presentToast('Files already exist!','danger');  **return**;  }**else**{  **this**.files.push(e.target.files[i]);  **this**.dupefile.push(e.target.files[i].name);  }  **var** ext = **this**.getext(**this**.files[i].name);  **if**(**this**.img.includes(ext)){  **this**.fileButton = **false**;  **this**.photoButton = **true**;  **this**.videoButton = **false**;  **this**.withfile = **1**;  }**else** **if**(**this**.video.includes(ext)){  **this**.fileButton = **false**;  **this**.photoButton = **false**;  **this**.videoButton = **true**;  **this**.withfile = **2**;  }**else**{  **this**.fileButton = **true**;  **this**.photoButton = **false**;  **this**.videoButton = **false**;  **this**.withfile = **5**; }}  *//view img*  **if** (e.target.files && e.target.files[**0**]) {  **let** reader = **new** FileReader();  reader.onload = (e:**any**) => {  **this**.img1 = e.target.result; }  reader.readAsDataURL(e.target.files[**0**]); *// to trigger onload}*  *//end view img*  **if** (sum > **30000000**) {  **this**.\_user.presentToast((sum / **3048576**).toFixed(**2**).toString() + "File to big error", 'danger');  **this**.files = [];  }**else**{  **if** (**this**.files.length == **0**) {**this**.withfile = **0**; } }}  async Filepath(){  **let** formdata: **FormData**;  formdata = **this**.uploadFile(**this**.files);  **return** **new** Promise((resolve, rejects) => {  **this**.\_ds.\_httpRequest('upload/' + **this**.\_user.getUserID(), formdata, **3**).subscribe((data: **any**) => {  data = **this**.\_user.\_decrypt(data.a);  resolve(data.payload.filepath);  }, er => {er = **this**.\_user.\_decrypt(er.error.a);  rejects(er.error.a) });}) }  uploadFile(payload) {  **this**.isUploading = **true**;  **const** formData = **new** FormData();  **for** (**var** i = **0**; i < **this**.files.length; i++) {  formData.append('file[]', **this**.files[i]); }  formData.append('payload', JSON.stringify(payload));  **return** formData; }  splitFilestring(filestring: **string**) {  **let** arr1 = filestring.split(':');  **let** filearray: { name: **string**; link: **string** }[] = [];  **for** (**let** i = **0**; i < arr1.length; i++) {  **let** arr2 = arr1[i].split('?');  filearray.push({ name: **arr2**[**0**], link: **this.\_ds.baseURL** + arr2[**1**] });} **return** filearray; }  getFilename(name) {  **return** name.replace(/^.\*[\\\/]/, ''); }  getfileExtension(filename) {  **let** regex = **new** RegExp('[^.]+$');  **if**(filename.match(regex) == 'docx'){ **return** '#004db3'; }  **if**(filename.match(regex) == 'ppt'){ **return** '#c34524'; }  **if**(filename.match(regex) == 'txt'){ **return** '#95a5a6'; }  **if**(filename.match(regex) == 'pdf'){ **return** '#ad0b00'; }  }  getext(data) { **return** data.split('.').pop(); }  addVideoUrl(array) {  **for** (**let** i = **0**; i < array.length; i++) {  **if** (array[i].content\_fld.includes('youtube.com') || array[i].content\_fld.includes('google.com/file/d/')) {  array[i].video\_fld = **this**.getVideoEmbedURL(array[i].content\_fld);  } **else** {array[i].video\_fld = **null**; } } **return** array; }  *// Addedd by Melner*  getVideoEmbedURL(content: **string**): any {  **let** replacePattern1 = /(\b(https?|ftp):\/\/[-A-Z0-9+&@#\/%?=~\_|!:,.;]\*[-A-Z0-9+&@#\/%=~\_|])/gim;  **let** replacedText = content.replace(replacePattern1, '<\*>$1<\*>');  **let** content\_arr: **string**[] = replacedText.split('<\*>');  **let** a: **any** = content\_arr.filter((x) => {  **if** (x.includes('http')) { **return** x }; });  **for** (**let** i = **0**; i < a.length; i++) {  **if** (a[i].includes('youtube.com')) {  a[i] = a[i].replace('watch?v=', 'embed/'); }  **else** {a[i] = a[i].replace('view?usp=sharing', 'preview');  }a[i] = **this**.sanitizer.bypassSecurityTrustResourceUrl(a[i]);  } **return** a; }  back(){ **this**.\_modal.dismiss('exit'); }  edit-post.page.ts  @Input() classfeed;  textcontent ='';  files: **any** = []; dupefile = []; alreadyuploaded = [];  fileButton:**boolean** = **true**;  photoButton:**boolean** = **true**;  videoButton:**boolean** = **true**;  withfile: **number**;  filepathtoupload: **any**;  index = **0**;i = **0**;  img = ['jpg','gif','bmp','png'];  video = ['avi','mpg','m4v','mp4'];  filext = ['txt', 'doc' , 'docx' ,'html' ,'htm' ,'odt' ,'pdf' ,'xls', 'xlsx' ,'ods', 'pdf' ,'pptx', 'ppt'];  updateFile =[]; classobject = [];    ngOnInit() {  **this**.classobject =**this**.\_user.getSelectedClass();  **this**.textcontent = **this**.classfeed['content\_fld'];  **this**.withfile = **this**.classfeed['withfile\_fld']  **if**(**this**.classfeed['dir\_fld'] == ''){  **this**.alreadyuploaded = [];  }**else**{  **this**.alreadyuploaded = **this**.splitFilestring(**this**.classfeed['dir\_fld']);  **if**(**this**.img.includes(**this**.getext(**this**.alreadyuploaded[**0**].name))){ **this**.fileButton = **false**;  **this**.photoButton = **true**;  **this**.videoButton = **false**;  }**else** **if**(**this**.video.includes(**this**.getext(**this**.alreadyuploaded[**0**].name))){ **this**.fileButton = **false**;  **this**.photoButton = **false**;  **this**.videoButton = **true**;  }**else** { **this**.fileButton = **true**;  **this**.photoButton = **false**;  **this**.videoButton = **false**; } } }  async editpost(e){  **const** loading = await **this**.\_loading.create({  message: 'Please wait...', });  **if**(**this**.withfile == **0** && **this**.textcontent.length == **0**){  **this**.\_user.presentToast("Enter Something!",'danger');  **return**; }  **if**(**this**.files.length != **0**) {  await **this**.Filepath().then((filepath:**any**)=>{  **this**.filepathtoupload = filepath; }); }  **for** (**let** index = **0**; index < **this**.alreadyuploaded.length; index++) {  **if**(**this**.alreadyuploaded[index].name == ''){  **this**.alreadyuploaded = [];  }**else**{ **let** name = **this**.alreadyuploaded[index].name;  **let** link = **this**.removeBaseUrl(**this**.alreadyuploaded[index].link);  **this**.alreadyuploaded[index] = name +"?"+ link; } }    **if**(**this**.filepathtoupload == **undefined**){  **this**.filepathtoupload = '';  **this**.filepathtoupload = **this**.alreadyuploaded.join(":");  }**else** **if**(**this**.alreadyuploaded.join(':') == ''){  **this**.alreadyuploaded = [];  **this**.filepathtoupload = **this**.filepathtoupload;  }**else**{  **this**.filepathtoupload = **this**.filepathtoupload + ":" + **this**.alreadyuploaded.join(":");}  **if**(**this**.updateFile.length != **0**){  **for** (**let** index = **0**; index < **this**.updateFile.length; index++) {  **this**.deleteFile(**this**.updateFile[index]); } **this**.updateFile = [];}  loading.present();  **let** load = { data:{ content\_fld: **this.textcontent**,  dir\_fld: **this.filepathtoupload**,  withfile\_fld: **this.withfile**  }, notif:{ id: **this.\_user.getUserID**(),  recipient: **this.classobject**['empcode\_fld'],  message: **this.\_user.getFullname**() + ' edited a post on ' + **this**.classobject['subjdesc\_fld'] ,  module: 'classroom' } }  **this**.\_ds.\_httpRequest('editpost/'+ **this**.classfeed['postcode\_fld'] , load , **1**).subscribe((res) => {  res = **this**.\_user.\_decrypt(res.a); **this**.files = [];  **this**.alreadyuploaded = [];  **this**.\_modal.dismiss(res.payload);  **this**.\_user.presentToast('Successfully Updated','dark');  loading.dismiss();  },er=>{ loading.dismiss(); }); }  async getFile(e) {  **let** sum: **number** = **0**;  **for** (**var** i = **0**; i < e.target.files.length; i++) {  sum += e.target.files[i].size;  **if**(**this**.dupefile.includes(e.target.files[i].name) || **this**.alreadyuploaded.includes(e.target.files[i].name)){  **this**.\_user.presentToast('Files already exist!','danger');  **return**; }**else**{ **this**.files.push(e.target.files[i]);  **this**.dupefile.push(e.target.files[i].name); }  **var** ext = **this**.getext(**this**.files[i].name);  **if**(**this**.img.includes(ext)){  **this**.fileButton = **false**;  **this**.photoButton = **true**;  **this**.videoButton = **false**;  **this**.withfile = **1**;  }**else** **if**(**this**.video.includes(ext)){  **this**.fileButton = **false**;  **this**.photoButton = **false**;  **this**.videoButton = **true**;  **this**.withfile = **2**;  }**else**{  **this**.fileButton = **true**;  **this**.photoButton = **false**;  **this**.videoButton = **false**;  **this**.withfile = **5**; } }  **if** (sum > **30000000**) {  **this**.\_user.presentToast((sum / **3048576**).toFixed(**2**).toString() + "File to big error",'danger');  }**else**{ **if** (**this**.files.length == **0**) { **this**.withfile = **0**; } } }  async Filepath(){  **let** formdata: **FormData**;  formdata = **this**.uploadFile(**this**.files);  **return** **new** Promise((resolve, rejects) => {  **this**.\_ds.\_httpRequest('upload/' + **this**.\_user.getUserID(), formdata, **3**).subscribe((data: **any**) => {  data = **this**.\_user.\_decrypt(data.a);  resolve(data.payload.filepath);  }, er => { er = **this**.\_user.\_decrypt(er.error.a);  rejects(er.error.a) }); }) }  uploadFile(payload) {  **const** formData = **new** FormData();  **for** (**var** i = **0**; i < **this**.files.length; i++) {  formData.append('file[]', **this**.files[i]); }  formData.append('payload', JSON.stringify(payload));  **return** formData }  *//remove one in the file list*  async removeFile(i,data) {  **if**(data == 'add'){  **this**.files.splice(i, **1**);  **this**.dupefile.splice(i,**1**);  }**else** { **this**.dupefile.splice(i,**1**); **this**.updateFile.push(**this**.removeBaseUrl(**this**.alreadyuploaded[i].link)); **this**.alreadyuploaded.splice(i,**1**); }  **if**(**this**.files.length == **0** && **this**.alreadyuploaded.join() == '' ) {  **this**.withfile = **0**;  **this**.files = [];**this**.alreadyuploaded = [];  **this**.fileButton = **true**;  **this**.videoButton = **true**;  **this**.photoButton = **true**; } }  async deleteFile(path){  **let** load = { dir\_fld:**path** }  **return** **new** Promise((resolve,rejects)=>{  **this**.\_ds.\_httpRequest('deletefile',load,**1**).subscribe((res:**any**)=>{ res = **this**.\_user.\_decrypt(res.a);  resolve(res.status.remarks);  },er=>{ er = **this**.\_user.\_decrypt(er.error.a);  **return** **new** Promise(()=>{ rejects(er.status.remarks) }) }) }) }  removeBaseUrl(data){  **let** substring = data.replace(**this**.\_ds.baseURL, "");  **return** substring;  }  *//end of remove*  splitFilestring(filestring: **string**) {  **let** arr1 = filestring.split(':');  **let** filearray: { name: **string**; link: **string** }[] = [];  **for** (**let** i = **0**; i < arr1.length; i++) {  **let** arr2 = arr1[i].split('?');  **this**.dupefile.push(arr2[**0**]);  filearray.push({ name: **arr2**[**0**], link: **this.\_ds.baseURL** + arr2[**1**] }); } **return** filearray; }  getFilename(name) { **return** name.replace(/^.\*[\\\/]/, ''); }  getfileExtension(filename) {  **let** regex = **new** RegExp('[^.]+$');  **if**(filename.match(regex) == 'docx'){ **return** '#004db3'; }  **if**(filename.match(regex) == 'ppt'){ **return** '#c34524'; }  **if**(filename.match(regex) == 'txt'){ **return** '#95a5a6'; }  **if**(filename.match(regex) == 'pdf'){ **return** '#ad0b00'; }}  getext(data) { **return** data.split('.').pop(); }  getProfilePic(data){  **if**(data ==" " || data == **null** || data ==""){  **return** "assets/img/profile.jpg";  }**else**{ **return** **this**.\_ds.fileUrl + data; } }  *//end of get profile pic*  addVideoUrl(array) {  **for** (**let** i = **0**; i < array.length; i++) {  **if** (array[i].content\_fld.includes('youtube.com') || array[i].content\_fld.includes('google.com/file/d/')) { array[i].video\_fld = **this**.getVideoEmbedURL(array[i].content\_fld); }  **else** { array[i].video\_fld = **null**; } } **return** array; }  *// Addedd by Melner*  getVideoEmbedURL(content: **string**): any {  **let** replacePattern1 = /(\b(https?|ftp):\/\/[-A-Z0-9+&@#\/%?=~\_|!:,.;]\*[-A-Z0-9+&@#\/%=~\_|])/gim;  **let** replacedText = content.replace(replacePattern1, '<\*>$1<\*>');  **let** content\_arr: **string**[] = replacedText.split('<\*>');  **let** a: **any** = content\_arr.filter((x) => {  **if** (x.includes('http')) { **return** x}; });  **for** (**let** i = **0**; i < a.length; i++) {  **if** (a[i].includes('youtube.com')) {  a[i] = a[i].replace('watch?v=', 'embed/');; }  **else** { a[i] = a[i].replace('view?usp=sharing', 'preview'); }  a[i] = **this**.sanitizer.bypassSecurityTrustResourceUrl(a[i]); }  **return** a; }  async disabled(e){  **this**.\_user.presentToast('Disabled Button!','danger');  }  back(){**this**.\_modal.dismiss('exit'); } } |

**Preview**

|  |
| --- |
| @Input() preview: **any**;  @Input() newsPreview: **any**;  img = ['jpg', 'gif', 'bmp', 'png', 'jpeg'];  video = ['avi', 'mpg', 'm4v', 'mp4'];  filext = ['txt', 'doc', 'docx', 'html', 'htm', 'odt', 'pdf', 'xls','xlsx', 'ods', 'pdf','pptx', 'ppt',];  getext(data) { **return** data.split('.').pop(); }  download(data) {  **var** request: **DownloadRequest** = {  uri: **data**['link'], title: **data**['name'], description: '',  mimeType: '', visibleInDownloadsUi: **true**,  notificationVisibility: **NotificationVisibility.VisibleNotifyCompleted**,  destinationInExternalFilesDir: { dirType: 'Downloads',  subPath: **data**['name'], }, };  **this**.downloader  .download(request)  .then((location: **string**) =>  **this**.\_user.presentToast('File downloaded at:' + location, 'dark'))  .**catch**((error: **any**) => console.error(error)); }  removeExt(data) {  **let** link = data.slice(**0**, data.lastIndexOf('.'));  **return** link; }  sanitizer(data) {**return** **this**.\_sanitizer.bypassSecurityTrustResourceUrl(data); }  previewDoc(data) { **return** **this**.sanitizer(  'https://docs.google.com/gview?url=' + data + '&embedded=true' ); }  back() { **this**.\_modal.dismiss(); } } |

**Slides**

|  |
| --- |
| @Input() quiz = [];  @Input() page ='login';  @ViewChild(IonSlides, {**static**: **true**}) slides: **IonSlides**;  slideOpts = { initialSlide: **0**, speed: **400** };  question : **any**[];  ans:'';  questionObject =[{  questionid: '',  question: '',  questiontype: '',  choices:''  }];  answerObject = [{  questionid: '',  question: [],  answer: ''  }];  quizScore = **0**; totalscore =**0**;quiztotal = **0**;passingscore = **0**;  questionaire;  previewScore = **false**;  actcode; classcode; progress = **0**;  classobject = [];  confirm = **false**;  progress\_value = **0**;  quizObject =[]; answerkeyObject = [];  quizpath = "";  async ngOnInit() {  **if**(**this**.page == 'quiz'){  **this**.classobject =**this**.\_user.getSelectedClass();  **this**.getquiz();} }  async getStarted(){  **this**.\_storage.set('tutorial', **true** );  **this**.\_route.navigate(['login']);}  async updateSlide(e){ e.target.update(); }  splitChoices(**string**) {**return** **string**.split('//\*//'); }  getquiz(){  **this**.\_ds.\_httpRequest('getquiz',{filepath: **this.quiz**['filedir\_fld']},**1**).subscribe(async (res:**any**)=>{  res = **this**.\_user.\_decrypt(res.a);  **this**.quizObject = await res.payload.quiz;  **this**.answerkeyObject = await res.payload.answerkey.content;  **this**.questionObject = **this**.arrayshuffle(res.payload.quiz['content']);  **for** (**let** i = **0**; i < **this**.questionObject.length - **1**; i++) {  **this**.answerObject.push({ questionid: '', question: [], answer: '' })} }); }  getQuestionImage(data){ **return** **this**.\_ds.fileUrl + data; }  arrayshuffle(data){  **let** newPos:**number**, temp:**number**;  **for** (**let** i = data.length - **1**; i > **0**; i--) {  newPos = Math.floor(Math.random() \* ( i+**1** ));  temp = data[i];  data[i] = data[newPos];  data[newPos] = temp; } **return** data; }  async setAnswers(i, answer) {  **this**.questionaire = **this**.questionObject[i]  **this**.answerObject[i] = {  question: **this.questionaire**['question'],  questionid: **this.questionaire**,  answer: **answer.target.value** }; }  async addAnswer(e,id){  **this**.questionaire = **this**.questionObject[id];  **this**.answerObject[id] = ({  question:**this.questionaire**['question'],  questionid: **this.questionaire**,  answer: **this.answerObject**[id].answer }) }  calculateQuiz(){  **let** answer = **this**.answerObject  **let** answerkey = **this**.answerkeyObject  **let** totalScore = **0**  **let** totalQuizScore = **0**  answer.forEach((element, i) => {  **let** key = answerkey.find((item: **any** ) => { **return** element.questionid['questionid'] == item.questionid });  answer[i]['points'] = **0**  answer[i]['mark'] = **0**  totalQuizScore += key.score  **let** studAnswer = element.answer.trim()  **let** quizAnswer = key.answer.trim()  **if**(studAnswer.toUpperCase() == quizAnswer.toUpperCase()){  answer[i]['points'] = key.score  answer[i]['mark'] = **1**  totalScore += key.score} })  **let** answerJson = {  totalscore: **totalScore**,  quiztotal: **totalQuizScore**,  result: **answer** } **return** answerJson }  async submitQuiz(e){  **const** alert = await **this**.\_alert.create({  cssClass: 'my-custom-class',  message: 'Submit Quiz?',  buttons: [{text: 'Cancel', role: 'cancel',cssClass: 'secondary',  }, {text: 'Yes', handler: () => {**this**.confirm = **true**;  **for** (**let** index = **0**; index < **this**.answerObject.length; index++) { **if**(**this**.answerObject[index] != **null** && **this**.answerObject[index].question.length != **0** ){  }**else**{  **let** emptyAnswer = [{  question: "Not Answer Yet!",  questionid: **index.toString**(),  answer: "Not Answer Yet!" } **this**.answerObject[index].question = emptyAnswer } }  **for** (**let** index = **0**; index < **this**.answerObject.length; index++) {  **if**(**this**.answerObject[index].question[**0**].question == "Not Answer Yet!"){  **this**.\_user.presentToast('Please Answer all the items!','danger'); **return**; } }  **let** load = { data:{  filepath:**this.quiz**['filedir\_fld'],  answer: **this.calculateQuiz**()  },notif:{ id:**this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message: **this.\_user.getFullname**()+' submitted the quiz ' + **this**.classobject['subjdesc\_fld'],  module: 'classroom' } }    **this**.\_ds.\_httpRequest('addanswer', load, **1**).subscribe((res: **any**) => {res = **this**.\_user.\_decrypt(res.a);  **this**.quizpath = res.payload.filepath;  **this**.quizScore = res.payload.result;  **this**.totalscore = res.payload.result.totalscore;  **this**.quiztotal = res.payload.result.quiztotal;  **this**.passingscore = **this**.quiztotal \* .**6**;  **this**.saveScore(res.payload.filepath);  }, er => { er = **this**.\_user.\_decrypt(er.error.a); }) } } ]  }); alert.present(); }  saveScore(data){ **let** load = {data:{  classcode\_fld:**this.\_user.getSelectedClass**().classcode\_fld,  actcode\_fld: **this.quiz**['actcode\_fld'],  studnum\_fld:**this.\_user.getUserID**(),  type\_fld:**this.quiz**['type\_fld'],  dir\_fld: **data**,  issubmitted\_fld: **1**,  score\_fld: **this.totalscore**,  isscored\_fld: **1**,  },notif:{ id:**this.\_user.getUserID**(),  recipient: **this.\_user.getSelectedClass**().empcode\_fld,  message: **this.\_user.getFullname**()+' submitted the quiz on' + **this**.classobject['subjdesc\_fld'],  module: 'activities' } }  **this**.\_ds.\_httpRequest('addwork',load,**1**).subscribe((res:**any**)=>{ res = **this**.\_user.\_decrypt(res.a);  **this**.\_modal.dismiss({data: **res.payload**, score: **this.totalscore**, quiztotal: **this.quiztotal**});  },er => { er = **this**.\_user.\_decrypt(er.error.a); }) } } |

**Tabs**

|  |
| --- |
| getPage = "classes";  async openOER(e) {  await Browser.open({ url: 'https://chedro3.ched.gov.ph/open-educational-resources/' });  }  async setNav(data){ **this**.getPage = data; }  async viewProfile(e){ **this**.\_router.navigate(['main/tabs/profile',{ id:**this.getPage** }]);}  ProfilePic(data){  **if**(data ==" " || data == **null** || data ==""){  **return** "assets/img/profile.jpg";  }**else**{ **return** **this**.\_ds.fileUrl + data; } }  async logout(){  await **this**.\_storage.create();  **const** loading = await **this**.\_loading.create({  message:'Please Wait...'  }).then(async (res) => {  await **this**.\_storage.set('autologin',{ us: "", ps:"" ,isAuth: **true**} ).then(async (res) =>{  await **this**.\_storage.get('autologin').then(async (res)=>{  await **this**.\_router.navigateByUrl('login'); })});})}} |

**App**

|  |
| --- |
| initializeApp() {  **this**.platform.ready().then(() => {  **this**.statusBar.styleBlackTranslucent();  **this**.splashScreen.hide();  **this**.\_storage.create();  **this**.\_storage.get('tutorial').then((data)=>{  **if**(data){} **else**{**this**.\_router.navigate(['slide']); }});  **this**.platform.backButton.subscribeWithPriority(**10**, (processNextHandler) => {  **if** (**this**.\_location.isCurrentPathEqualTo('/login')) {  **this**.showExitConfirm();  processNextHandler();  } **else** {  **this**.\_location.back();} });  **this**.platform.backButton.subscribeWithPriority(**5**, () => {  **this**.\_alert.getTop().then(r => {  **if** (r) {navigator['app'].exitApp();}  }).**catch**(e => {console.log(e); })});}) }  showExitConfirm() {  **this**.\_alert.create({  header: 'App termination',  message: 'Do you want to close the app?',  backdropDismiss: **false**,  buttons: [{text: 'Stay',role: 'cancel',  }, {text: 'Exit', handler: () => {  navigator['app'].exitApp();}}]})  .then(alert => {alert.present();});}} |