McDonalds eLearning Application

TECHNICAL DOCUMENT

William Nolan | C00216986 | April 3rd, 2020 | Chris Meudec

Contents

App

app-routing,module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { RouteReuseStrategy } from '@angular/router';
import { HttpClientModule } from '@angular/common/http';

import { IonicModule, IonicRouteStrategy } from '@ionic/angular';
import { SplashScreen } from '@ionic-native/splash-screen/ngx';
import { StatusBar } from '@ionic-native/status-bar/ngx';

import { AppComponent } from './app.component';
import { AppRoutingModule } from './app-routing.module';
import firebaseConfig from './firebase';
import { AngularFireModule } from '@angular/fire';
import { AngularFireAuthModule } from '@angular/fire/auth';
import { ServiceWorkerModule } from '@angular/service-worker';
import { environment } from '../environments/environment';
import { BrowserAnimationsModule } from '@angular/platform-
browser/animations';

@NgModule({
  declarations: [AppComponent],
  entryComponents: [],
  imports: [
    BrowserModule,
    IonicModule.forRoot(),
    AppRoutingModule,
    AngularFireModule.initializeApp(firebaseConfig),
    AngularFireAuthModule,
    HttpClientModule,
    ServiceWorkerModule.register('ngsw-
worker.js', { enabled: environment.production }),
    BrowserAnimationsModule
  ],
  providers: [
    StatusBar,
    SplashScreen,
    { provide: RouteReuseStrategy, useClass: IonicRouteStrategy }
  ],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

app.component.html

```
<ion-app>
  <ion-menu content-id="main" side="start" menuId="navId">
    <ion-header>
      <ion-toolbar>
        <ion-title>
          McDonald's eLearning
        </ion-title>
      </ion-toolbar>
    </ion-header>

    <ion-content>
      <ion-list>
        <ion-menu-toggle menuId="navId">
          <ion-item lines="none" routerLink="view-soc">
            <ion-icon name="list" slot="start"></ion-icon>
            <ion-label>My SOCs</ion-label>
          </ion-item>
        </ion-menu-toggle>
        <ion-menu-toggle menuId="navId" *ngIf="!isLoading">
          <ion-item lines="none" routerLink="{{ resultsUrl }}">
            <ion-icon name="Star" slot="start"></ion-icon>
            <ion-label>My Results</ion-label>
          </ion-item>
        </ion-menu-toggle>
        <ion-menu-
toggle menuId="navId" *ngIf="!isLoadingRole && userRole > 0">
          <ion-item lines="none" routerLink="review-soc">
            <ion-icon name="clipboard" slot="start"></ion-icon>
            <ion-label>Review SOCs</ion-label>
          </ion-item>
        </ion-menu-toggle>
        <ion-menu-toggle menuId="navId">
          <ion-item lines="none" routerLink="leaderboard">
            <ion-icon name="trophy" slot="start"></ion-icon>
            <ion-label>Leaderboard</ion-label>
          </ion-item>
        </ion-menu-toggle>
        <ion-menu-toggle menuId="navId">
          <ion-item lines="none" routerLink="{{ progressionUrl }}">
            <ion-icon name="trending-up" slot="start"></ion-icon>
            <ion-label>My Progression</ion-label>
          </ion-item>
        </ion-menu-toggle>
        <ion-menu-
toggle menuId="navId" *ngIf="!isLoadingRole && userRole > 0">
          <ion-item lines="none" routerLink="review-progression">
            <ion-icon name="trending-up" slot="start"></ion-icon>
            <ion-label>Review Progression</ion-label>
          </ion-item>
        </ion-menu-toggle>
        <ion-menu-
toggle menuId="navId" *ngIf="!isLoadingRole && userRole > 1">
          <ion-item lines="none" routerLink="admin">
            <ion-icon name="contact" slot="start"></ion-icon>
            <ion-label>Admin</ion-label>
          </ion-item>
        </ion-menu-toggle>
        <ion-menu-toggle menuId="navId">
          <ion-item lines="none" (click)="onLogout()">
            <ion-icon name="exit" slot="start"></ion-icon>
            <ion-label>Logout</ion-label>
          </ion-item>
        </ion-menu-toggle>
      </ion-list>
    </ion-content>
  </ion-menu>
  <ion-router-outlet id="main"></ion-router-outlet>
</ion-app>
```

app.component.ts

```
import { Component } from '@angular/core';

import { Platform } from '@ionic/angular';
import { SplashScreen } from '@ionic-native/splash-screen/ngx';
import { StatusBar } from '@ionic-native/status-bar/ngx';
import { AuthService } from './services/auth.service';
import { Router } from '@angular/router';

@Component({
  selector: 'app-root',
  templateUrl: 'app.component.html',
  styleUrls: ['app.component.scss']
})
export class AppComponent {
  userId: string;
  userRole: number;
  resultsUrl: string;
  progressionUrl: string;
  isLoading = false;
  isLoadingRole = false;

  constructor(
    private platform: Platform,
    private splashScreen: SplashScreen,
    private statusBar: StatusBar,
    private authService: AuthService,
    private router: Router,
  ) {
    this.initializeApp();
  }

  initializeApp() {
    this.isLoading = true;
    this.authService.userId.subscribe(userId => {
      this.userId = userId;
      this.resultsUrl = '/my-results/' + userId;
      this.progressionUrl = '/my-progression/' + userId;
      this.isLoading = false;
    });
    this.isLoadingRole = true;
    this.authService.userRole.subscribe(role => {
      this.userRole = role;
      this.isLoadingRole = false;
    });
    this.platform.ready().then(() => {
      this.statusBar.styleDefault();
      this.splashScreen.hide();
    });
  }

  onLogout() {
    this.authService.logout();
    this.router.navigateByUrl('/auth');
  }
}
```

app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { RouteReuseStrategy } from '@angular/router';
import { HttpClientModule } from '@angular/common/http';

import { IonicModule, IonicRouteStrategy } from '@ionic/angular';
import { SplashScreen } from '@ionic-native/splash-screen/ngx';
import { StatusBar } from '@ionic-native/status-bar/ngx';

import { AppComponent } from './app.component';
import { AppRoutingModule } from './app-routing.module';
import firebaseConfig from './firebase';
import { AngularFireModule } from '@angular/fire';
import { AngularFireAuthModule } from '@angular/fire/auth';
import { ServiceWorkerModule } from '@angular/service-worker';
import { environment } from '../environments/environment';
import { BrowserAnimationsModule } from '@angular/platform-
browser/animations';

@NgModule({
  declarations: [AppComponent],
  entryComponents: [],
  imports: [
    BrowserModule,
    IonicModule.forRoot(),
    AppRoutingModule,
    AngularFireModule.initializeApp(firebaseConfig),
    AngularFireAuthModule,
    HttpClientModule,
    ServiceWorkerModule.register('ngsw-
worker.js', { enabled: environment.production }),
    BrowserAnimationsModule
  ],
  providers: [
    StatusBar,
    SplashScreen,
    { provide: RouteReuseStrategy, useClass: IonicRouteStrategy }
  ],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

firebase.ts

```
const firebaseConfig = {
    apiKey: 'AIzaSyCXS3IGKZp2xeyw_3e2ZSi9-5TiYuvJh-Y',
    authDomain: 'fyp-wnolan.firebaseapp.com',
    databaseURL: 'https://fyp-wnolan.firebaseio.com',
    projectId: 'fyp-wnolan',
    storageBucket: 'fyp-wnolan.appspot.com',
    messagingSenderId: '949115855043',
    appId: '1:949115855043:web:3ef58f0e680c758d598276',
    measurementId: 'G-S6K9XLDBNW'
};

export default firebaseConfig;
```

Auth

auth.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-title>{{ isLogin ? 'Login' : 'Sign Up'}}</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content class="ion-padding">
  <form #authForm="ngForm" (ngSubmit)="onSubmit(authForm)">
    <ion-grid>
      <ion-row>
        <ion-col size-sm="6" offset-sm="3">
          <ion-list>
            <div *ngIf="!isLogin">
              <ion-item>
                <ion-label position="floating">First Name</ion-label>
                <ion-input
                  type="text"
                  ngModel
                  name="fname"
                  required
                ></ion-input>
              </ion-item>
              <ion-item>
                <ion-label position="floating">Last Name</ion-label>
                <ion-input
                  type="text"
                  ngModel
                  name="lname"
                  required
                ></ion-input>
              </ion-item>
            </div>
            <ion-item>
              <ion-label position="floating">E-Mail</ion-label>
              <ion-input
                type="email"
                ngModel
                name="email"
                required
                email
                #emailCtrl="ngModel"
              ></ion-input>
            </ion-item>
            <ion-
item *ngIf="!emailCtrl.valid && emailCtrl.touched" lines="none">
              <ion-label>Should be a valid email address.</ion-label>
            </ion-item>
            <ion-item>
              <ion-label position="floating">Password</ion-label>
              <ion-input
                type="password"
                ngModel
                name="password"
                required
                minlength="8"
                #passwordCtrl="ngModel"
              ></ion-input>
            </ion-item>
            <ion-
item *ngIf="!passwordCtrl.valid && passwordCtrl.touched" lines="none">
              <ion-label>Should at least be 8 characters long.</ion-label>
            </ion-item>
          </ion-list>
        </ion-col>
      </ion-row>
      <ion-row>
        <ion-col size-sm="6" offset-sm="3">
          <ion-button
            type="button"
            color="primary"
            fill="clear"
            expand="block"
            (click)="onSwitchAuthMode()"
          >
            Switch to {{ isLogin ? 'Sign Up' : 'Login'}}
          </ion-button>
          <ion-button
            type="submit"
            color="primary"
            expand="block"
            [disabled]="!authForm.valid"
          >
          {{ isLogin ? 'Login' : 'Sign Up'}}
          </ion-button>
        </ion-col>
      </ion-row>
    </ion-grid>
  </form>
</ion-content>
```

auth.page.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: Typescript file for the auth page.
 */
import { Component, OnInit } from '@angular/core';
import { AuthService, AuthResponseData } from '../services/auth.service';
import { Router } from '@angular/router';
import { LoadingController, AlertController } from '@ionic/angular';
import { NgForm } from '@angular/forms';
import { Observable } from 'rxjs';

@Component({
  selector: 'app-auth',
  templateUrl: './auth.page.html',
  styleUrls: ['./auth.page.scss'],
})
export class AuthPage implements OnInit {
  isLoading = false;
  isLogin = true;

  constructor(
    private authService: AuthService,
    private router: Router,
    private loadingCtrl: LoadingController,
    private alertCtrl: AlertController,
  ) { }

  ngOnInit() {
  }

  authenticateLogin(email: string, password: string) {
    this.isLoading = true;
    this.loadingCtrl
      .create({ keyboardClose: true, message: 'Logging in...' })
      .then(loadingEl => {
        loadingEl.present();
        let authObs: Observable<AuthResponseData>;
        authObs = this.authService.login(email, password);
        authObs.subscribe(resData => {
          this.authService.updateCurrUser(resData.localId).subscribe(() =>
 {
            this.isLoading = false;
            loadingEl.dismiss();
            this.router.navigateByUrl('view-soc');
          });
        }, errRes => {
          loadingEl.dismiss();
          const code = errRes.error.error.message;
          let message = 'Could not log you in, please try again';
          if (code === 'EMAIL_NOT_FOUND') {
            message = 'Email address could not be found';
          } else if (code === 'INVALID_PASSWORD') {
            message = 'This password is incorrect';
          }
          this.showAlert(message);
        });
      });
  }
  authenticateSignUp(email: string, password: string, fname: string, lname
: string) {
    this.isLoading = true;
    let generatedId: string;
    this.loadingCtrl
      .create({ keyboardClose: true, message: 'Signing up...' })
      .then(loadingEl => {
        loadingEl.present();
        let authObs: Observable<AuthResponseData>;
        authObs = this.authService.signUp(email, password);
        authObs.subscribe(resData => {
          this.authService.createUser(resData.localId, email, fname, lname
).subscribe();
          this.isLoading = false;
          loadingEl.dismiss();
          this.router.navigateByUrl('view-soc');
          generatedId = resData.localId;
        }, errRes => {
          loadingEl.dismiss();
          const code = errRes.error.error.message;
          let message = 'Could not sign you up, please try again';
          if (code === 'EMAIL_EXISTS') {
            message = 'This email address already exists!';
          }
          this.showAlert(message);
        });
      });
    this.isLogin = true;
  }

  onSubmit(form: NgForm) {
    if (!form.valid) {
      return;
    }
    const email = form.value.email;
    const password = form.value.password;
    if (this.isLogin) {
      this.authenticateLogin(email, password);
    } else {
      const fname = form.value.fname;
      const lname = form.value.lname;
      const role = form.value.role;
      console.log(role);
      this.authenticateSignUp(email, password, fname, lname);
    }
    form.reset();
  }

  private showAlert(message: string) {
    this.alertCtrl.create(
      {
        header: 'Authentication failed',
        message,
        buttons: ['Ok']
      }
    )
    .then(alertEl =>
      alertEl.present()
    );
  }

  onSwitchAuthMode() {
    this.isLogin = !this.isLogin;
  }

}
```

Models

allResults.model.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: All Results object.
 */
import { Result } from './result.model';

export class AllResults {
    constructor(
        public socId: string,
        public results: Result[],
    ) {}
}
```

feedback.model.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: Feedback object.
 */
export class Feedback {
    constructor(
        public id: string,
        public feedback: string,
        public senderName: string,
        public date: Date,
    ) {}
}
```

leaderboard.model.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: Leaderboard object.
 */
export class Leaderboard {
    constructor(
        public id: string,
        public name: string,
        public score: number,
        public date: Date,
    ) {}
}
```

result.model.ts

```
/**
 * Name:        Wiliam Nolan
 * Student ID:  C00216986
 * Description: Result object.
 */
import { Feedback } from './feedback.model';

export class Result {
    constructor(
        public id: string,
        public result: number,
        public total: number,
        public incorrect: string[],
        public feedback: Feedback[],
        public date: Date,
    ) {}
}
```

soc-answer.model.ts

```
/**
 * Name:        Wiliam Nolan
 * Student ID:  C00216986
 * Description: SOC Answer object.
 */
export class SocAnswer {
    constructor(
      public id: string,
      public name: string,
      public isAnswer: boolean,
    ) {}
}
```

soc-question.model.ts

```
/**
 * Name:        Wiliam Nolan
 * Student ID:  C00216986
 * Description: SOC Question object.
 */
import { SocAnswer } from './soc-answer.model';
```

```
export class SocQuestion {
    constructor(
      public id: string,
      public name: string,
      public answers: SocAnswer[],
    ) {}
}
```

soc.model.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: SOC object.
 */
import { SocQuestion } from './soc-question.model';

export class Soc {
    constructor(
      public id: string,
      public name: string,
      public description: string,
      public percent: number,
      public questions: SocQuestion[]
    ) {}
}
```

user.model.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: User object.
 */
export class User {
    role: any;
    constructor(
        public id: string,
        public email: string,
        private _token: string,
        private tokenExperationDate: Date
    ) {}

    get token() {
        if (!this.tokenExperationDate || this.tokenExperationDate <= new D
ate()) {
            return null;
        }
        return this._token;
    }
}
```

userData.model.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: User data object.
 */
export class UserData {
    constructor(
        public id: string,
        public email: string,
        public fname: string,
        public lname: string,
        public role: number,
        public socs: string[],
    ) {}
}
```

Admin

admin-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { AdminPage } from './admin.page';

const routes: Routes = [
  {
    path: '',
    component: AdminPage
  },
  {
    path: 'change-role/:userId',
    loadChildren: () => import('../Admin/change-role/change-
role.module').then( m => m.ChangeRolePageModule)
  }
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule],
})
export class AdminPageRoutingModule {}
```

admin.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-menu-button menuId="navId"></ion-menu-button>
    </ion-buttons>
    <ion-title>Admin</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-
searchbar showCancelButton="focus" (ionInput)="filter($event)"></ion-
searchbar>
  <ion-list>
    <ion-item
      *ngFor = "let item of listUsers"
      [routerLink]="['/', 'admin', 'change-role', item.id]"
      detail
    >
      <ion-label>{{ item.fname }} {{ item.lname }}</ion-label>
    </ion-item>
  </ion-list>
</ion-content>
```

admin.page.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: Typescript file for the admin page.
 */
import { Component, OnInit, OnDestroy } from '@angular/core';
import { UserData } from 'src/app/models/userData.model';
import { Subscription } from 'rxjs';
import { AuthService } from 'src/app/services/auth.service';

@Component({
  selector: 'app-admin',
  templateUrl: './admin.page.html',
  styleUrls: ['./admin.page.scss'],
})
export class AdminPage implements OnInit, OnDestroy {
  loadedUsers: UserData[];
  listUsers: UserData[];
  fullName: string;
  private usersSub: Subscription;
  isLoading = false;
  isItemAvailable = false;

  constructor(
    private authService: AuthService
  ) { }

  ngOnInit() {
    this.usersSub = this.authService.users.subscribe(users => {
      this.listUsers = users;
      this.loadedUsers = users;
    });
  }

  ionViewWillEnter() {
    this.isLoading = true;
    this.authService.fetchUsers().subscribe(() => {
      this.isLoading = false;
    });
  }

  initializeItems() {
    this.listUsers = this.loadedUsers;
  }

  filter(event: any) {
    this.initializeItems();
    this.fullName = '';
    const val = event.target.value;
    console.log(val);
    if (val && val.trim() !== '') {
      this.isItemAvailable = true;
      this.listUsers = this.listUsers.filter((item) => {
        this.fullName = item.fname + ' ' + item.lname;
        return (this.fullName.toLowerCase().indexOf(val.toLowerCase()) > -
1);
      });
    }
  }


  ngOnDestroy() {
    if (this.usersSub) {
      this.usersSub.unsubscribe();
    }
  }
}
```

Change Role

change-role.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-back-button defaultHref="admin"></ion-back-button>
    </ion-buttons>
    <ion-title>Change Role</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
 <form [formGroup]="form">
    <ion-grid>
      <ion-row>
        <ion-col size-sm="6" offset-sm="3">
          <ion-item>
            <ion-label position="floating">Role</ion-label>
            <ion-input
              type="number"
              formControlName="role"
            ></ion-input>
          </ion-item>
        </ion-col>
      </ion-row>
      <ion-row>
        <ion-col size-sm="6" offset-sm="3">
          <ion-
button color="primary" size="full" (click)="onSubmit()" [disabled]="!form.
valid">Submit</ion-button>
        </ion-col>
      </ion-row>
    </ion-grid>
  </form>
</ion-content>
```

change-role.page.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: Typescript file for the change role page.
 */
import { Component, OnInit } from '@angular/core';
import { FormGroup, FormBuilder, FormControl, Validators } from '@angular/
forms';
import { LoadingController } from '@ionic/angular';
import { AuthService } from 'src/app/services/auth.service';
import { ActivatedRoute, Router } from '@angular/router';
import { UserData } from 'src/app/models/userData.model';

@Component({
  selector: 'app-change-role',
  templateUrl: './change-role.page.html',
  styleUrls: ['./change-role.page.scss'],
})
export class ChangeRolePage implements OnInit {
  form: FormGroup;
  selectedUser: UserData;

  constructor(
    private fb: FormBuilder,
    private loadingCtrl: LoadingController,
    private authService: AuthService,
    private route: ActivatedRoute,
    private router: Router,
  ) { }

  ngOnInit() {
    this.route.paramMap.subscribe(paramMap => {
      this.authService.getUser(paramMap.get('userId'))
        .subscribe(user => {
          this.selectedUser = user;
        });
    });
    this.form = this.fb.group({
      role: new FormControl(null, {
        updateOn: 'blur',
        validators: [Validators.required]
      }),
    });
  }

  onSubmit() {
    if (!this.form.valid) {
      return;
    }
    this.loadingCtrl.create({
      message: 'Updating role...'
    }).then(loadingEl => {
      loadingEl.present();
      this.route.paramMap.subscribe(paramMap => {
        this.authService.updateRole(
          this.form.value.role,
          this.selectedUser
        ).subscribe(() => {
          loadingEl.dismiss();
          this.form.reset();
          this.router.navigateByUrl('/admin');
        });
      });
    });
  }

}
```

Leaderboard

leaderboard-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { LeaderboardPage } from './leaderboard.page';

const routes: Routes = [
  {
    path: '',
    redirectTo: 'search-leaderboards',
    pathMatch: 'full'
  },
  {
    path: 'search-leaderboards',
    loadChildren: () => import('./search-leaderboards/search-
leaderboards.module').then( m => m.SearchLeaderboardsPageModule)
  },
  {
    path: ':socId',
    loadChildren: () => import('./soc-leaderboard/soc-
leaderboard.module').then( m => m.SocLeaderboardPageModule)
  }
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule],
})
export class LeaderboardPageRoutingModule {}
```

Search Leaderboards

search-leaderboards.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-menu-button menuId="navId"></ion-menu-button>
    </ion-buttons>
    <ion-title>Leaderboard</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-
searchbar showCancelButton="focus" (ionInput)="filter($event)"></ion-
searchbar>
  <div *ngIf="isLoading" class="ion-text-center">
    <ion-spinner color="primary"></ion-spinner>
  </div>
  <p *ngIf="!isLoading && loadedSocs.length <= 0" class="ion-text-center">
    No SOCs found.
  </p>
  <p *ngIf="!isLoading && loadedSocs.length > 0 && listSocs.length <= 0" c
lass="ion-text-center">
    SOC not found.
  </p>
  <ion-list *ngIf="!isLoading && listSocs.length > 0">
    <ion-item
      *ngFor="let soc of listSocs"
      [routerLink]="['/', 'leaderboard', soc.id]"
      detail
    >
    {{ soc.name }}
    </ion-item>
  </ion-list>
</ion-content>
```

search-leaderboards.page.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: Typescript file for the search leaderboard page.
 */
import { Component, OnInit, OnDestroy } from '@angular/core';
import { SocsService } from 'src/app/services/socs.service';
import { Subscription } from 'rxjs';
import { Soc } from 'src/app/models/soc.model';

@Component({
  selector: 'app-search-leaderboards',
  templateUrl: './search-leaderboards.page.html',
  styleUrls: ['./search-leaderboards.page.scss'],
})
export class SearchLeaderboardsPage implements OnInit, OnDestroy {
  private socSub: Subscription;
  loadedSocs: Soc[];
  listSocs: Soc[];
  isLoading = false;
  isItemAvailable = false;

  constructor(
    private socService: SocsService,
  ) { }

  ngOnInit() {
    this.socService.socs.subscribe(socs => {
      this.loadedSocs = socs;
      this.listSocs = socs;
    });
  }

  ionViewWillEnter() {
    this.isLoading = true;
    this.socService.fetchSocs().subscribe(() => {
      this.isLoading = false;
    });
  }

  initializeItems() {
    this.listSocs = this.loadedSocs;
  }

  filter(event: any) {
    this.initializeItems();
    const val = event.target.value;
    if (val && val.trim() !== '') {
      this.isItemAvailable = true;
      this.listSocs = this.listSocs.filter((item) => {
        return (item.name.toLowerCase().indexOf(val.toLowerCase()) > -1);
      });
    }
  }

  ngOnDestroy() {
    if (this.socSub) {
      this.socSub.unsubscribe();
    }
  }

}
```

SOC Leaderboard

soc-leaderboard.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-back-button defaultHref="leaderboard"></ion-back-button>
    </ion-buttons>
    <ion-
title>{{ isLoadingSoc ? 'Loading...' : soc.name + ' Leaderboard'}}</ion-
title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <div *ngIf="isLoading" class="ion-text-center">
    <ion-spinner color="primary"></ion-spinner>
  </div>
  <ion-grid *ngIf="!isLoading && leaderboard.length > 0">
    <ion-row class="border">
      <ion-col class="heading">
        Rank
      </ion-col>
      <ion-col class="heading">
        Name
      </ion-col>
      <ion-col class="heading">
        Score
      </ion-col>
      <ion-col class="heading">
        Date
      </ion-col>
    </ion-row>
    <ion-row
      class="borderLight"
      *ngFor="let record of leaderboard"
    >
      <ion-col>
        {{ leaderboard.indexOf(record) + 1}}
      </ion-col>
      <ion-col>
        {{ record.name }}
      </ion-col>
      <ion-col>
        {{ record.score }}
      </ion-col>
      <ion-col>
        {{ record.date | date: 'longDate' }}
      </ion-col>
    </ion-row>
  </ion-grid>
</ion-content>
```

soc-leaderboard.page.scss

```
.heading {
    font-weight: bold;
}

.border {
    border-bottom: solid 1px #000000;
    padding: 5px;
}

.borderLight {
    border-bottom: solid 1px #bbbbbb;
    padding: 5px;
}
```

soc-leaderboard.page.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: Typescript file for the soc leaderboard page.
 */
import { Component, OnInit } from '@angular/core';
import { Leaderboard } from 'src/app/models/Leaderboard.model';
import { Subscription } from 'rxjs';
import { ResultsService } from 'src/app/services/results.service';
import { NavController } from '@ionic/angular';
import { ActivatedRoute } from '@angular/router';
import { Soc } from 'src/app/models/soc.model';
import { SocsService } from 'src/app/services/socs.service';
import { LeaderboardService } from 'src/app/services/leaderboard.service';

@Component({
  selector: 'app-soc-leaderboard',
  templateUrl: './soc-leaderboard.page.html',
  styleUrls: ['./soc-leaderboard.page.scss'],
})
export class SocLeaderboardPage implements OnInit {
  leaderboard: Leaderboard[];
  soc: Soc;
  private leaderSub: Subscription;
  private socSub: Subscription;
  isLoading = false;
  isLoadingSoc = false;

  constructor(
    private resultService: ResultsService,
    private leaderService: LeaderboardService,
    private socService: SocsService,
    private route: ActivatedRoute,
    private navCtrl: NavController
  ) { }

  ngOnInit() {
    this.leaderSub = this.leaderService
        .leaderboard
        .subscribe(leaderboard => {
          this.leaderboard = leaderboard;
    });
    this.isLoadingSoc = true;
    this.route.paramMap.subscribe(paramMap => {
      if (!paramMap.has('socId')) {
        this.navCtrl.navigateBack('/socs/review');
        return;
      }
      this.socSub = this.socService
        .getSoc(paramMap.get('socId'))
        .subscribe(soc => {
          this.soc = soc;
          this.isLoadingSoc = false;
        });
    });
  }

  ionViewWillEnter() {
    this.isLoading = true;
    this.route.paramMap.subscribe(paramMap => {
      if (!paramMap.has('socId')) {
        this.navCtrl.navigateBack('/socs/review');
```

```
          return;
        }
        this.leaderSub = this.leaderService
          .fetchLeaderboard(paramMap.get('socId'))
          .subscribe(() => {
            this.isLoading = false;
          });
      });
    }

  }
```

My Progression

my-progression.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-menu-button menuId="navId"></ion-menu-button>
    </ion-buttons>
    <ion-title>My Progression</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-grid>
    <ion-row>
      <ion-col size-sm="6" offset-sm="3">
        <canvas #lineCanvas></canvas>
      </ion-col>
    </ion-row>
  </ion-grid>
</ion-content>
```

my-progression.page.scss

```
ion-grid {
    height: 100%;

    ion-row {
      height: 50%;
      ion-col {
        height: 100%;
      }
    }
  }
```

my-progression.page.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: Typescript file for the my progression page.
 */
import { Component, OnInit, OnDestroy, ViewChild, ElementRef } from '@angu
lar/core';
import { Chart } from 'chart.js';
import { Subscription } from 'rxjs';
import { Soc } from 'src/app/models/soc.model';
import { UserData } from 'src/app/models/userData.model';
import { ActivatedRoute } from '@angular/router';
import { NavController } from '@ionic/angular';
import { ResultsService } from 'src/app/services/results.service';
import { ReviewDetailService } from 'src/app/services/review-
detail.service';
import { AuthService } from 'src/app/services/auth.service';

@Component({
  selector: 'app-my-progression',
  templateUrl: './my-progression.page.html',
  styleUrls: ['./my-progression.page.scss'],
})
export class MyProgressionPage implements OnInit, OnDestroy {
  @ViewChild('lineCanvas', {static: false}) lineCanvas: ElementRef;

  private lineChart: Chart;
  reviewDetailSub: Subscription;
  userSub: Subscription;
  socs: Soc[];
  user: UserData;
  dataset: object[] = [];
  newData: object[] = [];
  colors: string[] = ['#3e95cd', '#8e5ea2', '#3cba9f', '#e8c3b9', '#c45850
', '#f57f17',
                      '#ff1744', '#d500f9', '#2979ff', '#00c853', '#bf360c
', '#5d4037',
                      '#546e7a', '#1a237e', '#006064', '#33691e', '#e65100
', '#ffd600',
                      ]; // ADD MORE COLORS
  isLoading = false;

  constructor(
    public route: ActivatedRoute,
    private navCtrl: NavController,
    public resultService: ResultsService,
    public reviewDetailService: ReviewDetailService,
    public authService: AuthService
  ) { }

  ngOnInit() {
    this.route.paramMap.subscribe(paramMap => {
      if (!paramMap.get('userId')) {
        this.navCtrl.navigateBack('/review-progression');
        return;
      }
      this.isLoading = true;
      this.userSub = this.authService
        .getUser(paramMap.get('userId'))
        .subscribe(user => {
          this.user = user;
          this.isLoading = false;
        });
      this.reviewDetailSub = this.reviewDetailService.getSocs(paramMap.get
('userId')).subscribe(socs => {
        console.log(socs);
        this.setChartData(paramMap.get('userId'), socs);
      });
    });
  }

  setChartData(userId: string, socs: Soc[]) {
    this.resultService.getResultObject(userId).subscribe(object => {
      const test = object;
      let index = 0;
      for (const soc in test) {
        if (test.hasOwnProperty(soc)) {
          this.newData = [];
          for (const result in test[soc]) {
            if (test[soc].hasOwnProperty(result)) {
              this.newData.push(
                {
                  x: test[soc][result].date,
                  y: test[soc][result].result / test[soc][result].total *
100
                }
              );
            }
          }
          this.dataset.push(
            {
              data: this.newData,
              label: socs[index].name,
              borderColor: this.colors[index],
              fill: false
            }
          );
          index++;
        }
      }
      this.setChart();
    });
  }

  setChart() {
    this.lineChart = new Chart(this.lineCanvas.nativeElement, {
      type: 'line',
      data: {
        datasets: this.dataset
      },
      options: {
          scales: {
              xAxes: [{
                  type: 'time',
                  time: {
                      unit: 'month',
                      displayFormats: {
                        day: 'MMM YY'
                      }
                  }
              }]
          },
          title: {
            display: true,
            text: 'My Progression'
          },
          elements: {
            line: {
                tension: 0
            }
          },
          maintainAspectRatio: false,
          responsive: true,
      }
    });
  }

  ngOnDestroy() {
    if (this.userSub) {
      this.userSub.unsubscribe();
    }
    if (this.reviewDetailSub) {
      this.reviewDetailSub.unsubscribe();
    }
  }
}
```

My Results

my-results-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { MyResultsPage } from './my-results.page';

const routes: Routes = [
  {
    path: '',
    component: MyResultsPage
  },
  {
    path: ':socId',
    loadChildren: () => import('./view-soc-result/view-soc-
result.module').then( m => m.ViewSocResultPageModule)
  },
  {
    path: ':socId/:resultId',
    loadChildren: () => import('./view-soc-result-detail/view-soc-result-
detail.module').then( m => m.ViewSocResultDetailPageModule)
  }
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule],
})
export class MyResultsPageRoutingModule {}
```

my-results.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-menu-button menuId="navId"></ion-menu-button>
    </ion-buttons>
    <ion-title>{{ isLoading ? 'Loading...' : user.fname }}'s SOCs</ion-
title>
  </ion-toolbar>
</ion-header>

<ion-content>
```

```
  <ion-
searchbar showCancelButton="focus" (ionInput)="filter($event)"></ion-
searchbar>
  <div *ngIf="isLoading" class="ion-text-center">
    <ion-spinner color="primary"></ion-spinner>
  </div>
  <p *ngIf="!isLoading && !isLoadingSoc && loadedSocs.length <= 0" class="
ion-text-center">
    User has no SOC results.
  </p>
  <p *ngIf="!isLoading && loadedSocs.length > 0 && listSocs.length <= 0" c
lass="ion-text-center">
    SOC not found.
  </p>
  <ion-list *ngIf="!isLoading && !isLoadingSoc && listSocs.length > 0">
    <ion-item
      *ngFor="let soc of listSocs"
      [routerLink]="['/', 'my-results', userId, soc.id]"
      detail
    >
    {{ soc.name }}
    </ion-item>
  </ion-list>
</ion-content>
```

my-results.page.ts

```
import { Component, OnInit, OnDestroy } from '@angular/core';
import { UserData } from 'src/app/models/userData.model';
import { Soc } from 'src/app/models/soc.model';
import { Subscription } from 'rxjs';
import { ActivatedRoute, Router } from '@angular/router';
import { NavController, AlertController } from '@ionic/angular';
import { AuthService } from 'src/app/services/auth.service';
import { ReviewDetailService } from 'src/app/services/review-
detail.service';

@Component({
  selector: 'app-my-results',
  templateUrl: './my-results.page.html',
  styleUrls: ['./my-results.page.scss'],
})
export class MyResultsPage implements OnInit, OnDestroy {
  user: UserData;
  loadedSocs: Soc[];
  listSocs: Soc[];
  userId: string;
  private userSub: Subscription;
  private reviewDetailSub: Subscription;
  isLoading = false;
  isLoadingSoc = false;
  isItemAvailable = false;

  constructor(
    private route: ActivatedRoute,
    private navCtrl: NavController,
    private authService: AuthService,
    private alertCtrl: AlertController,
    private router: Router,
    private reviewDetailService: ReviewDetailService,
  ) { }

  ngOnInit() {
    this.isLoadingSoc = true;
    this.reviewDetailSub = this.reviewDetailService.socs.subscribe(socs =>
{
      this.loadedSocs = socs;
      this.listSocs = socs;
      this.isLoadingSoc = false;
    });
    this.route.paramMap.subscribe(paramMap => {
      if (!paramMap.has('userId')) {
        this.navCtrl.navigateBack('/socs/review');
        return;
      }
      this.userId = paramMap.get('userId');
      this.isLoading = true;
      this.userSub = this.authService
        .getUser(paramMap.get('userId'))
        .subscribe(user => {
          this.user = user;
          this.isLoading = false;
        }, error => {
          this.alertCtrl.create({
            header: 'An error occured',
            message: 'Could not load User',
            buttons: [
              {
                text: 'Okay',
                handler: () => {
                  this.router.navigate(['socs/review']);
                }
              }
            ]
          }).then(alertEl => alertEl.present());
        });
    });
  }

  ionViewWillEnter() {
    this.isLoading = true;
    this.route.paramMap.subscribe(paramMap => {
      if (!paramMap.has('userId')) {
        this.navCtrl.navigateBack('/socs/review');
        return;
      }
      this.reviewDetailService.getSocs(paramMap.get('userId')).subscribe((
) => {
        this.isLoading = false;
      });
    });
  }

  ngOnDestroy() {
    if (this.userSub) {
      this.userSub.unsubscribe();
    }
    if (this.reviewDetailSub) {
      this.reviewDetailSub.unsubscribe();
    }
  }

  initializeItems() {
    this.listSocs = this.loadedSocs;
  }

  filter(event: any) {
    this.initializeItems();
    const val = event.target.value;
    if (val && val.trim() !== '') {
      this.isItemAvailable = true;
      this.listSocs = this.listSocs.filter((item) => {
        return (item.name.toLowerCase().indexOf(val.toLowerCase()) > -1);
      });
    }
  }

}
```

View SOC Result

view-soc-result.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-back-button defaultHref="my-results"></ion-back-button>
    </ion-buttons>
    <ion-title>Results</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <div *ngIf="isLoading" class="ion-text-center">
    <ion-spinner color="primary"></ion-spinner>
  </div>
  <ion-
grid *ngIf="!isLoading && !isLoadingResult && !isLoadingSoc && results.len
gth > 0">
    <ion-row class="border">
      <ion-col class="heading">
        Date
      </ion-col>
      <ion-col class="heading">
        Wrong Answers
      </ion-col>
      <ion-col class="heading">
        Result(%)
      </ion-col>
    </ion-row>
    <ion-row
      class="borderLight"
      *ngFor="let result of results"
      [routerLink]="['/', 'my-results', userId, soc.id, result.id]"
    >
      <ion-col>
        {{ result.date | date: 'longDate' }}
      </ion-col>
      <ion-col>
        {{ result.total - result.result }}
      </ion-col>
      <ion-col>
        {{ result.result / result.total * 100  | number: '1.0-0'}}%
      </ion-col>
    </ion-row>
  </ion-grid>
</ion-content>
```

view-soc-result.page.scss

```
.heading {
    font-weight: bold;
}

.border {
    border-bottom: solid 1px #000000;
    padding: 5px;
}

.borderLight {
    border-bottom: solid 1px #bbbbbb;
    padding: 5px;
}
```

view-soc-result.page.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: Typescript file for the view soc result page.
 */
import { Component, OnInit, OnDestroy } from '@angular/core';
import { Soc } from 'src/app/models/soc.model';
import { Subscription } from 'rxjs';
import { ActivatedRoute } from '@angular/router';
import { NavController } from '@ionic/angular';
import { Result } from 'src/app/models/result.model';
import { SocsService } from 'src/app/services/socs.service';
import { ResultsService } from 'src/app/services/results.service';

@Component({
  selector: 'app-view-soc-result',
  templateUrl: './view-soc-result.page.html',
  styleUrls: ['./view-soc-result.page.scss'],
})
export class ViewSocResultPage implements OnInit, OnDestroy {
  isLoading = false;
  isLoadingResult = false;
  isLoadingSoc = false;
  soc: Soc;
```

```
results: Result[];
percents: number[];
socId: string;
userId: string;
private socSub: Subscription;
private resultsSub: Subscription;

constructor(
  private socService: SocsService,
  private resultsService: ResultsService,
  private navCtrl: NavController,
  private route: ActivatedRoute
) { }

ngOnInit() {
  this.isLoadingResult = true;
  this.isLoadingSoc = true;
  this.route.paramMap.subscribe(paramMap => {
    if (!paramMap.get('socId')) {
      this.navCtrl.navigateBack('/review-soc');
      return;
    }
    if (!paramMap.get('userId')) {
      this.navCtrl.navigateBack('/review-soc');
      return;
    }
    this.socId = paramMap.get('socId');
    this.userId = paramMap.get('userId');
    this.socSub = this.socService.getSoc(this.socId).subscribe(soc => {
      this.soc = soc;
      this.isLoadingSoc = false;
    });
    this.resultsSub = this.resultsService
      .results
      .subscribe(results => {
        this.results = results;
        this.isLoadingResult = false;
      });
  });

}

ionViewWillEnter() {
  this.isLoading = true;
```

```
    this.resultsService
      .fetchResults(this.userId, this.socId)
      .subscribe(() => {
        this.isLoading = false;
    });
  }

  ngOnDestroy() {
    if (this.resultsSub) {
      this.resultsSub.unsubscribe();
    }
    if (this.socSub) {
      this.socSub.unsubscribe();
    }
  }

}
```

View SOC Result Detail

view-soc-result-detail.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-back-button defaultHref="my-results"></ion-back-button>
    </ion-buttons>
    <ion-title>Review Result</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <div *ngIf="isLoading" class="ion-text-center">
    <ion-spinner color="primary"></ion-spinner>
  </div>
  <ion-grid>
    <ion-row>
      <ion-col></ion-col>
      <ion-col><canvas #doughnutCanvas></canvas></ion-col>
      <ion-col></ion-col>
    </ion-row>
  </ion-grid>
  <ion-grid *ngIf="!isLoading">
    <ion-row>
      <ion-col size-sm="6" offset-sm="3">
        {{ result.date | date: 'longDate'}}
      </ion-col>
    </ion-row>
    <ion-row *ngFor="let question of socQuestions">
      <ion-col size-sm="6" offset-sm="3">
        {{ question.name }}
      </ion-col>
    </ion-row>
    <ion-row *ngFor="let item of feedback">
      <ion-col size-sm="6" offset-sm="3">
        <ion-card>
          <ion-card-header>
            <ion-card-title>{{ item.senderName }}</ion-card-title>
          </ion-card-header>
          <ion-card-content>
            <time>{{ item.date | date:'medium' }}</time>
            <p>{{ item.feedback}}</p>
          </ion-card-content>
        </ion-card>
      </ion-col>
    </ion-row>
  </ion-grid>
</ion-content>
```

view-soc-result-detail.page.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: Typescript file for the view soc result detail page.
 */
import { Component, OnInit, OnDestroy, ViewChild, ElementRef } from '@angu
lar/core';
import { Result } from 'src/app/models/result.model';
import { Soc } from 'src/app/models/soc.model';
import { SocQuestion } from 'src/app/models/soc-question.model';
import { Subscription } from 'rxjs';
import { ResultsService } from 'src/app/services/results.service';
import { ReviewDetailService } from 'src/app/services/review-
detail.service';
import { SocQuestionService } from 'src/app/services/soc-
question.service';
import { ActivatedRoute } from '@angular/router';
import { NavController } from '@ionic/angular';
import { Chart } from 'chart.js';
import { Feedback } from 'src/app/models/feedback.model';

@Component({
  selector: 'app-view-soc-result-detail',
  templateUrl: './view-soc-result-detail.page.html',
  styleUrls: ['./view-soc-result-detail.page.scss'],
})
export class ViewSocResultDetailPage implements OnInit, OnDestroy {
  @ViewChild('doughnutCanvas', {static: false}) doughnutCanvas: ElementRef
;

  private doughnutChart: Chart;
  result: Result;
  soc: Soc;
  socQuestions: SocQuestion[];
  socId: string;
  userId: string;
  resultId: string;
  isLoading = false;
  private resultSub: Subscription;
  private reviewDetailSub: Subscription;
  feedback: Feedback[];

  constructor(
    public resultService: ResultsService,
    public reviewDetailService: ReviewDetailService,
    public socQuestionService: SocQuestionService,
    public route: ActivatedRoute,
    private navCtrl: NavController,
  ) { }

  ngOnInit() {
    this.isLoading = true;
    this.route.paramMap.subscribe(paramMap => {
      if (!paramMap.get('socId')) {
        this.navCtrl.navigateBack('/socs/review');
        return;
      }
      if (!paramMap.get('userId')) {
        this.navCtrl.navigateBack('/socs/review');
        return;
      }
      if (!paramMap.get('resultId')) {
        this.navCtrl.navigateBack('/socs/review');
        return;
      }
      this.resultId = paramMap.get('resultId');
      this.socId = paramMap.get('socId');
      this.userId = paramMap.get('userId');
      this.getResults();
    });
  }

  getResults() {
    this.resultSub = this.resultService.getResult(this.resultId, this.socI
d, this.userId).subscribe(result => {
      this.result = result;
      console.log(this.result.feedback);
      const feedback = [];
      for (const key in this.result.feedback) {
        if (this.result.feedback.hasOwnProperty(key)) {
          feedback.push(new Feedback(
            key,
            this.result.feedback[key].feedback,
            this.result.feedback[key].senderName,
            this.result.feedback[key].date,
          ));
        }
      }
      this.feedback = feedback.sort((a, b) => {
        return b.date.localeCompare(a.date) || 0;
      });
      this.setDoughnut();
      this.getQuestions();
      this.isLoading = false;
    });
  }

  getQuestions() {
    if (this.result.incorrect !== undefined) {
      this.reviewDetailSub = this.reviewDetailService.getQuestions(this.re
sult.incorrect, this.socId).subscribe(questions => {
        this.socQuestions = questions;
      });
    }
  }
  setDoughnut() {
    this.doughnutChart = new Chart(this.doughnutCanvas.nativeElement, {
      type: 'doughnut',
      data: {
        labels: ['Correct', 'Incorrect'],
        datasets: [
          {
            data: [this.result.result, this.result.total - this.result.res
ult],
            backgroundColor: [
              '#00ff00',
              '#ff0000',
            ],
          }
        ]
      },
      options: {
        rotation: 1 * Math.PI,
        circumference: 1 * Math.PI,
      }
    });
  }

  ngOnDestroy() {
    if (this.resultSub) {
      this.resultSub.unsubscribe();
    }
    if (this.reviewDetailSub) {
      this.reviewDetailSub.unsubscribe();
    }
  }
}
```

Review Progression

review-progression-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

const routes: Routes = [
  {
    path: '',
    redirectTo: 'search',
    pathMatch: 'full'
  },
  {
    path: 'search',
    loadChildren: () => import('./search/search.module').then( m => m.Sear
chPageModule)
  },
  {
    path: ':userId',
    loadChildren: () => import('./review-user-progression/review-user-
progression.module').then( m => m.ReviewUserProgressionPageModule)
  }
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule],
})
export class ReviewProgressionPageRoutingModule {}
```

Review User Progression

review-user-progression.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-menu-button menuId="navId"></ion-menu-button>
    </ion-buttons>
    <ion-
title>Review {{ isLoading ? 'Loading...' : user.fname }}'s Progression</io
n-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-grid>
    <ion-row>
      <!-- <ion-col></ion-col> -->
      <ion-col size-sm="6" offset-
sm="3"><canvas #lineCanvas></canvas></ion-col>
      <!-- <ion-col></ion-col> -->
    </ion-row>
  </ion-grid>
</ion-content>
```

review-user-progression.page.scss

```
ion-grid {
    height: 100%;

    ion-row {
      height: 50%;
      ion-col {
        height: 100%;
      }
    }
  }
```

review-user-progression.page.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: Typescript file for the review user progression page.
 */
import { Component, OnInit, ViewChild, ElementRef, OnDestroy } from '@angu
lar/core';
import { Chart } from 'chart.js';
import { ActivatedRoute } from '@angular/router';
import { NavController } from '@ionic/angular';
import { ResultsService } from 'src/app/services/results.service';
import { Subscription } from 'rxjs';
import { ReviewDetailService } from 'src/app/services/review-
detail.service';
import { Soc } from 'src/app/models/soc.model';
import { AuthService } from 'src/app/services/auth.service';
import { UserData } from 'src/app/models/userData.model';

@Component({
  selector: 'app-review-user-progression',
  templateUrl: './review-user-progression.page.html',
  styleUrls: ['./review-user-progression.page.scss'],
})
export class ReviewUserProgressionPage implements OnInit, OnDestroy {
  @ViewChild('lineCanvas', {static: false}) lineCanvas: ElementRef;

  private lineChart: Chart;
  reviewDetailSub: Subscription;
  userSub: Subscription;
  socs: Soc[];
  user: UserData;
  dataset: object[] = [];
  newData: object[] = [];
  colors: string[] = ['#3e95cd', '#8e5ea2', '#3cba9f', '#e8c3b9', '#c45850
', '#f57f17',
                      '#ff1744', '#d500f9', '#2979ff', '#00c853', '#bf360c
', '#5d4037',
                      '#546e7a', '#1a237e', '#006064', '#33691e', '#e65100
', '#ffd600',
                      ]; // ADD MORE COLORS
  isLoading = false;

  constructor(
    public route: ActivatedRoute,
    private navCtrl: NavController,
    public resultService: ResultsService,
    public reviewDetailService: ReviewDetailService,
    public authService: AuthService
  ) { }

  ngOnInit() {
    this.route.paramMap.subscribe(paramMap => {
      if (!paramMap.get('userId')) {
        this.navCtrl.navigateBack('/review-progression');
        return;
      }
      this.isLoading = true;
      this.userSub = this.authService
        .getUser(paramMap.get('userId'))
        .subscribe(user => {
          this.user = user;
          this.isLoading = false;
        });
      this.reviewDetailSub = this.reviewDetailService.getSocs(paramMap.get
('userId')).subscribe(socs => {
        console.log(socs);
        this.setChartData(paramMap.get('userId'), socs);
      });
    });
  }

  setChartData(userId: string, socs: Soc[]) {
    this.resultService.getResultObject(userId).subscribe(object => {
      const test = object;
      let index = 0;
      for (const soc in test) {
        if (test.hasOwnProperty(soc)) {
          this.newData = [];
          for (const result in test[soc]) {
            if (test[soc].hasOwnProperty(result)) {
              this.newData.push(
                {
                  x: test[soc][result].date,
                  y: test[soc][result].result / test[soc][result].total *
100

                }
              );
            }
          }
          this.dataset.push(
            {
              data: this.newData,
              label: socs[index].name,
              borderColor: this.colors[index],
              fill: false
            }
          );
          index++;
        }
      }
      this.setChart();
    });
  }

  setChart() {
    this.lineChart = new Chart(this.lineCanvas.nativeElement, {
      type: 'line',
      data: {
        datasets: this.dataset
      },
      options: {
          scales: {
              xAxes: [{
                  type: 'time',
                  time: {
                      unit: 'month',
                      displayFormats: {
                        day: 'MMM YY'
                      }
                  }
              }]
          },
          title: {
            display: true,
            text: this.user.fname + '\'s Progression'
          },
          elements: {
            line: {
                tension: 0
            }
          },
          maintainAspectRatio: false,
          responsive: true,
      }
    });
  }
  ngOnDestroy() {
    if (this.userSub) {
      this.userSub.unsubscribe();
    }
    if (this.reviewDetailSub) {
      this.reviewDetailSub.unsubscribe();
    }
  }
}
```

Search

search.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-menu-button menuId="navId"></ion-menu-button>
    </ion-buttons>
    <ion-title>Review Progression</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-
searchbar showCancelButton="focus" (ionInput)="filter($event)"></ion-
searchbar>
  <ion-list>
    <ion-item
      *ngFor = "let item of listUsers"
      [routerLink]="['/', 'review-progression', item.id]"
      detail
    >
      <ion-label>{{ item.fname }} {{ item.lname }}</ion-label>
    </ion-item>
  </ion-list>
</ion-content>
```

search.page.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: Typescript file for the search page.
 */
import { Component, OnInit, OnDestroy } from '@angular/core';
import { UserData } from 'src/app/models/userData.model';
import { Subscription } from 'rxjs';
import { AuthService } from 'src/app/services/auth.service';

@Component({
  selector: 'app-search',
  templateUrl: './search.page.html',
  styleUrls: ['./search.page.scss'],
})
export class SearchPage implements OnInit, OnDestroy {
  loadedUsers: UserData[];
  listUsers: UserData[];
  fullName: string;
  private usersSub: Subscription;
  isLoading = false;
  isItemAvailable = false;

  constructor(
    private authService: AuthService
  ) { }

  ngOnInit() {
    this.usersSub = this.authService.users.subscribe(users => {
      this.listUsers = users;
      this.loadedUsers = users;
    });
  }

  ionViewWillEnter() {
    this.isLoading = true;
    this.authService.fetchUsers().subscribe(() => {
      this.isLoading = false;
    });
  }

  initializeItems() {
    this.listUsers = this.loadedUsers;
  }

  filter(event: any) {
    this.initializeItems();
    this.fullName = '';
    const val = event.target.value;
    console.log(val);
    if (val && val.trim() !== '') {
      this.isItemAvailable = true;
      this.listUsers = this.listUsers.filter((item) => {
        this.fullName = item.fname + ' ' + item.lname;
        return (this.fullName.toLowerCase().indexOf(val.toLowerCase()) > -
1);
      });
    }
  }
  ngOnDestroy() {
    if (this.usersSub) {
      this.usersSub.unsubscribe();
    }
  }

}
```

Review SOC

review-soc-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { ReviewSocPage } from './review-soc.page';

const routes: Routes = [
  {
    path: '',
    redirectTo: 'search',
    pathMatch: 'full'
  },
  {
    path: 'search',
    loadChildren: () => import('./search/search.module').then( m => m.Sear
chPageModule)
  },
  {
    path: ':userId',
    loadChildren: () => import('./view-user-result/view-user-
result.module').then( m => m.ViewUserResultPageModule)
  },
  {
    path: ':userId/:socId',
    loadChildren: () => import('./view-soc-result/view-soc-
result.module').then( m => m.ViewSocResultPageModule)
  },
  {
    path: ':userId/:socId/:resultId',
    loadChildren: () => import('./view-soc-result-detail/view-soc-result-
detail.module').then( m => m.ViewSocResultDetailPageModule)
  }
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule],
})
export class ReviewSocPageRoutingModule {}
```

Search

search.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-menu-button menuId="navId"></ion-menu-button>
    </ion-buttons>
    <ion-title>Review SOC</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-
searchbar showCancelButton="focus" (ionInput)="filter($event)"></ion-
searchbar>
  <ion-list>
    <ion-item
      *ngFor = "let item of listUsers"
      [routerLink]="['/', 'review-soc', item.id]"
      detail
    >
      <ion-label>{{ item.fname }} {{ item.lname }}</ion-label>
    </ion-item>
  </ion-list>
</ion-content>
```

search.page.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: Typescript file for the search page.
 */
import { Component, OnInit, OnDestroy } from '@angular/core';
import { UserData } from 'src/app/models/userData.model';
import { Subscription } from 'rxjs';
import { AuthService } from 'src/app/services/auth.service';

@Component({
  selector: 'app-search',
  templateUrl: './search.page.html',
  styleUrls: ['./search.page.scss'],
})
export class SearchPage implements OnInit, OnDestroy {
  loadedUsers: UserData[];
  listUsers: UserData[];
  fullName: string;
  private usersSub: Subscription;
  isLoading = false;
  isItemAvailable = false;

  constructor(
    private authService: AuthService
  ) { }

  ngOnInit() {
    this.usersSub = this.authService.users.subscribe(users => {
      this.listUsers = users;
      this.loadedUsers = users;
    });
  }

  ionViewWillEnter() {
    this.isLoading = true;
    this.authService.fetchUsers().subscribe(() => {
      this.isLoading = false;
    });
  }

  initializeItems() {
    this.listUsers = this.loadedUsers;
  }

  filter(event: any) {
    this.initializeItems();
    this.fullName = '';
    const val = event.target.value;
    console.log(val);
    if (val && val.trim() !== '') {
      this.isItemAvailable = true;
      this.listUsers = this.listUsers.filter((item) => {
        this.fullName = item.fname + ' ' + item.lname;
        return (this.fullName.toLowerCase().indexOf(val.toLowerCase()) > -
1);
      });
    }
  }
  ngOnDestroy() {
    if (this.usersSub) {
      this.usersSub.unsubscribe();
    }
  }

}
```

View SOC Result

view-soc-result.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-back-button defaultHref="review-soc"></ion-back-button>
    </ion-buttons>
    <ion-title>Results</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <div *ngIf="isLoading" class="ion-text-center">
    <ion-spinner color="primary"></ion-spinner>
  </div>
  <ion-
grid *ngIf="!isLoading && !isLoadingResult && !isLoadingSoc && results.len
gth > 0">
    <ion-row class="border">
      <ion-col class="heading">
        Date
      </ion-col>
      <ion-col class="heading">
        Wrong Answers
      </ion-col>
      <ion-col class="heading">
        Result(%)
      </ion-col>
    </ion-row>
    <ion-row
      class="borderLight"
      *ngFor="let result of results"
      [routerLink]="['/', 'review-soc', userId, soc.id, result.id]"
    >
      <ion-col>
        {{ result.date | date: 'longDate' }}
      </ion-col>
      <ion-col>
        {{ result.total - result.result }}
      </ion-col>
      <ion-col>
        {{ result.result / result.total * 100  | number: '1.0-0'}}%
      </ion-col>
    </ion-row>
  </ion-grid>
</ion-content>
```

view-soc-result.page.scss

```
.heading {
    font-weight: bold;
}

.border {
    border-bottom: solid 1px #000000;
    padding: 5px;
}

.borderLight {
    border-bottom: solid 1px #bbbbbb;
    padding: 5px;
}
```

view-soc-result.page.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: Typescript file for the view soc result page.
 */
import { Component, OnInit, OnDestroy } from '@angular/core';
import { Soc } from 'src/app/models/soc.model';
import { Subscription } from 'rxjs';
import { ActivatedRoute } from '@angular/router';
import { NavController } from '@ionic/angular';
import { Result } from 'src/app/models/result.model';
import { SocsService } from 'src/app/services/socs.service';
import { ResultsService } from 'src/app/services/results.service';

@Component({
  selector: 'app-view-soc-result',
  templateUrl: './view-soc-result.page.html',
  styleUrls: ['./view-soc-result.page.scss'],
})
export class ViewSocResultPage implements OnInit, OnDestroy {
  isLoading = false;
  isLoadingResult = false;
  isLoadingSoc = false;
  soc: Soc;
```

```
results: Result[];
percents: number[];
socId: string;
userId: string;
private socSub: Subscription;
private resultsSub: Subscription;

constructor(
  private socService: SocsService,
  private resultsService: ResultsService,
  private navCtrl: NavController,
  private route: ActivatedRoute
) { }

ngOnInit() {
  this.isLoadingResult = true;
  this.isLoadingSoc = true;
  this.route.paramMap.subscribe(paramMap => {
    if (!paramMap.get('socId')) {
      this.navCtrl.navigateBack('/review-soc');
      return;
    }
    if (!paramMap.get('userId')) {
      this.navCtrl.navigateBack('/review-soc');
      return;
    }
    this.socId = paramMap.get('socId');
    this.userId = paramMap.get('userId');
    this.socSub = this.socService.getSoc(this.socId).subscribe(soc => {
      this.soc = soc;
      this.isLoadingSoc = false;
    });
    this.resultsSub = this.resultsService
      .results
      .subscribe(results => {
        this.results = results;
        this.isLoadingResult = false;
      });
  });

}

ionViewWillEnter() {
  this.isLoading = true;
```

```
    this.resultsService
      .fetchResults(this.userId, this.socId)
      .subscribe(() => {
        this.isLoading = false;
    });
  }

  ngOnDestroy() {
    if (this.resultsSub) {
      this.resultsSub.unsubscribe();
    }
    if (this.socSub) {
      this.socSub.unsubscribe();
    }
  }

}
```

View SOC Result Detail

view-soc-result-detail.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-back-button defaultHref="review-soc"></ion-back-button>
    </ion-buttons>
    <ion-title>Review Result</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <div *ngIf="isLoading" class="ion-text-center">
    <ion-spinner color="primary"></ion-spinner>
  </div>
  <ion-grid>
    <ion-row>
      <ion-col></ion-col>
      <ion-col><canvas #doughnutCanvas></canvas></ion-col>
      <ion-col></ion-col>
    </ion-row>
  </ion-grid>
  <ion-grid *ngIf="!isLoading">
    <ion-row>
      <ion-col size-sm="6" offset-sm="3">
        {{ result.date | date: 'longDate'}}
      </ion-col>
    </ion-row>
    <ion-row *ngFor="let question of socQuestions">
      <ion-col size-sm="6" offset-sm="3">
        {{ question.name }}
      </ion-col>
    </ion-row>
    <ion-row *ngFor="let item of feedback">
      <ion-col size-sm="6" offset-sm="3">
        <ion-card>
          <ion-card-header>
            <ion-card-title>{{ item.senderName }}</ion-card-title>
          </ion-card-header>
          <ion-card-content>
            <time>{{ item.date | date:'medium' }}</time>
            <p>{{ item.feedback}}</p>
          </ion-card-content>
        </ion-card>
        <!-- <div>
          <div>
            <h4>{{ item.userId }}</h4>
            <time>{{ item.date | date:'medium' }}</time>
            <p>{{ item.feedback}}</p>
          </div>
        </div>  -->
      </ion-col>
    </ion-row>
  </ion-grid>
</ion-content>
<ion-footer>
  <ion-grid>
    <ion-row *ngIf="!feedbackLoading">
      <ion-col size-sm="6" offset-sm="3">
        <ion-
button color="primary" (click)="giveFeedback()" size="full">Give Feedback<
/ion-button>
      </ion-col>
    </ion-row>
    <ion-row *ngIf="feedbackLoading">
      <ion-col size-sm="6" offset-sm="3">
        <form [formGroup]="form">
          <ion-item>
            <ion-label position="floating">Feedback</ion-label>
            <ion-textarea
              autocomplete
              autocorrect
              rows="3"
              formControlName="feedback"
            ></ion-textarea>
          </ion-item>
        </form>
      </ion-col>
    </ion-row>
    <ion-row *ngIf="feedbackLoading">
      <ion-col size-sm="3" offset-sm="3">
        <ion-
button color="primary" (click)="submitFeedback()" [disabled]="!form.valid"
 size="full">Submit Feedback</ion-button>
      </ion-col>
      <ion-col size-sm="3">
```

```
        <ion-
button color="danger" (click)="submitFeedback()" size="full">Cancel</ion-
button>
      </ion-col>
    </ion-row>
  </ion-grid>
</ion-footer>
```

view-soc-result-detail.page.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: Typescript file for the view soc result detail page.
 */
import { Component, OnInit, OnDestroy, ViewChild, ElementRef } from '@angu
lar/core';
import { Result } from 'src/app/models/result.model';
import { Soc } from 'src/app/models/soc.model';
import { SocQuestion } from 'src/app/models/soc-question.model';
import { Subscription } from 'rxjs';
import { ResultsService } from 'src/app/services/results.service';
import { ReviewDetailService } from 'src/app/services/review-
detail.service';
import { SocQuestionService } from 'src/app/services/soc-
question.service';
import { ActivatedRoute } from '@angular/router';
import { NavController, LoadingController, AlertController } from '@ionic/
angular';
import { Chart } from 'chart.js';
import { FormGroup, FormBuilder, FormControl, Validators } from '@angular/
forms';
import { AuthService } from 'src/app/services/auth.service';
import { Feedback } from 'src/app/models/feedback.model';

@Component({
  selector: 'app-view-soc-result-detail',
  templateUrl: './view-soc-result-detail.page.html',
  styleUrls: ['./view-soc-result-detail.page.scss'],
})
export class ViewSocResultDetailPage implements OnInit, OnDestroy {
  @ViewChild('doughnutCanvas', {static: false}) doughnutCanvas: ElementRef
;
```

```
private doughnutChart: Chart;
result: Result;
soc: Soc;
socQuestions: SocQuestion[];
socId: string;
userId: string;
resultId: string;
isLoading = false;
private resultSub: Subscription;
private reviewDetailSub: Subscription;
feedbackLoading = false;
form: FormGroup;
senderName: string;
feedback: Feedback[];

constructor(
  public resultService: ResultsService,
  public reviewDetailService: ReviewDetailService,
  public socQuestionService: SocQuestionService,
  public route: ActivatedRoute,
  private navCtrl: NavController,
  private fb: FormBuilder,
  private loadingCtrl: LoadingController,
  private authService: AuthService,
) { }

ngOnInit() {
  this.authService.currUser.subscribe(user => {
    this.senderName = user.fname + ' ' + user.lname;
  });
  this.form = this.fb.group({
    feedback: new FormControl(null, {
      updateOn: 'blur',
      validators: [Validators.required]
    }),
  });
  this.isLoading = true;
  this.route.paramMap.subscribe(paramMap => {
    if (!paramMap.get('socId')) {
      this.navCtrl.navigateBack('/socs/review');
      return;
    }
    if (!paramMap.get('userId')) {
      this.navCtrl.navigateBack('/socs/review');
```

```
        return;
      }
      if (!paramMap.get('resultId')) {
        this.navCtrl.navigateBack('/socs/review');
        return;
      }
      this.resultId = paramMap.get('resultId');
      this.socId = paramMap.get('socId');
      this.userId = paramMap.get('userId');
      this.getResults();
    });
  }

  getResults() {
    this.resultSub = this.resultService.getResult(this.resultId, this.socI
d, this.userId).subscribe(result => {
      this.result = result;
      console.log(this.result.feedback);
      const feedback = [];
      for (const key in this.result.feedback) {
        if (this.result.feedback.hasOwnProperty(key)) {
          feedback.push(new Feedback(
            key,
            this.result.feedback[key].feedback,
            this.result.feedback[key].senderName,
            this.result.feedback[key].date,
          ));
        }
      }
      this.feedback = feedback.sort((a, b) => {
        return b.date.localeCompare(a.date) || 0;
      });
      this.setDoughnut();
      this.getQuestions();
      this.isLoading = false;
    });
  }
  getQuestions() {
    if (this.result.incorrect !== undefined) {
      this.reviewDetailSub = this.reviewDetailService.getQuestions(this.re
sult.incorrect, this.socId).subscribe(questions => {
        this.socQuestions = questions;
      });
    }
  }
  setDoughnut() {
    this.doughnutChart = new Chart(this.doughnutCanvas.nativeElement, {
      type: 'doughnut',
      data: {
        labels: ['Correct', 'Incorrect'],
        datasets: [
          {
            data: [this.result.result, this.result.total - this.result.res
ult],
            backgroundColor: [
              '#00ff00',
              '#ff0000',
            ],
          }
        ]
      },
      options: {
        rotation: 1 * Math.PI,
        circumference: 1 * Math.PI,
      }
    });
  }

  giveFeedback() {
    this.feedbackLoading = true;
  }

  submitFeedback() {
    if (!this.form.valid) {
      return;
    }
    this.loadingCtrl.create({
      message: 'Submitting Feedback...'
    }).then(loadingEl => {
      loadingEl.present();
      this.resultService.addFeedback(
        this.form.value.feedback,
        this.senderName,
        this.userId,
        this.socId,
        this.resultId,
      ).subscribe(() => {
        loadingEl.dismiss();
```

```
          this.form.reset();
          this.getResults();
          this.feedbackLoading = false;
        });
      });
    }

    ngOnDestroy() {
      if (this.resultSub) {
        this.resultSub.unsubscribe();
      }
      if (this.reviewDetailSub) {
        this.reviewDetailSub.unsubscribe();
      }
    }
  }
```

View User Result

view-user-result.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-back-button defaultHref="review-soc"></ion-back-button>
    </ion-buttons>
    <ion-title>{{ isLoading ? 'Loading...' : user.fname }}'s SOCs</ion-
title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-
searchbar showCancelButton="focus" (ionInput)="filter($event)"></ion-
searchbar>
  <div *ngIf="isLoading" class="ion-text-center">
    <ion-spinner color="primary"></ion-spinner>
  </div>
  <p *ngIf="!isLoading && !isLoadingSoc && loadedSocs.length <= 0" class="
ion-text-center">
    User has no SOC results.
  </p>
  <p *ngIf="!isLoading && loadedSocs.length > 0 && listSocs.length <= 0" c
lass="ion-text-center">
    SOC not found.
  </p>
  <ion-list *ngIf="!isLoading && !isLoadingSoc && listSocs.length > 0">
    <ion-item
      *ngFor="let soc of listSocs"
      [routerLink]="['/', 'review-soc', userId, soc.id]"
      detail
    >
    {{ soc.name }}
    </ion-item>
  </ion-list>
</ion-content>
```

view-user-result.page.ts

```
/**
 * Name:        Wiliam Nolan
 * Student ID:  C00216986
 * Description: Typescript file for the view user result page.
 */
import { Component, OnInit, OnDestroy } from '@angular/core';
import { UserData } from 'src/app/models/userData.model';
import { Soc } from 'src/app/models/soc.model';
import { Subscription } from 'rxjs';
import { ActivatedRoute, Router } from '@angular/router';
import { NavController, AlertController } from '@ionic/angular';
import { AuthService } from 'src/app/services/auth.service';
import { ReviewDetailService } from 'src/app/services/review-
detail.service';

@Component({
  selector: 'app-view-user-result',
  templateUrl: './view-user-result.page.html',
  styleUrls: ['./view-user-result.page.scss'],
})
export class ViewUserResultPage implements OnInit, OnDestroy {
  user: UserData;
  loadedSocs: Soc[];
  listSocs: Soc[];
  userId: string;
  private userSub: Subscription;
  private reviewDetailSub: Subscription;
  isLoading = false;
  isLoadingSoc = false;
  isItemAvailable = false;

  constructor(
    private route: ActivatedRoute,
    private navCtrl: NavController,
    private authService: AuthService,
    private alertCtrl: AlertController,
    private router: Router,
    private reviewDetailService: ReviewDetailService,
  ) { }

  ngOnInit() {
    this.isLoadingSoc = true;
    this.reviewDetailSub = this.reviewDetailService.socs.subscribe(socs =>
 {
      this.loadedSocs = socs;
      this.listSocs = socs;
      this.isLoadingSoc = false;
    });
    this.route.paramMap.subscribe(paramMap => {
      if (!paramMap.has('userId')) {
        this.navCtrl.navigateBack('/socs/review');
        return;
      }
      this.userId = paramMap.get('userId');
      this.isLoading = true;
      this.userSub = this.authService
        .getUser(paramMap.get('userId'))
        .subscribe(user => {
          this.user = user;
          this.isLoading = false;
        }, error => {
          this.alertCtrl.create({
            header: 'An error occured',
            message: 'Could not load User',
            buttons: [
              {
                text: 'Okay',
                handler: () => {
                  this.router.navigate(['socs/review']);
                }
              }
            ]
          }).then(alertEl => alertEl.present());
        });
    });
  }

  ionViewWillEnter() {
    this.isLoading = true;
    this.route.paramMap.subscribe(paramMap => {
      if (!paramMap.has('userId')) {
        this.navCtrl.navigateBack('/socs/review');
        return;
      }
      this.reviewDetailService.getSocs(paramMap.get('userId')).subscribe((
) => {
        this.isLoading = false;
      });
    });
  }
  ngOnDestroy() {
    if (this.userSub) {
      this.userSub.unsubscribe();
    }
    if (this.reviewDetailSub) {
      this.reviewDetailSub.unsubscribe();
    }
  }

  initializeItems() {
    this.listSocs = this.loadedSocs;
  }

  filter(event: any) {
    this.initializeItems();
    const val = event.target.value;
    if (val && val.trim() !== '') {
      this.isItemAvailable = true;
      this.listSocs = this.listSocs.filter((item) => {
        return (item.name.toLowerCase().indexOf(val.toLowerCase()) > -1);
      });
    }
  }
}
```

Take SOC

take-sco-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { TakeSocPage } from './take-soc.page';

const routes: Routes = [
  {
    path: '',
    redirectTo: '/view-soc/tabs/todo',
    pathMatch: 'full'
  },
  {
    path: 'start-soc/:socId',
    loadChildren: () => import('./start-soc/start-
soc.module').then( m => m.StartSocPageModule)
  },
  {
    path: 'soc-result/:socId',
    loadChildren: () => import('./soc-result/soc-
result.module').then( m => m.SocResultPageModule)
  },
  {
    path: ':socId/:questionId',
    loadChildren: () => import('./soc-question/soc-
question.module').then( m => m.SocQuestionPageModule)
  },
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule],
})
export class TakeSocPageRoutingModule {}
```

SOC Question

soc-question.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-title>{{ isLoading ? 'Loading..' : soc.name }}</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content class="ion-padding">
  <div *ngIf="isLoadingQuestion" class="ion-text-center">
    <ion-spinner color="primary"></ion-spinner>
  </div>
  <ion-grid *ngIf="!isLoadingProgress" class="ion-text-center">
    <progress id="progressBar" max="{{ questions.length }}" value="{{ prog
ress }}"></progress>
  </ion-grid>
  <ion-grid *ngIf="!isLoadingQuestion" class="ion-text-center">
    <h2>{{ question.name }}</h2>
  </ion-grid>
  <ion-grid *ngIf="!isLoadingAnswer" class="ion-text-center">
    <section class="fullWidth" *ngFor="let answer of answers">
      <ion-button
        #button
        (click)="runAll(question.id, question.name, answer.isAnswer, answe
rs.indexOf(answer))"
        expand="block"
        fill="clear"
        [style.background-
color]="toggle[answers.indexOf(answer)] ? '#55acee' : correct ? '#2ecc71'
: '#e74c3c'"
        [style.box-
shadow]="toggle[answers.indexOf(answer)] ? '0px 5px 0px 0px #3C93D5' : cor
rect ? '0px 5px 0px 0px #27ae60' : '0px 5px 0px 0px #c0392b'"
        size="large"
        color="light"
        class="ion-text-wrap tallerButton"
      >
        {{ answer.name }}
      </ion-button>
    </section>
  </ion-grid>
</ion-content>
<ion-footer no-border class="correct" *ngIf="answered && correct">
  <ion-grid class="ion-padding">
```

```
    <h3 class="correctText">You are correct</h3>
    <ion-
button [routerLink]="[url]" (click)="reset()" expand="block" fill="clear"
class="correctButton">Continue</ion-button>
  </ion-grid>
</ion-footer>
<ion-footer no-border class="incorrect" *ngIf="answered && !correct">
  <ion-grid class="ion-padding">
    <h3 class="incorrectText">You are incorrect</h3>
    <h5 class="incorrectAnswer">The correct answer is: <br>{{ correctAnswe
r }}</h5>
    <ion-
button [routerLink]="[url]" (click)="reset()" expand="block" fill="clear"
class="incorrectButton">Continue</ion-button>
  </ion-grid>
</ion-footer>
```

soc-question.page.scss

```
.incorrect {
    background-color: #ffabab;
}

.incorrectText {
    color: #c0392b;
    font-weight: bold;
}

.incorrectAnswer{
    color: #c0392b;
}

.incorrectButton {
    background-color: #e74c3c;
    box-shadow: 0px 5px 0px 0px #c0392b;
    float: center;
}

.correct {
    background-color: #c0ffad;
}

.correctText {
    color: #27ae60;
    font-weight: bold;
}

.correctButton {
    background-color: #2ecc71;
    box-shadow: 0px 5px 0px 0px #27ae60;
    float: center;
}

section:not(.full-width),
  .full-width > header {
    padding: 0 10px;
}

ion-button,
ion-button:disabled,
ion-button[disabled]{
    color: #fff;
    margin: 20px;
    font-weight: bold;
}

progress {
    display:inline-block;
    width: 100%;
    height: 10px;
    border-radius: 5px;
}
progress::-webkit-progress-bar {
    background-color: #b9b9b9;
    border-radius: 5px;
}
progress::-webkit-progress-value {
    background: #2ecc71;
    border-radius: 5px;
}

.tallerButton {
    height: 75px;
}
```

soc-question.page.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: Typescript file for the soc question page.
 */
import { Component, OnInit, OnDestroy } from '@angular/core';
import { Soc } from 'src/app/models/soc.model';
import { SocQuestion } from 'src/app/models/soc-question.model';
import { SocAnswer } from 'src/app/models/soc-answer.model';
import { Subscription } from 'rxjs';
import { Router, ActivatedRoute } from '@angular/router';
import { NavController, AlertController } from '@ionic/angular';
import { SocsService } from 'src/app/services/socs.service';
import { SocQuestionService } from 'src/app/services/soc-
question.service';
import { SocAnswerService } from 'src/app/services/soc-answer.service';
import { QuestionService } from 'src/app/services/question.service';
import { LeaderboardService } from 'src/app/services/leaderboard.service';
import { AuthService } from 'src/app/services/auth.service';

@Component({
  selector: 'app-soc-question',
  templateUrl: './soc-question.page.html',
  styleUrls: ['./soc-question.page.scss'],
})
export class SocQuestionPage implements OnInit, OnDestroy {
  disabled = false;
  toggle: boolean[] = [];
  soc: Soc;
  question: SocQuestion;
  questions: SocQuestion[];
  answers: SocAnswer[];
  url: string;
  nextIndex: number;
  score: number;
  isLoading = false;
  isLoadingQuestion = false;
  isLoadingAnswer = false;
  isLoadingProgress = false;
  private socSub: Subscription;
  private socQuestionsSub: Subscription;
  private socAnswerSub: Subscription;
  answered: boolean;
  correct: boolean;
  correctAnswer: string;
  progress: number;
  now: Date;
  timeAnswered: Date;
```

```
constructor(
  private router: Router,
  private route: ActivatedRoute,
  private navCtrl: NavController,
  private socsService: SocsService,
  private socQuestionsService: SocQuestionService,
  private socAnswersService: SocAnswerService,
  private questionService: QuestionService,
  private leaderService: LeaderboardService,
  private authService: AuthService,
  private alertCtrl: AlertController,
) { }

ngOnInit() {
  this.route.paramMap.subscribe(paramMap => {
    if (!paramMap.has('socId')) {
      this.navCtrl.navigateBack('view-soc');
      return;
    }
    if (!paramMap.has('questionId')) {
      this.navCtrl.navigateBack('view-soc');
      return;
    }
    this.isLoading = true;
    this.socSub = this.socsService
      .getSoc(paramMap.get('socId'))
      .subscribe(soc => {
        this.soc = soc;
        this.progress = this.questionService.getProgress();
        this.isLoading = false;
      }, error => {
        this.alertCtrl.create({
          header: 'An error occured',
          message: 'Could not load SOC',
          buttons: [
            {
              text: 'Okay',
              handler: () => {
                this.router.navigate(['view-soc']);
              }
            }
          ]
        }).then(alertEl => alertEl.present());
      });
    this.isLoadingQuestion = true;
    this.socQuestionsSub = this.socQuestionsService
      .getQuestion(paramMap.get('socId'), paramMap.get('questionId'))
      .subscribe(socQuestion => {
        this.question = socQuestion;
        this.isLoadingQuestion = false;
      }, error => {
        this.alertCtrl.create({
          header: 'An error occured',
          message: 'Could not load question',
          buttons: [
            {
              text: 'Okay',
              handler: () => {
                this.router.navigate(['view-soc']);
              }
            }
          ]
        }).then(alertEl => alertEl.present());
      });
    this.isLoadingProgress = true;
    this.socQuestionsSub = this.socQuestionsService
      .fetchQuestions(paramMap.get('socId'))
      .subscribe(socQuestions => {
        this.questions = socQuestions;
        this.isLoadingProgress = false;
      });
    this.isLoadingAnswer = true;
    this.socAnswerSub = this.socAnswersService
      .fetchAnswers(paramMap.get('socId'), paramMap.get('questionId'))
      .subscribe(socAnswers => {
        this.answers = this.shuffle(socAnswers);
        // tslint:disable-next-line: prefer-for-of
        for (let i = 0; i < this.answers.length; i++) {
          this.toggle.push(true);
        }
        this.correctAnswer = this.answers[this.answers.findIndex(x => x.
isAnswer === true)].name;
        this.isLoadingAnswer = false;
      });
    this.now = new Date();
    this.now.setSeconds(this.now.getSeconds() + 5);
  });
  }

  runAll(questionID: string, questionName: string, answer: boolean, index:
 number) {
    if (!this.disabled) {
      this.selectionMade(index);
      this.checkAnswer(questionID, questionName, answer);
    }
  }

  selectionMade(index: number) {
    this.toggle[index] = !this.toggle[index];
    this.disabled = true;
  }

  reset() {
    this.answered = false;
    this.correct = null;
    this.disabled = false;
    this.toggle = [];
    // tslint:disable-next-line: prefer-for-of
    for (let i = 0; i < this.answers.length; i++) {
      this.toggle.push(true);
    }
  }

  checkAnswer(questionID: string, questionName: string, answer: boolean) {
    this.timeAnswered = new Date();
    this.nextIndex = this.questions.findIndex(x => x.id === this.question.
id) + 1;
    if (!answer) {
      this.correct = false;
      this.questionService.addIncorrectQuestion(questionID, questionName);
      if (this.nextIndex
        === this.questions.length) {
          this.questionService.firstRunDone();
        }
    } else {
      this.questionService.addProgress();
      this.progress = this.questionService.getProgress();
      this.correct = true;
      if (this.questionService.isFirstRun()) {
        if (this.nextIndex
          === this.questions.length) {
            this.questionService.firstRunDone();
          }
        this.questionService.addResult();
        this.questionService.addScore((this.now.getTime() - this.timeAnswe
red.getTime()) / 10);
      }
    }
    if (this.questionService.isFirstRun()) {
      this.url =
        '/take-soc/'
        + this.soc.id
        + '/'
        + this.questions[this.nextIndex].id;
    } else {
      if (this.questionService.getIncorrectQuestions().length === 0) {
        this.authService.currUser.subscribe(user => {
          this.leaderService.compareScores(this.soc.id, user.fname + user.
lname);
        });
        this.url =
          '/take-soc/soc-result/' +
          this.soc.id;
      } else {
        this.url =
          '/take-soc/'
          + this.soc.id
          + '/'
          + this.questionService.getIncorrectQuestions()[0];
        this.questionService.removeIncorrectQuestion();
      }
    }
    this.answered = true;
    this.score = this.questionService.getScore();
  }

  shuffle(array: SocAnswer[]) {
    let m = array.length, t, i;
    while (m) {
      i = Math.floor(Math.random() * m--);
      t = array[m];
      array[m] = array[i];
      array[i] = t;
    }
    return array;
  }

  ngOnDestroy() {
    if (this.socSub) {
      this.socSub.unsubscribe();
    }
    if (this.socQuestionsSub) {
      this.socQuestionsSub.unsubscribe();
    }
    if (this.socAnswerSub) {
      this.socAnswerSub.unsubscribe();
    }
  }

}
```

SOC Result

soc-result.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-menu-button menuId="navId"></ion-menu-button>
    </ion-buttons>
    <ion-title>Results</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-grid>
    <ion-row>
      <ion-col>
        <h1>Here are your results:</h1>
      </ion-col>
    </ion-row>
    <ion-row>
      <ion-col>
        {{ result }} out of {{ totalQuestions }}
      </ion-col>
    </ion-row>
  </ion-grid>
  <ion-grid *ngIf="incorrect.length > 0">
    <ion-row>
      <ion-col>
        <h3>You got these questions wrong:</h3>
      </ion-col>
    </ion-row>
    <ion-row>
      <ion-col>
        <ion-virtual-scroll
          [items]="incorrect"
          approxItemHeight="60px"
        >
          <ion-item
            *virtualItem="let socQuestion"
          >
            <ion-label>
              <h2>{{ socQuestion }}</h2>
            </ion-label>
          </ion-item>
        </ion-virtual-scroll>
      </ion-col>
    </ion-row>
  </ion-grid>
</ion-content>
```

soc-result.page.ts

```
/**
 * Name:        Wiliam Nolan
 * Student ID:  C00216986
 * Description: Typescript file for the soc result page.
 */
import { Component, OnInit, OnDestroy } from '@angular/core';
import { Subscription } from 'rxjs';
import { QuestionService } from 'src/app/services/question.service';
import { SocQuestionService } from 'src/app/services/soc-
question.service';
import { ResultsService } from 'src/app/services/results.service';
import { AuthService } from 'src/app/services/auth.service';
import { ActivatedRoute } from '@angular/router';
import { NavController } from '@ionic/angular';
import { LeaderboardService } from 'src/app/services/leaderboard.service';

@Component({
  selector: 'app-soc-result',
  templateUrl: './soc-result.page.html',
  styleUrls: ['./soc-result.page.scss'],
})
export class SocResultPage implements OnInit, OnDestroy {
  score: number;
  result: number;
  incorrect: string[];
  incorrectIds: string[];
  totalQuestions: number;
  socQuestionSub: Subscription;
  userId: string;
  socId: string;
  name: string;

  constructor(
    private questionService: QuestionService,
    private socQuestionService: SocQuestionService,
    private resultsService: ResultsService,
    private leaderService: LeaderboardService,
    private authService: AuthService,
    private route: ActivatedRoute,
    private navCtrl: NavController,
  ) { }

  ngOnInit() {
    this.route.paramMap.subscribe(paramMap => {
      if (!paramMap.has('socId')) {
        this.navCtrl.navigateBack('view-soc');
        return;
      }
      this.socId = paramMap.get('socId');
      this.result = this.questionService.getResult();
      this.incorrect = this.questionService.getFinalIncorrectQuestionNames
();
      this.incorrectIds = this.questionService.getFinalIncorrectQuestionID
s();
      this.score = this.questionService.getScore();
    });
    this.socQuestionSub = this.socQuestionService.socQuestions.subscribe(q
uestions => {
      this.totalQuestions = questions.length;
    });
    this.authService.userId.subscribe(id => {
      this.userId = id;
    });
    this.resultsService.addResult(
      this.userId,
      this.socId,
      this.result,
      this.totalQuestions,
      this.incorrectIds,
    ).subscribe();
    this.leaderService.addLeaderboard(
      this.socId,
      this.score,
    ).subscribe();
  }

  ngOnDestroy() {
    if (this.socQuestionSub) {
      this.socQuestionSub.unsubscribe();
    }
  }
```

```
}
```

Start SOC

start-soc.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-back-button defaultHref="view-soc"></ion-back-button>
    </ion-buttons>
    <ion-title>{{ isLoading ? 'Loading..' : soc.name }}</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <div *ngIf="!isLoading && !isLoadingQuestions">
    <!-- [routerLink]="['/take-soc', soc.id, questions[0].id]" -->
    <ion-button *ngIf="!started" size="large" (click)="startSoc()">
      Start
    </ion-button>
    <h1 *ngIf="started">{{ countdown }}</h1>
   </div>

</ion-content>
```

start-soc.page.scss

```
ion-content {
    div {
      height: 100%;
      width: 100%;
      display: flex;
      justify-content: center;
      align-items: center;
    }
    h1 {
        color: #111;
        font-size: 275px;
        font-weight: bold;
        letter-spacing: -1px;
        line-height: 1;
        text-align: center;
    }
}
```

start-soc.page.ts

```
/**
 * Name:        Wiliam Nolan
 * Student ID:  C00216986
 * Description: Typescript file for the start soc page.
 */
import { Component, OnInit, OnDestroy } from '@angular/core';
import { Soc } from 'src/app/models/soc.model';
import { SocQuestion } from 'src/app/models/soc-question.model';
import { Subscription } from 'rxjs';
import { Router, ActivatedRoute } from '@angular/router';
import { NavController, AlertController } from '@ionic/angular';
import { SocsService } from 'src/app/services/socs.service';
import { SocQuestionService } from 'src/app/services/soc-
question.service';
import { QuestionService } from 'src/app/services/question.service';

@Component({
  selector: 'app-start-soc',
  templateUrl: './start-soc.page.html',
  styleUrls: ['./start-soc.page.scss'],
})
export class StartSocPage implements OnInit, OnDestroy {
  soc: Soc;
  questions: SocQuestion[];
  isLoading = false;
  isLoadingQuestions = false;
  started = false;
  countdown: string;
  private socSub: Subscription;
  private socQuestionsSub: Subscription;

  constructor(
    private router: Router,
    private route: ActivatedRoute,
    private navCtrl: NavController,
    private socsService: SocsService,
    private socQuestionsService: SocQuestionService,
    private questionsService: QuestionService,
    private alertCtrl: AlertController
  ) { }

  ngOnInit() {
    this.questionsService.reset();
    this.route.paramMap.subscribe(paramMap => {
      if (!paramMap.has('socId')) {
        this.navCtrl.navigateBack('view-soc');
        return;
      }
      this.isLoading = true;
      this.socSub = this.socsService
        .getSoc(paramMap.get('socId'))
        .subscribe(soc => {
          this.soc = soc;
          this.isLoading = false;
        }, error => {
          this.alertCtrl.create({
            header: 'An error occured',
            message: 'Could not load SOC',
            buttons: [
              {
                text: 'Okay',
                handler: () => {
                  this.router.navigate(['view-soc']);
                }
              }
            ]
          }).then(alertEl => alertEl.present());
        });
      this.isLoadingQuestions = true;
      this.socQuestionsSub = this.socQuestionsService.socQuestions.subscri
be(socQuestions => {
        this.questions = socQuestions;
        this.isLoadingQuestions = false;
      });
    });
  }

  startSoc() {
    this.started = true;
    let counter = 0;
    const i = setInterval(() => {
      switch (counter) {
        case 0:
          this.countdown = '3';
          break;
        case 1:
          this.countdown = '2';
          break;
        case 2:
          this.countdown = '1';
          break;
        case 3:
          this.countdown = 'GO!';
          break;
        default:
          break;
      }

      counter++;
      if (counter === 4) {
          clearInterval(i);
          this.router.navigate(['/', 'take-
soc', this.soc.id, this.questions[0].id]);
      }
    }, 1000);
  }

  ngOnDestroy() {
    if (this.socSub) {
      this.socSub.unsubscribe();
    }
    if (this.socQuestionsSub) {
      this.socQuestionsSub.unsubscribe();
    }
  }

}
```

View SOC

view-soc-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { ViewSocPage } from './view-soc.page';

const routes: Routes = [
  {
    path: '',
    redirectTo: 'tabs/todo',
    pathMatch: 'full'
  },
  {
    path: 'tabs',
    component: ViewSocPage,
    children: [
      {
        path: 'todo',
        loadChildren: () => import('./todo/todo.module').then( m => m.Todo
PageModule)
      },
      {
        path: 'search',
        loadChildren: () => import('./search/search.module').then( m => m.
SearchPageModule)
      },
      {
        path: '',
        redirectTo: 'todo',
        pathMatch: 'full'
      }
    ]
  },
  {
    path: 'cud-soc',
    loadChildren: () => import('./cud-soc/cud-
soc.module').then( m => m.CudSocPageModule)
  },
  {
    path: ':socId',
    loadChildren: () => import('./view-soc-detail/view-soc-
detail.module').then( m => m.ViewSocDetailPageModule)
```

```
  },
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule],
})
export class ViewSocPageRoutingModule {}
```

view-soc-routing.page.html

```
<ion-tabs>
  <ion-tab-bar slot="bottom">
    <ion-tab-button tab="todo">
      <ion-label>Pending</ion-label>
      <ion-icon name="list"></ion-icon>
    </ion-tab-button>
    <ion-tab-button tab="search">
      <ion-label>Search</ion-label>
      <ion-icon name="search"></ion-icon>
    </ion-tab-button>
  </ion-tab-bar>
</ion-tabs>
```

CUD SOC

cud-soc.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-back-button defaultHref="view-soc"></ion-back-button>
    </ion-buttons>
    <ion-title>Create SOC</ion-title>
    <ion-buttons slot="primary">
      <ion-button (click)="onCreateSoc()" [disabled]="!form.valid">
        <ion-icon name="checkmark" slot="icon-only"></ion-icon>
      </ion-button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>

<ion-content>
  <form [formGroup]="form">
    <!-- Value : {{ form.value | json}} -->
    <ion-grid>
      <ion-row>
        <ion-col size-sm="6" offset-sm="3">
          <ion-item>
            <ion-label position="floating">Name</ion-label>
            <ion-input
              type="text"
              autocomplete
              autocorrect
              formControlName="name"
            ></ion-input>
          </ion-item>
        </ion-col>
      </ion-row>
      <ion-row>
        <ion-col size-sm="6" offset-sm="3">
          <ion-item>
            <ion-label position="floating">Description</ion-label>
            <ion-input
              type="text"
              autocomplete
              autocorrect
              formControlName="description"
            ></ion-input>
          </ion-item>
        </ion-col>
      </ion-row>
      <ion-row>
        <ion-col size-sm="6" offset-sm="3">
          <ion-item>
            <ion-label position="floating">Percentage</ion-label>
            <ion-input
              type="number"
              autocomplete
              autocorrect
              formControlName="percentage"
            ></ion-input>
          </ion-item>
        </ion-col>
      </ion-row>
      <div formArrayName="questions">
          <div *ngFor="let question of form['controls'].questions['control
s']; let iQ = index">
              <div formGroupName="{{iQ}}">
                <ion-row>
                  <ion-col size-sm="6" offset-sm="3">
                    <ion-item>
                      <ion-label position="floating">Question</ion-label>
                      <ion-input
                        type="text"
                        autocomplete
                        autocorrect
                        formControlName="questionName"
                      ></ion-input>
                    </ion-item>
                  </ion-col>
                </ion-row>
                <ion-row>
                  <ion-col size-sm="3" offset-sm="3">
                    <ion-
button color="primary" size="full" (click)="addAnswer(iQ)">Add Answer</ion
-button>
                  </ion-col>
                  <ion-col size-sm="3">
                    <ion-
button color="danger" size="full" (click)="deleteQuestion(iQ)">Delete Ques
tion</ion-button>
                  </ion-col>
                </ion-row>
                <div formArrayName="answers">
                    <div *ngFor="let answer of question['controls'].answer
s['controls']; let iA = index">
                      <div formGroupName="{{iA}}">
                        <ion-row>
                          <ion-col size-sm="6" offset-sm="3">
                            <ion-item>
                              <ion-
label position="floating">{{ iA === 0 ? 'Correct Answer' : 'Answer'}}</ion
-label>
                              <ion-input
                                type="text"
                                autocomplete
                                autocorrect
                                formControlName="answerName"
                              ></ion-input>
                            </ion-item>
                          </ion-col>
                        </ion-row>
                        <ion-row>
                          <ion-col size-sm="6" offset-sm="3">
                            <ion-
button color="danger" size="full" (click)="deleteAnswer(iA, iQ)">Delete An
swer</ion-button>
                          </ion-col>
                        </ion-row>
                      </div>
                    </div>
                </div>
              </div>
            </div>
      </div>
    </ion-grid>
  </form>
</ion-content>
<ion-footer>
  <ion-grid>
    <ion-row>
      <ion-col size-sm="6" offset-sm="3">
        <ion-
button color="primary" size="full" (click)="addQuestion()">Add Question</i
on-button>
      </ion-col>
    </ion-row>
  </ion-grid>
</ion-footer>
```

cud-soc.page.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: Typescript file for the cud soc page.
 */
import { Component, OnInit } from '@angular/core';
import { FormGroup, FormBuilder, FormArray, FormGroupName, Form, Validator
s, FormControl } from '@angular/forms';
import { AlertController, LoadingController } from '@ionic/angular';
import { SocsService } from 'src/app/services/socs.service';
import { Router } from '@angular/router';

@Component({
  selector: 'app-cud-soc',
  templateUrl: './cud-soc.page.html',
  styleUrls: ['./cud-soc.page.scss'],
})
export class CudSocPage implements OnInit {
  form: FormGroup;
  errorMsg: string;

  constructor(
    private fb: FormBuilder,
    private socsService: SocsService,
    private router: Router,
    private loadingCtrl: LoadingController,
    private alertCtrl: AlertController,
  ) { }

  ngOnInit() {
    this.form = this.fb.group({
      name: new FormControl(null, {
        updateOn: 'blur',
        validators: [Validators.required]
      }),
      description: new FormControl(null, {
        updateOn: 'blur',
        validators: [Validators.required]
      }),
      percentage: new FormControl(null, {
        updateOn: 'blur',
        validators: [Validators.required]
      }),
      questions: this.fb.array([
        this.initQuestion()
      ])
    });
    this.deleteQuestion(0);
  }

  initQuestion() {
    return this.fb.group({
      questionName: new FormControl(null, {
        updateOn: 'blur',
        validators: [Validators.required, Validators.maxLength(100)]
      }),
      answers: this.fb.array([
        this.initAnswer()
      ])
    });
  }

  initAnswer() {
    return this.fb.group({
      answerName: new FormControl(null, {
        updateOn: 'blur',
        validators: [Validators.required, Validators.maxLength(100)]
      }),
    });
  }


  addQuestion() {
    const control = this.form.controls.questions as FormArray;
    control.push(this.initQuestion());
  }


  addAnswer(iQ) {
    const control = (this.form.controls.questions as FormArray).at(iQ).get
('answers') as FormArray;
    console.log(control);
    if (control.length < 4) {
      control.push(this.initAnswer());
    } else {
      this.showAlert('Limit to 4 answers per question');
    }
  }

  deleteQuestion(i) {
    const control = this.form.controls.questions as FormArray;
    control.removeAt(i);
  }


  deleteAnswer(i, iQ) {
    const control = (this.form.controls.questions as FormArray).at(iQ).get
('answers') as FormArray;
    control.removeAt(i);
  }

  private showAlert(message: string) {
    this.alertCtrl.create(
      {
        header: 'Error',
        message,
        buttons: ['Ok']
      }
    )
    .then(alertEl =>
      alertEl.present()
    );
  }

  onCreateSoc() {
    if (!this.form.valid) {
      return;
    }
    this.loadingCtrl.create({
      message: 'Creating SOC...'
    }).then(loadingEl => {
      loadingEl.present();
      this.socsService.createSoc(
        this.form.value.name,
        this.form.value.description,
        this.form.value.percentage,
        this.form.value.questions,
      ).subscribe(() => {
        loadingEl.dismiss();
        this.form.reset();
        this.router.navigateByUrl('/view-soc');
      });
    });
  }
}
```

Search

search.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-menu-button menuId="navId"></ion-menu-button>
    </ion-buttons>
    <ion-title>Search</ion-title>
    <ion-
buttons slot="primary" *ngIf="!isLoadingUser && userData.role > 1">
      <ion-button routerLink="/view-soc/cud-soc">
        <ion-icon name="add" slot="icon-only"></ion-icon>
      </ion-button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>

<ion-content class="ion-padding">
  <ion-
searchbar showCancelButton="focus" (ionInput)="filter($event)"></ion-
searchbar>
  <div *ngIf="isLoading" class="ion-text-center">
    <ion-spinner color="primary"></ion-spinner>
  </div>
  <p *ngIf="!isLoading && loadedSocs.length > 0 && listSocs.length <= 0" c
lass="ion-text-center">
    SOC not found.
  </p>
  <ion-list *ngIf="!isLoading && !isLoadingSoc && listSocs.length > 0">
    <ion-item
      *ngFor="let soc of listSocs"
      [routerLink]="['/', 'view-soc', soc.id]"
      detail
    >
    {{ soc.name }}
    </ion-item>
  </ion-list>
</ion-content>
```

search.page.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:  C00216986
 * Description: Typescript file for the search page.
 */
import { Component, OnInit, OnDestroy } from '@angular/core';
import { Soc } from 'src/app/models/soc.model';
import { UserData } from 'src/app/models/userData.model';
import { Subscription } from 'rxjs';
import { SocsService } from 'src/app/services/socs.service';
import { AuthService } from 'src/app/services/auth.service';

@Component({
  selector: 'app-search',
  templateUrl: './search.page.html',
  styleUrls: ['./search.page.scss'],
})
export class SearchPage implements OnInit, OnDestroy {

  loadedSocs: Soc[];
  listSocs: Soc[];
  userData: UserData;
  private socsSub: Subscription;
  private authSub: Subscription;
  isLoading = false;
  isLoadingSoc = false;
  isLoadingUser = false;
  isItemAvailable = false;

  constructor(
    private socsService: SocsService,
    private authService: AuthService
  ) { }

  ngOnInit() {
    this.isLoadingSoc = true;
    this.socsSub = this.socsService.socs.subscribe(socs => {
      this.loadedSocs = socs;
      this.listSocs = socs;
      this.isLoadingSoc = false;
    });
    this.isLoadingUser = true;
    this.authSub = this.authService.currUser.subscribe(userData => {
      this.userData = userData;
      this.isLoadingUser = false;
    });
  }

  ionViewWillEnter() {
    this.isLoading = true;
    this.socsService.fetchSocs().subscribe(() => {
      this.isLoading = false;
    });
  }

  initializeItems() {
    this.listSocs = this.loadedSocs;
  }

  filter(event: any) {
    this.initializeItems();
    const val = event.target.value;
    if (val && val.trim() !== '') {
      this.isItemAvailable = true;
      this.listSocs = this.listSocs.filter((item) => {
        return (item.name.toLowerCase().indexOf(val.toLowerCase()) > -1);
      });
    }
  }

  ngOnDestroy() {
    if (this.socsSub) {
      this.socsSub.unsubscribe();
    }
    if (this.authSub) {
      this.authSub.unsubscribe();
    }
  }

}
```

Todo

todo.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-menu-button menuId="navId"></ion-menu-button>
    </ion-buttons>
    <ion-title>Pending</ion-title>
    <ion-
buttons slot="primary" *ngIf="!isLoadingUser && userData.role > 1">
      <ion-button routerLink="/view-soc/cud-soc">
        <ion-icon name="add" slot="icon-only"></ion-icon>
      </ion-button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>

<ion-content class="ion-padding">
  <div *ngIf="isLoading" class="ion-text-center">
    <ion-spinner color="primary"></ion-spinner>
  </div>
  <p *ngIf="!isLoading && loadedSocs.length <= 0" class="ion-text-
center">No SOCs found.</p>
  <ion-grid *ngIf="!isLoading && loadedSocs.length > 0">
    <ion-row>
      <ion-col size="12" size-sm="8" offset-sm="2" class="ion-text-
center">
        <ion-card>
          <ion-card-header>
            <ion-card-title>{{ loadedSocs[0].name }}</ion-card-title>
          </ion-card-header>
          <ion-card-content>
            <p>{{ loadedSocs[0].description }} </p>
          </ion-card-content>
          <div class="ion-text-right">
            <ion-button
              fill="clear"
              color="primary"
              [routerLink]="['/', 'view-soc', loadedSocs[0].id]"
              >
              More
            </ion-button>
          </div>
        </ion-card>
      </ion-col>
    </ion-row>
    <ion-row>
      <ion-col size="12" size-sm="8" offset-sm="2" class="ion-text-
center">
        <ion-virtual-scroll
          [items]="listedLoadedPlaces"
          approxItemHeight="60px"
        >
          <ion-item
            [routerLink]="['/', 'view-soc', soc.id]"
            detail
            *virtualItem="let soc"
          >
            <ion-label>
              <h2>{{ soc.name }}</h2>
              <p>{{ soc.description }}</p>
            </ion-label>
          </ion-item>
        </ion-virtual-scroll>
      </ion-col>
    </ion-row>
  </ion-grid>
</ion-content>
```

todo.page.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: Typescript file for the todo page.
 */
import { Component, OnInit, OnDestroy } from '@angular/core';
import { Soc } from 'src/app/models/soc.model';
import { Subscription } from 'rxjs';
import { SocsService } from 'src/app/services/socs.service';
import { AuthService } from 'src/app/services/auth.service';
import { UserData } from 'src/app/models/userData.model';

@Component({
  selector: 'app-todo',
  templateUrl: './todo.page.html',
  styleUrls: ['./todo.page.scss'],
})
export class TodoPage implements OnInit, OnDestroy {
  loadedSocs: Soc[];
  listedLoadedPlaces: Soc[];
  private socsSub: Subscription;
  private authSub: Subscription;
  isLoading = false;
  isLoadingUser = false;
  userData: UserData;

  constructor(
    private socsService: SocsService,
    private authService: AuthService
  ) { }

  ngOnInit() {
    // console.log(navigator.userAgent);
    // const ua = navigator.userAgent;
    // if (/Android|webOS|iPhone|iPad|iPod|BlackBerry|IEMobile|Opera Mini|
Mobile|mobile|CriOS/i.test(ua)) {
    //    console.log('mobile');
    // } else if (/Chrome/i.test(ua)) {
    //   console.log('chrome');
    // } else {
    //       console.log('desktop');
    // }
    this.isLoadingUser = true;
    this.authSub = this.authService.currUser.subscribe(userData => {
      this.userData = userData;
      this.isLoadingUser = false;
    });
    this.socsSub = this.socsService.pendingSocs.subscribe(socs => {
      this.loadedSocs = socs;
      this.listedLoadedPlaces = this.loadedSocs.slice(1);
    });
  }

  ionViewWillEnter() {
    this.isLoading = true;
    this.socsService.getPendingSocs(this.userData.id).subscribe(() => {
      this.isLoading = false;
    });
  }

  ngOnDestroy() {
    if (this.socsSub) {
      this.socsSub.unsubscribe();
    }
    if (this.authSub) {
      this.authSub.unsubscribe();
    }
  }

}
```

View SOC Detail

view-soc-detail-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { ViewSocDetailPage } from './view-soc-detail.page';

const routes: Routes = [
  {
    path: '',
    component: ViewSocDetailPage
  },
  {
    path: 'edit-delete-soc',
    loadChildren: () => import('./edit-delete-soc/edit-delete-
soc.module').then( m => m.EditDeleteSocPageModule)
  },
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule],
})
export class ViewSocDetailPageRoutingModule {}
```

view-soc-detail.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-back-button defaultHref="view-soc"></ion-back-button>
    </ion-buttons>
    <ion-title>{{ isLoading ? 'Loading...' : soc.name }}</ion-title>
    <ion-buttons slot="primary">
      <ion-button *ngIf="!isLoading" [routerLink]="['edit-delete-soc']">
        <ion-icon name="create" slot="icon-only"></ion-icon>
      </ion-button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>

<ion-content class="ion-padding">
  <div class="ion-text-center" *ngIf="isLoading">
```

```
    <ion-spinner color="primary"></ion-spinner>
  </div>
  <ion-grid class="ion-no-padding" *ngIf="!isLoading">
    <ion-row>
      <ion-col size-sm="6" offset-sm="3" class="ion-padding ion-text-
center">
        <h1>{{ soc.description }}</h1>
      </ion-col>
    </ion-row>
    <ion-row>
      <ion-col size-sm="6" offset-sm="3" class="ion-padding ion-text-
center">
        <ion-button
            color="primary"
            expand="block"
            [routerLink]="['/take-soc', soc.id, loadedSocQuestions[0].id]"
            (click)="reset()"
          >
          <!-- [routerLink]="['/take-soc','start-soc', soc.id]" -->
            Start SOC
        </ion-button>
      </ion-col>
    </ion-row>
  </ion-grid>
  <p *ngIf="!isLoading && loadedSocQuestions.length <= 0" class="ion-text-
center">No Questions found.</p>
  <ion-grid *ngIf="!isLoading && loadedSocQuestions.length > 0">
    <ion-row>
      <ion-col size="12" size-sm="8" offset-sm="2" class="ion-text-
center">
        <h2>Questions</h2>
        <ion-virtual-scroll
          [items]="loadedSocQuestions"
          approxItemHeight="60px"
        >
          <ion-item
            *virtualItem="let socQuestion"
          >
            <ion-label class="ion-text-wrap">
              <h3>{{ socQuestion.name }}</h3>
            </ion-label>
          </ion-item>
        </ion-virtual-scroll>
      </ion-col>
    </ion-row>
  </ion-grid>
</ion-content>
```

view-soc-detail.page.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: Typescript file for the view soc detail page.
 */
import { Component, OnInit, OnDestroy } from '@angular/core';
import { Soc } from 'src/app/models/soc.model';
import { SocQuestion } from 'src/app/models/soc-question.model';
import { SocAnswer } from 'src/app/models/soc-answer.model';
import { Subscription } from 'rxjs';
import { Router, ActivatedRoute } from '@angular/router';
import { NavController, AlertController } from '@ionic/angular';
import { SocsService } from 'src/app/services/socs.service';
import { SocQuestionService } from 'src/app/services/soc-
question.service';
import { QuestionService } from 'src/app/services/question.service';

@Component({
  selector: 'app-view-soc-detail',
  templateUrl: './view-soc-detail.page.html',
  styleUrls: ['./view-soc-detail.page.scss'],
})
export class ViewSocDetailPage implements OnInit, OnDestroy {
  soc: Soc;
  loadedSocQuestions: SocQuestion[];
  loadedSocAnswers: SocAnswer[];
  private socSub: Subscription;
  private socQuestionSub: Subscription;
  private socAnswerSub: Subscription;
  public socId: string;
  public questionId: string;
  isLoading = false;
  isAnswersLoading = false;

  constructor(
    private router: Router,
    private route: ActivatedRoute,
    private navCtrl: NavController,
    private socsService: SocsService,
    private socQuestionsService: SocQuestionService,
    private alertCtrl: AlertController,
    private questionsService: QuestionService,
  ) { }

  ngOnInit() {
    this.route.paramMap.subscribe(paramMap => {
      if (!paramMap.has('socId')) {
        this.navCtrl.navigateBack('/view-soc');
        return;
      }
      this.isLoading = true;
      this.socSub = this.socsService
        .getSoc(paramMap.get('socId'))
        .subscribe(soc => {
          this.soc = soc;
          this.isLoading = false;
        }, error => {
          this.alertCtrl.create({
            header: 'An error occured',
            message: 'Could not load SOC',
            buttons: [
              {
                text: 'Okay',
                handler: () => {
                  this.router.navigate(['view-soc']);
                }
              }
            ]
          }).then(alertEl => alertEl.present());
        });
    });
    this.socQuestionSub = this.socQuestionsService.socQuestions.subscribe(
socQuestions => {
      this.loadedSocQuestions = socQuestions;
    });
  }

  ionViewWillEnter() {
    this.socId = this.route.snapshot.paramMap.get('socId');
    this.isLoading = true;
    this.socQuestionsService.fetchQuestions(this.socId).subscribe(() => {
      this.isLoading = false;
    });
  }

  reset() {
    this.questionsService.reset();
  }

  ngOnDestroy() {
    if (this.socSub) {
      this.socSub.unsubscribe();
    }
    if (this.socQuestionSub) {
      this.socQuestionSub.unsubscribe();
    }
    if (this.socAnswerSub) {
      this.socAnswerSub.unsubscribe();
    }
  }
}
```

Edit/Delete SOC

edit-delete-soc.page.html

```html
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start">
      <ion-back-button defaultHref="view-soc"></ion-back-button>
    </ion-buttons>
    <ion-title>Edit SOC</ion-title>
    <ion-buttons *ngIf="!isLoading && edit" slot="secondary">
      <ion-button (click)="deleteSOCAlert()">
        <ion-icon name="trash" slot="icon-only"></ion-icon>
      </ion-button>
    </ion-buttons>
    <ion-buttons *ngIf="!isLoading && edit" slot="secondary">
      <ion-button (click)="saveChangesAlert()" [disabled]="!form.valid">
        <ion-icon name="checkmark" slot="icon-only"></ion-icon>
      </ion-button>
    </ion-buttons>
    <ion-buttons *ngIf="!isLoading && !edit" slot="primary">
      <ion-button (click)="toggleEdit()" [disabled]="!form.valid">
        <ion-icon name="create" slot="icon-only"></ion-icon>
      </ion-button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>

<ion-content>
  <form *ngIf="!isLoading" [formGroup]="form">
    <!-- Value : {{ form.value | json}} -->
    <ion-grid>
      <ion-row>
        <ion-col size-sm="6" offset-sm="3">
          <ion-item>
            <ion-label position="floating">Name</ion-label>
            <ion-input
              type="text"
              autocomplete
              autocorrect
              formControlName="name"
              readonly = "{{ !edit }}"
            ></ion-input>
          </ion-item>
        </ion-col>
      </ion-row>
      <ion-row>
        <ion-col size-sm="6" offset-sm="3">
          <ion-item>
            <ion-label position="floating">Description</ion-label>
            <ion-input
              type="text"
              autocomplete
              autocorrect
              formControlName="description"
              readonly = "{{ !edit }}"
            ></ion-input>
          </ion-item>
        </ion-col>
      </ion-row>
      <ion-row>
        <ion-col size-sm="6" offset-sm="3">
          <ion-item>
            <ion-label position="floating">Percentage</ion-label>
            <ion-input
              type="number"
              autocomplete
              autocorrect
              formControlName="percentage"
              readonly = "{{ !edit }}"
            ></ion-input>
          </ion-item>
        </ion-col>
      </ion-row>
      <div formArrayName="questions">
          <div *ngFor="let question of form['controls'].questions['control
s']; let iQ = index">
              <div formGroupName="{{iQ}}">
                <ion-row>
                  <ion-col size-sm="6" offset-sm="3">
                    <hr>
                    <ion-item>
                      <ion-label position="floating">Question</ion-label>
                      <ion-input
                        type="text"
                        autocomplete
                        autocorrect
                        formControlName="questionName"
                        readonly = "{{ !edit }}"
                      ></ion-input>
                    </ion-item>
                  </ion-col>
                </ion-row>
                <ion-row *ngIf="edit">
                  <ion-col size-sm="3" offset-sm="3">
                    <ion-
button color="primary" size="full" (click)="addAnswer(iQ)">Add Answer</ion
-button>
                  </ion-col>
                  <ion-col size-sm="3">
                    <ion-
button color="danger" size="full" (click)="deleteQuestionAlert(iQ)">Delete
 Question</ion-button>
                  </ion-col>
                </ion-row>
                <div formArrayName="answers">
                    <div *ngFor="let answer of question['controls'].answer
s['controls']; let iA = index">
                      <div formGroupName="{{iA}}">
                        <ion-row>
                          <ion-col size-sm="6" offset-sm="3">
                            <ion-item>
                              <ion-
label position="floating">{{ iA === 0 ? 'Correct Answer' : 'Answer'}}</ion
-label>
                              <ion-input
                                type="text"
                                autocomplete
                                autocorrect
                                formControlName="answerName"
                                readonly = "{{ !edit }}"
                              ></ion-input>
                            </ion-item>
                          </ion-col>
                        </ion-row>
                        <ion-row *ngIf="edit">
                          <ion-col size-sm="6" offset-sm="3">
                            <ion-
button color="danger" size="full" (click)="deleteAnswerAlert(iA, iQ)">Dele
te Answer</ion-button>
                          </ion-col>
                        </ion-row>
                      </div>
                    </div>
                </div>
              </div>
            </div>
      </div>
    </ion-grid>
  </form>
</ion-content>
<ion-footer *ngIf="edit">
  <ion-grid>
    <ion-row>
      <ion-col size-sm="6" offset-sm="3">
        <ion-
button color="primary" size="full" (click)="addQuestion()">Add Question</i
on-button>
      </ion-col>
    </ion-row>
  </ion-grid>
</ion-footer>
```

edit-delete-soc.page.scss

```
hr {
    display: block;
    overflow: hidden;
    background-color: black;
    border-style: solid;
}
```

edit-delete-soc.page.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: Typescript file for the edit/delete soc page.
 */
import { Component, OnInit, OnDestroy } from '@angular/core';
import { FormGroup, FormBuilder, Validators, FormControl, FormArray } from
 '@angular/forms';
import { SocsService } from 'src/app/services/socs.service';
import { Router, ActivatedRoute } from '@angular/router';
import { LoadingController, AlertController, NavController } from '@ionic/
angular';
```

```
import { SocQuestionService } from 'src/app/services/soc-
question.service';
import { Soc } from 'src/app/models/soc.model';
import { Subscription } from 'rxjs';
import { SocAnswerService } from 'src/app/services/soc-answer.service';

@Component({
  selector: 'app-edit-delete-soc',
  templateUrl: './edit-delete-soc.page.html',
  styleUrls: ['./edit-delete-soc.page.scss'],
})
export class EditDeleteSocPage implements OnInit, OnDestroy {
  form: FormGroup;
  errorMsg: string;
  socId: string;
  soc: Soc;
  socSub: Subscription;
  isLoading = false;
  index: number;
  edit = false;

  constructor(
    private fb: FormBuilder,
    private socsService: SocsService,
    private socQuestionsService: SocQuestionService,
    private socAnswerService: SocAnswerService,
    private router: Router,
    private route: ActivatedRoute,
    private loadingCtrl: LoadingController,
    private navCtrl: NavController,
    private alertCtrl: AlertController,
  ) { }

  ngOnInit() {
    this.route.paramMap.subscribe(paramMap => {
      if (!paramMap.has('socId')) {
        this.navCtrl.navigateBack('/view-soc');
        return;
      }
      this.isLoading = true;
      this.socSub = this.socsService
        .getSoc(paramMap.get('socId'))
        .subscribe(soc => {
          this.soc = soc;
          this.initForm();
          this.isLoading = false;
        }, error => {
          this.alertCtrl.create({
            header: 'An error occured',
            message: 'Could not load SOC',
            buttons: [
              {
                text: 'Okay',
                handler: () => {
                  this.router.navigate(['view-soc']);
                }
              }
            ]
          }).then(alertEl => alertEl.present());
        });
    });
  }

  toggleEdit() {
    this.edit = !this.edit;
  }

  initForm() {
    this.form = this.fb.group({
      name: new FormControl(this.soc.name, {
        updateOn: 'blur',
        validators: [Validators.required]
      }),
      description: new FormControl(this.soc.description, {
        updateOn: 'blur',
        validators: [Validators.required]
      }),
      percentage: new FormControl(this.soc.percent, {
        updateOn: 'blur',
        validators: [Validators.required]
      }),
      questions: this.fb.array([
        this.initQuestion()
      ])
    });
    this.deleteQuestion(0);
    this.initExistingQuestions();
    this.index = 0;
    // tslint:disable-next-line: forin
    for (const key in this.soc.questions) {
      this.initExistingAnswers(this.soc.questions[key].answers, this.index
);
      this.index++;
    }
  }

  initExistingQuestions() {
    const control = this.form.controls.questions as FormArray;
    // tslint:disable-next-line: forin
    for (const key in this.soc.questions) {
      control.push(this.fb.group(
        {
          questionName: new FormControl(this.soc.questions[key].name, {
            updateOn: 'blur',
            validators: [Validators.required, Validators.maxLength(100)]
          }),
          questionId: key,
          answers: this.fb.array([
            this.initAnswer()
          ])
        }
      ));
    }
  }

  initExistingAnswers(answers: any[], index: number) {
    const control = (this.form.controls.questions as FormArray).at(index).
get('answers') as FormArray;
    this.deleteAnswer(0, index);
    // tslint:disable-next-line: forin
    for (const key in answers) {
      if (answers[key].isAnswer) {
        control.insert(0, this.fb.group(
          {
            answerName: new FormControl(answers[key].name, {
              updateOn: 'blur',
              validators: [Validators.required, Validators.maxLength(100)]
            }),
            answerId: key
          }
        ));
      } else {
        control.push(this.fb.group(
          {
            answerName: new FormControl(answers[key].name, {
              updateOn: 'blur',
              validators: [Validators.required, Validators.maxLength(100)]
            }),
            answerId: key
          }
        ));
      }
    }
  }

  initQuestion() {
    return this.fb.group({
      questionName: new FormControl(null, {
        updateOn: 'blur',
        validators: [Validators.required, Validators.maxLength(100)]
      }),
      questionId: null,
      answers: this.fb.array([
        this.initAnswer()
      ])
    });
  }

  initAnswer() {
    return this.fb.group({
      answerName: new FormControl('', {
        updateOn: 'blur',
        validators: [Validators.required, Validators.maxLength(100)]
      }),
      answerId: null
    });
  }


  addQuestion() {
    const control = this.form.controls.questions as FormArray;
    control.push(this.initQuestion());
  }


  addAnswer(iQ) {
    const control = (this.form.controls.questions as FormArray).at(iQ).get
('answers') as FormArray;
    if (control.length < 4) {
      control.push(this.initAnswer());
    } else {
      this.showAlert('Limit to 4 answers per question');
    }
  }

  async deleteQuestionAlert(iQ) {
    const alert = await this.alertCtrl.create({
      header: 'Confirm delete',
      message: 'Are you sure you want to delete this question?' ,
      buttons: [
        {
          text: 'Delete Question',
          handler: () => {
            this.deleteQuestion(iQ);
          }
        },
        {
          text: 'Cancel',
          role: 'cancel',
          cssClass: 'secondary',
          handler: () => {
            return;
          }
        }
      ]
    });

    await alert.present();
  }

  deleteQuestion(i) {
    const control = this.form.controls.questions as FormArray;
    const questionId = control.value[i].questionId;
    if (questionId !== null) {
      this.loadingCtrl.create({
        message: 'Deleting Question...'
      }).then(loadingEl => {
          loadingEl.present();
          this.socQuestionsService.deleteQuestion(
            this.soc.id,
          questionId
        ).subscribe(() => {
          loadingEl.dismiss();
        });
      });
    }
    control.removeAt(i);
  }

  async deleteAnswerAlert(iA, iQ) {
    const alert = await this.alertCtrl.create({
      header: 'Confirm delete',
      message: 'Are you sure you want to delete this answer?' ,
      buttons: [
        {
          text: 'Delete Answer',
          handler: () => {
            this.deleteAnswer(iA, iQ);
          }
        },
        {
          text: 'Cancel',
          role: 'cancel',
          cssClass: 'secondary',
          handler: () => {
            return;
          }
        }
      ]
    });

    await alert.present();
  }


  deleteAnswer(i, iQ) {
    const control = (this.form.controls.questions as FormArray).at(iQ).get
('answers') as FormArray;
    const questionId = this.form.controls.questions.value[iQ].questionId;
    const answerId = control.value[i].answerId;
    if (questionId !== null && answerId !== null) {
      this.loadingCtrl.create({
        message: 'Deleting Answer...'
      }).then(loadingEl => {
          loadingEl.present();
          this.socAnswerService.deleteAnswer(
            this.soc.id,
            questionId,
            answerId
          ).subscribe(() => {
            loadingEl.dismiss();
          });
      });
    }
    control.removeAt(i);
  }

  private showAlert(message: string) {
    this.alertCtrl.create(
      {
        header: 'Error',
        message,
        buttons: ['Ok']
      }
    )
    .then(alertEl =>
      alertEl.present()
    );
  }


  async saveChangesAlert() {
    const alert = await this.alertCtrl.create({
      header: 'Confirm changes',
      message: 'Would you like to save changes?',
      buttons: [
        {
          text: 'Save Changes',
          handler: () => {
            this.onEditSoc();
          }
        },
        {
          text: 'Cancel',
          role: 'cancel',
          cssClass: 'secondary',
          handler: () => {
            this.edit = false;
            this.initForm();
          }
        }
      ]
    });

    await alert.present();
  }

  onEditSoc() {
    if (!this.form.valid) {
      return;
    }
    this.loadingCtrl.create({
      message: 'Updating SOC...'
    }).then(loadingEl => {
      loadingEl.present();
      this.socsService.updateSoc(
        this.soc.id,
        this.form.value.name,
        this.form.value.description,
        this.form.value.percentage,
        this.form.value.questions,
      ).subscribe(() => {
        loadingEl.dismiss();
        this.edit = false;
      });
    });
  }

  async deleteSOCAlert() {
    const alert = await this.alertCtrl.create({
      header: 'Confirm deletion',
      message: 'Are you sure you want to delete this SOC?',
      buttons: [
        {
          text: 'Delete SOC',
          handler: () => {
            this.onDeleteSOC();
          }
        },
        {
          text: 'Cancel',
          role: 'cancel',
          cssClass: 'secondary',
        }
      ]
    });

    await alert.present();
  }

  onDeleteSOC() {
    this.loadingCtrl.create({
      message: 'Deleting SOC...'
    }).then(loadingEl => {
      loadingEl.present();
      this.socsService.deleteSOC(
        this.soc.id,
      ).subscribe(() => {
        this.router.navigateByUrl('/view-soc');
        loadingEl.dismiss();
        this.edit = false;
      });
    });
  }

  ngOnDestroy() {
    if (this.socSub) {
      this.socSub.unsubscribe();
    }
  }
}
```

Services

auth.guard.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: This file handles all the whether
 *              a user has access to a specified page.
 */
import { Injectable } from '@angular/core';
import {
  CanActivate,
  CanActivateChild,
  CanLoad,
  Route,
  UrlSegment,
  ActivatedRouteSnapshot,
  RouterStateSnapshot,
  UrlTree,
  Router
} from '@angular/router';
import { Observable } from 'rxjs';
import { AuthService } from './auth.service';
import { take, tap } from 'rxjs/operators';

@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate, CanActivateChild, CanLoad {
  constructor(private authService: AuthService, private router: Router) {}

  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<b
oolean | UrlTree> | boolean | UrlTree {
    return true;
  }
  canActivateChild(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<b
oolean | UrlTree> | boolean | UrlTree {
    return true;
  }
  canLoad(
    route: Route,
    segments: UrlSegment[]): Observable<boolean> | Promise<boolean> | bool
ean {
    return this.authService.userIsAuthenticated.pipe(
      take(1),
      tap(isAuthenticated => {
          if (!isAuthenticated) {
            this.router.navigateByUrl('/auth');
          }
        }
      )
    );
  }
}
```

auth.service.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: This service handles all the user authentication
 *               from the back-end.
 */
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { BehaviorSubject } from 'rxjs';
import { User } from '../models/user.model';
import { map, tap, switchMap, take } from 'rxjs/operators';
import { UserData } from '../models/userData.model';
import { environment } from 'src/environments/environment';

export interface AuthResponseData {
  kind: string;
  idToken: string;
  email: string;
  refreshToken: string;
  localId: string;
  expiresIn: string;
  registered?: boolean;
}

interface UserDataInterface {
  id: string;
  email: string;
  fname: string;
  lname: string;
  role: number;
  socs: string[];
}

@Injectable({
  providedIn: 'root'
})
export class AuthService {
  private _user = new BehaviorSubject<User>(null);
  private _currUser = new BehaviorSubject<UserData>(null);
  private _users = new BehaviorSubject<UserData[]>([]);


  /**
   * Returns whether the current user is authenticated.
   *
   * @return    true/ false if user is authenticated
   */
  get userIsAuthenticated() {
    return this._user.asObservable().pipe(
      map(user => {
        if (user) {
          return !!user.token;
        } else {
          return false;
        }
      })
    );
  }

  /**
   * Returns the current user ID.
   *
   * @return    User ID
   */
  get userId() {
    return this._currUser.asObservable().pipe(
      map(user => {
        if (user) {
          return user.id;
        } else {
          return null;
        }
      })
    );
  }

  /**
   * Returns the current user's role.
   *
   * @return      -1 - 2
   *              -1 - Default Unassigned Role
   *              0 - Crew Member
   *              1 - Crew Trainer
   *              2 - Manager
   */
  get userRole() {
    return this._currUser.asObservable().pipe(
      map(user => {
        if (user) {
          return user.role;
        } else {
          return null;
        }
      })
    );
  }

  /**
   * Returns the current user.
   *
   * @return    User
   */
  get currUser() {
    return this._currUser.asObservable();
  }

  /**
   * Returns the all users.
   *
   * @return    Users
   */
  get users() {
    return this._users.asObservable();
  }
```

```
constructor(
  private http: HttpClient,
) { }

/**
 * Fetches all the users from back-end.
 *
 * @return    Subscribable.
 */
fetchUsers() {
  return this.http
    .get<{[key: string]: UserDataInterface}>(
      'https://fyp-wnolan.firebaseio.com/user.json'
    )
    .pipe(map(resData => {
      const users = [];
      for (const key in resData) {
        if (resData.hasOwnProperty(key)) {
          users.push(new UserData(
            key,
            resData[key].email,
            resData[key].fname,
            resData[key].lname,
            resData[key].role,
            resData[key].socs
          ));
        }
      }
      users.sort((a, b) => {
        return a.lname.localeCompare(b.lname) ||
          a.fname.localeCompare(b.fname) || 0;
      });
      return users;
    }),
    tap(users => {
      this._users.next(users);
    })
  );
}

/**
 * Fetches specified user from back-end.
 *
 * @param     string id
```

```
   * @return    Subscribable.
   */
  getUser(id: string) {
    return this.http
    .get<UserData>(
      `https://fyp-wnolan.firebaseio.com/user/${id}.json`
    )
    .pipe(
      map(resData => {
          return new UserData(
            id,
            resData.email,
            resData.fname,
            resData.lname,
            resData.role,
            resData.socs
          );
      })
    );
  }


  /**
   * Signs up a user,
   *
   * @param     string email
   * @param     string password
   * @return    Subscribable.
   */
  signUp(email: string, password: string) {
    return this.http.post<AuthResponseData>(
      `https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=${
        environment.firebaseAPIKey
      }`,
      {email, password, returnSecureToken: true}
    ).pipe(tap(this.setUserData.bind(this)));
  }

  /**
   * Creates user object in back-end.
   *
   * @param     string userId
   * @param     string email
   * @param     string fname
   * @param     string lname
   * @return    Subscribable.
   */
  createUser(userId: string, email: string, fname: string, lname: string)
{
    const newUser = new UserData(
      userId,
      email,
      fname,
      lname,
      -1,
      []
    );
    return this.http
      .put(`https://fyp-wnolan.firebaseio.com/user/${userId}.json`, {
        ...newUser,
        id: null
      })
      .pipe(
        tap(() => {
          this._currUser.next(newUser);
        })
      );
  }

  /**
   * Logs user into system.
   *
   * @param     string email
   * @param     string password
   * @return    Subscribable.
   */
  login(email: string, password: string) {
    return this.http.post<AuthResponseData>(
      `https://identitytoolkit.googleapis.com/v1/accounts:signInWithPasswo
rd?key=${
        environment.firebaseAPIKey
      }`,
      {email, password}
    ).pipe(tap(this.setUserData.bind(this)));
  }

  /**
   * Updates currently logged in user info.
   *
   * @param     string id
   * @return    Subscribable.
   */
  updateCurrUser(id: string) {
    return this.http
      .get<UserDataInterface>(
        `https://fyp-wnolan.firebaseio.com/user/${id}.json`
      )
      .pipe(
        map(userData => {
          this._currUser.next(new UserData(
            id,
            userData.email,
            userData.fname,
            userData.lname,
            userData.role,
            userData.socs
          ));
        })
      );
  }


  /**
   * Updates a user's role.
   *
   * @param     number role
   * @param     UserData selectedUser
   * @return    Subscribable.
   */
  updateRole(role: number, selectedUser: UserData) {
    let generatedId: string;
    const newUser = new UserData(
      selectedUser.id,
      selectedUser.email,
      selectedUser.fname,
      selectedUser.lname,
      role,
      []
    );
    return this.http.put<{name: string}>(`https://fyp-
wnolan.firebaseio.com/user/${selectedUser.id}.json`, {
      ...newUser,
      id: null
    })
    .pipe(
      switchMap(resData => {
        generatedId = resData.name;
        return this.users;
      }),
      take(1),
      tap(users => {
        newUser.id = generatedId;
        this._users.next(users.concat(newUser));
      })
    );
  }

  /**
   * Logs the current user out.
   */
  logout() {
    this._user.next(null);
  }

  /**
   * Sets the current user's authenication data.
   *
   * @param AuthResponseData userData
   */
  private setUserData(userData: AuthResponseData) {
    const expirationTime = new Date(new Date().getTime() + +userData.expir
esIn * 1000);
    this._user.next(new User(
      userData.localId,
      userData.email,
      userData.idToken,
      expirationTime
    ));
  }
}
```

leaderboard.service.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: This service handles all the access
 *              to the back-end for all leaderboard actions.
 */
import { Injectable } from '@angular/core';
import { AuthService } from './auth.service';
import { HttpClient } from '@angular/common/http';
import { Leaderboard } from '../models/Leaderboard.model';
import { BehaviorSubject } from 'rxjs';
import { switchMap, take, tap, map } from 'rxjs/operators';

interface LeaderboardData {
  id: string;
  name: string;
  score: number;
  date: Date;
}

@Injectable({
  providedIn: 'root'
})
export class LeaderboardService {
  private _leaderboard = new BehaviorSubject<Leaderboard[]>([]);
  oldRecord: Leaderboard;


  constructor(
    private http: HttpClient,
    private authService: AuthService
  ) { }

  /**
   * Returns the leaderboard from the back-end.
   *
   * @return Leaderboard
   */
  get leaderboard() {
    return this._leaderboard.asObservable();
  }

  /**
   * Compares new leaderboard score with old leaderboard score and stores
best score.
   *
   * @param string socId
   * @param string name
   */
  compareScores(socId: string, name: string) {
    this.fetchLeaderboard(socId).subscribe(leaderboard => {
      if (leaderboard.find(x => x.name === name) !== undefined) {
        this.oldRecord = leaderboard.find(x => x.name === name);
      }
    });
  }

  /**
   * Adds new leaderboard record to back-end.
   *
   * @param string socId
   * @param number score
   * @returns Subscribable
   */
  addLeaderboard(socId: string, score: number) {
    let generateId: string;
    let name;
    let uid;
    this.authService.currUser.subscribe(user => {
      if (user) {
        name = user.fname + ' ' + user.lname;
        uid = user.id;
      }
    });
    const newLeaderboard = new Leaderboard (
      Math.random().toString(),
      name,
      score,
      new Date()
    );
    if (this.oldRecord === undefined) {
      return this.http
        .put<{name: string}>(`https://fyp-
wnolan.firebaseio.com/leaderboard/${socId}/${uid}.json`, {
          ...newLeaderboard,
          id: null,
        })
        .pipe(
          switchMap(resData => {
            generateId = resData.name;
            return this.leaderboard;
          }),
          take(1),
          tap(leaderboard => {
            newLeaderboard.id = generateId;
            this._leaderboard.next(leaderboard.concat(newLeaderboard));
          })
        );
    } else if (this.oldRecord.score < score && this.oldRecord) {
      return this.http
        .put<{name: string}>(`https://fyp-
wnolan.firebaseio.com/leaderboard/${socId}/${uid}.json`, {
          ...newLeaderboard,
          id: null,
        })
        .pipe(
          switchMap(resData => {
            generateId = resData.name;
            return this.leaderboard;
          }),
          take(1),
          tap(leaderboard => {
            newLeaderboard.id = generateId;
            this._leaderboard.next(leaderboard.concat(newLeaderboard));
          })
        );
    } else {
      return this.http
        .put<{name: string}>(`https://fyp-
wnolan.firebaseio.com/leaderboard/${socId}/${uid}.json`, {
          ...this.oldRecord,
          id: null,
        })
        .pipe(
          switchMap(resData => {
            generateId = resData.name;
            return this.leaderboard;
          }),
          take(1),
          tap(leaderboard => {
            newLeaderboard.id = generateId;
            this._leaderboard.next(leaderboard.concat(newLeaderboard));
          })
        );
    }
```

```
  }

  /**
   * Fetches the SOC leaderboard from the back-end.
   *
   * @param string socId
   * @returns Subscribable
   */
  fetchLeaderboard(socId: string) {
    return this.http
    .get<LeaderboardData>(
      `https://fyp-wnolan.firebaseio.com/leaderboard/${socId}.json`
    )
    .pipe(
      map(resData => {
        const leaderboard = [];
        for (const key in resData) {
          if (resData.hasOwnProperty(key)) {
            leaderboard.push(new Leaderboard(
              key,
              resData[key].name,
              resData[key].score,
              resData[key].date,
            )
            );
          }
        }
        leaderboard.sort((a, b) => {
          return parseFloat(b.score) - parseFloat(a.score);
        });
        return leaderboard;
      }),
      tap(leaderboard => {
        this._leaderboard.next(leaderboard);
      })
    );
  }
}
```

question.service.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: This service handles all actions
 *              for the take SOC process.
 */
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class QuestionService {
  incorrectQuestions: string[] = [];
  finalIncorrectNames: string[] = [];
  finalIncorrectIDs: string[] = [];
  firstRun = true;
  result = 0;
  progress = 0;
  score = 0;
  streak = 0;

  constructor() {
  }

  /**
   * Resets take SOC process.
   */
  reset() {
    this.incorrectQuestions = [];
    this.finalIncorrectIDs = [];
    this.finalIncorrectNames = [];
    this.firstRun = true;
    this.result = 0;
    this.progress = 0;
    this.score = 0;
    this.streak = 0;
  }

  /**
   * Adds incorrect questions to list
   *
   * @param string questionID
   * @param string questionName
   */
  addIncorrectQuestion(questionID: string, questionName: string) {
    this.incorrectQuestions.push(questionID);
    if (!this.finalIncorrectIDs.includes(questionID)) {
      this.finalIncorrectIDs.push(questionID);
      this.finalIncorrectNames.push(questionName);
    }
  }

  /**
   * Returns list of incorrect question IDs.
   *
   * @returns List of incorrect question IDs
   */
  getIncorrectQuestions() {
    return this.incorrectQuestions;
  }


  /**
   * Returns final list of incorrect question IDs.
   *
   * @returns Final list of incorrect question IDs
   */
  getFinalIncorrectQuestionIDs() {
    return this.finalIncorrectIDs;
  }

  /**
   * Returns final list of incorrect question names.
   *
   * @returns Final list of incorrect question names
   */
  getFinalIncorrectQuestionNames() {
    return this.finalIncorrectNames;
  }

  /**
   * Removes incorrect question from list.
   */
  removeIncorrectQuestion() {
    this.incorrectQuestions.shift();
  }

  /**
   * Return whether its the users first run through the take SOC process.
   *
   * @returns True/False whether its the users first run through the take
SOC process
```

```
   */
  isFirstRun() {
    return this.firstRun;
  }

  /**
   * Sets first run as done.
   */
  firstRunDone() {
    this.firstRun = false;
  }

  /**
   * Increments the result. Increments streak index.
   */
  addResult() {
    this.result = ++this.result;
    this.streak = ++this.streak;
  }

  /**
   * Returns the result.
   *
   * @returns Result.
   */
  getResult() {
    return this.result;
  }

  /**
   * Increments progression for progress bar.
   */
  addProgress() {
    this.progress++;
  }

  /**
   * Returns the current progress.
   *
   * @returns Current progress
   */
  getProgress() {
    return this.progress;
  }
  /**
   * Adds to the users score taking the time and streak bonus into account
.
   *
   * @param number bonus
   */
  addScore(bonus: number) {
    if (bonus > 0) {
      this.score += Math.round(bonus);
    }
    if (this.streak > 0) {
      this.score += ((this.streak - 1) * 100);
    }
    this.score += 100;
  }


  /**
   * Returns score.
   *
   * @returns Score
   */
  getScore() {
    return this.score;
  }

  /**
   * Resets the user's streak.
   */
  resetStreak() {
    this.streak = 0;
  }

}
```

results.service.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: This service handles all the access
 *               to the back-end for all results actions.
 */
```

```
import { Injectable } from '@angular/core';
import { BehaviorSubject } from 'rxjs';
import { Result } from '../models/result.model';
import { HttpClient } from '@angular/common/http';
import { switchMap, take, tap, map } from 'rxjs/operators';
import { AuthService } from './auth.service';
import { Feedback } from '../models/feedback.model';

interface ResultData {
  result: number;
  total: number;
  incorrect: string[];
  feedback: Feedback[];
  date: Date;
}

interface FeedbackData {
  feedback: string;
  senderName: string;
  date: Date;
}

@Injectable({
  providedIn: 'root'
})
export class ResultsService {
private _results = new BehaviorSubject<Result[]>([]);
private _feedback = new BehaviorSubject<Feedback[]>([]);

  constructor(
    private http: HttpClient,
    private authService: AuthService
  ) { }

  /**
   * Returns results.
   *
   * @returns Results
   */
  get results() {
    return this._results.asObservable();
  }

  /**
```

```
   * Returns feedback.
   *
   * @returns Feedback
   */
  get feedback() {
    return this._feedback.asObservable();
  }

  /**
   * Fetches specified user and SOC results.
   *
   * @param string userId
   * @param string socId
   * @returns Subscribable.
   */
  fetchResults(userId: string, socId: string) {
    return this.http
      .get<{[key: string]: ResultData}>(
        `https://fyp-wnolan.firebaseio.com/result/${userId}/${socId}.json`
      )
      .pipe(map(resData => {
        const results = [];
        for (const key in resData) {
          if (resData.hasOwnProperty(key)) {
            results.push(new Result(
              key,
              resData[key].result,
              resData[key].total,
              resData[key].incorrect,
              resData[key].feedback,
              resData[key].date
            ));
          }
        }
        results.sort((a, b) => {
          return b.date.localeCompare(a.date) || 0;
        });
        return results;
      }),
      tap(results => {
        this._results.next(results);
      })
      );
  }
  /**
   * Adds a result to the back-end.
   *
   * @param string userId
   * @param string socId
   * @param number result
   * @param number total
   * @param string[] incorrect
   * @returns Subscribable
   */
  addResult(userId: string, socId: string, result: number, total: number,
incorrect: string[]) {
    let generateId: string;
    const newResult = new Result(
      Math.random().toString(),
      result,
      total,
      incorrect,
      [],
      new Date()
    );
    return this.http
      .post<{name: string}>(`https://fyp-
wnolan.firebaseio.com/result/${userId}/${socId}.json`, {
        ...newResult,
        id: null
      })
      .pipe(
        switchMap(resData => {
          generateId = resData.name;
          return this.results;
        }),
        take(1),
        tap(results => {
          newResult.id = generateId;
          this._results.next(results.concat(newResult));
        })
      );
  }

  /**
   * Returns specified result.
   *
   * @param string id
   * @param string socId
   * @param string userId
   * @returns Subscribable
   */
  getResult(id: string, socId: string, userId: string) {
    return this.http
    .get<ResultData>(
      `https://fyp-
wnolan.firebaseio.com/result/${userId}/${socId}/${id}.json`
    )
    .pipe(
      map(resultData => {
        return new Result(
          id,
          resultData.result,
          resultData.total,
          resultData.incorrect,
          resultData.feedback,
          resultData.date
        );
      })
    );
  }

  /**
   * Adds feedback to specified result
   *
   * @param string feedback
   * @param string senderName
   * @param string userId
   * @param string socId
   * @param string resultId
   * @returns Subscribable
   */
  addFeedback(feedback: string, senderName: string, userId: string, socId:
 string, resultId: string) {
    let generateId: string;
    const newFeedback = new Feedback(
      Math.random().toString(),
      feedback,
      senderName,
      new Date()
    );
    return this.http
    .post<{name: string}>(`https://fyp-
wnolan.firebaseio.com/result/${userId}/${socId}/${resultId}/feedback.json`
, {
      ...newFeedback,
      id: null
    })
    .pipe(
      switchMap(resData => {
        generateId = resData.name;
        return this.feedback;
      }),
      take(1),
      tap(feedback => {
        newFeedback.id = generateId;
        this._feedback.next(feedback.concat(newFeedback));
      })
    );
  }

  /**
   * Fetches all users results.
   *
   * @param string userId
   * @returns Subscribable
   */
  getResultObject(userId: string) {
    return this.http
    .get(
      `https://fyp-wnolan.firebaseio.com/result/${userId}.json`
    )
    .pipe(
      map(resultData => {
        return resultData;
      })
    );
  }
}
```

review-detail.service.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: This service handles all the access
 *              to the back-end for all review details actions.
 */
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { map, tap } from 'rxjs/operators';
import { BehaviorSubject } from 'rxjs';
import { Soc } from '../models/soc.model';
import { SocQuestion } from '../models/soc-question.model';
import { SocAnswer } from '../models/soc-answer.model';

interface SocData {
  description: string;
  questions: SocQuestion[];
  name: string;
  percent: number;
}

interface SocQuestionData {
  answers: SocAnswer[];
  name: string;
}

@Injectable({
  providedIn: 'root'
})

export class ReviewDetailService {
  private socIds: string[];
  private _socs = new BehaviorSubject<Soc[]>([]);
  private _questions = new BehaviorSubject<SocQuestion[]>([]);

  constructor(
    private http: HttpClient,
  ) { }

  /**
   * Returns the SOCs from the back-end.
   *
   * @returns SOCs
   */
  get socs() {
    return this._socs.asObservable();
  }
```

```
/**
 * Returns the questions from the back-end.
 *
 * @returns Questions
 */
get questions() {
  return this._questions.asObservable();
}

/**
 * Fetches SOCs from back-end from a specified list of SOCs.
 *
 * @param string userId
 * @returns Subscribable
 */
getSocs(userId: string) {
  this.getSocIds(userId).subscribe();
  return this.http
  .get<{[key: string]: SocData}>(
    'https://fyp-wnolan.firebaseio.com/soc.json'
  )
  .pipe(map(resData => {
    const socs = [];
    for (const key in resData) {
      if (resData.hasOwnProperty(key)) {
        if (this.socIds.indexOf(key) !== -1) {
          socs.push(new Soc(
            key,
            resData[key].name,
            resData[key].description,
            resData[key].percent,
            []
          )
          );
        }
      }
    }
    return socs;
  }),
  tap(socs => {
    this._socs.next(socs);
  })
);
}
```

```
/**
 * Gets the IDs of all the SOCs the user has done.
 *
 * @param string userId
 * @returns Subscribable
 */
getSocIds(userId: string) {
  return this.http
    .get<{[key: string]: string[]}>(
      `https://fyp-wnolan.firebaseio.com/result/${userId}.json`
    )
    .pipe(map(resData => {
      const socs = [];
      for (const key in resData) {
        if (resData.hasOwnProperty(key)) {
          socs.push(key);
        }
      }
      return socs;
    }),
    tap(socs => {
      this.socIds = socs;
    }));
}

/**
 * Gets specified questions from back-end.
 *
 * @param string[] questionIds
 * @param string socId
 * @returns Subscribable
 */
getQuestions(questionIds: string[], socId: string) {
  return this.http
  .get<{[key: string]: SocQuestionData}>(
    `https://fyp-wnolan.firebaseio.com/soc/${socId}/questions.json`
  )
  .pipe(map(resData => {
    const questions = [];
    for (const key in resData) {
      if (resData.hasOwnProperty(key)) {
        if (questionIds.indexOf(key) !== -1) {
          questions.push(new SocQuestion(
              key,
              resData[key].name,
              resData[key].answers
            )
            );
          }
        }
      }
      return questions;
    }),
    tap(questions => {
      this._questions.next(questions);
    })
  );
  }
}
```

soc-answer.service.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: This service handles all the access
 *              to the back-end for all SOC answers actions.
 */
import { Injectable } from '@angular/core';
import { SocAnswer } from '../models/soc-answer.model';
import { BehaviorSubject } from 'rxjs';
import { AuthService } from 'src/app/services/auth.service';
import { HttpClient } from '@angular/common/http';
import { switchMap, take, tap, map } from 'rxjs/operators';

interface SocAnswerData {
  name: string;
  isAnswer: boolean;
}

@Injectable({
  providedIn: 'root'
})
export class SocAnswerService {
  private _socAnswers = new BehaviorSubject<SocAnswer[]>([]);

  constructor(
    private authService: AuthService,
    private http: HttpClient
  ) { }

  /**
   * Returns answers from back-end.
   *
   * @returns Answers.
   */
  get socAnswers() {
    return this._socAnswers.asObservable();
  }

  /**
   * Fetches answers from back-end.
   *
   * @param string socId
   * @param string questionId
   * @returns Subscribable
   */
  fetchAnswers(socId: string, questionId: string) {
    return this.http
      .get<{[key: string]: SocAnswerData}>(
        `https://fyp-
wnolan.firebaseio.com/soc/${socId}/questions/${questionId}/answers.json`
      )
      .pipe(map(resData => {
        const socQuestions = [];
        for (const key in resData) {
          if (resData.hasOwnProperty(key)) {
            socQuestions.push(new SocAnswer(
              key,
              resData[key].name,
              resData[key].isAnswer
            )
            );
          }
        }
        return socQuestions;
      }),
      tap(socAnswers => {
        this._socAnswers.next(socAnswers);
      })
    );
  }
  /**
   * Creates new answer and sends it to back-end.
   *
   * @param string socId
   * @param string questionId
   * @param string name
   * @param boolean isAnswer
   * @returns Subscribable
   */
  createAnswer(socId: string, questionId: string, name: string, isAnswer:
boolean) {
    let generatedId: string;
    const newSocAnswer = new SocAnswer(
      Math.random().toString(),
      name,
      isAnswer
    );
    return this.http
      .post<{name: string}>(`https://fyp-
wnolan.firebaseio.com/soc/${socId}/questions/${questionId}/answers.json`,
{
        ...newSocAnswer,
        id: null
      })
      .pipe(
        switchMap(resData => {
          generatedId = resData.name;
          return this.socAnswers;
        }),
        take(1),
        tap(socAnswers => {
          newSocAnswer.id = generatedId;
          this._socAnswers.next(socAnswers.concat(newSocAnswer));
        })
      );
  }

  /**
   * Updates answer and sends it to back-end.
   *
   * @param string socId
   * @param string questionId
   * @param string name
   * @param boolean isAnswer
   * @returns Subscribable
   */
  updateAnswer(socId: string, questionId: string, answerId: string, name:
string, isAnswer: boolean) {
    let generatedId: string;
    const newSocAnswer = new SocAnswer(
      Math.random().toString(),
      name,
      isAnswer
    );
    return this.http
      .put<{name: string}>(`https://fyp-
wnolan.firebaseio.com/soc/${socId}/questions/${questionId}/answers/${answe
rId}.json`, {
        ...newSocAnswer,
        id: null
      })
      .pipe(
        switchMap(resData => {
          generatedId = resData.name;
          return this.socAnswers;
        }),
        take(1),
        tap(socAnswers => {
          newSocAnswer.id = generatedId;
          this._socAnswers.next(socAnswers.concat(newSocAnswer));
          this.fetchAnswers(socId, questionId).subscribe();
        })
      );
  }

  /**
   * Deletes answer from back-end.
   *
   * @param string socId
   * @param string questionId
   * @param string answerId
   * @returns Subscribable.
   */
  deleteAnswer(socId: string, questionId: string, answerId: string) {
    return this.http.delete(`https://fyp-
wnolan.firebaseio.com/soc/${socId}/questions/${questionId}/answers/${answe
rId}.json`)
```

```
      .pipe(switchMap(() => {
        return this.socAnswers;
      }),
      take(1),
      tap(answers => {
        this._socAnswers.next(answers.filter(b => b.id !== answerId));
      }));
  }
}
```

soc-question.service.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: This service handles all the access
 *               to the back-end for all SOC Questions actions.
 */
import { Injectable } from '@angular/core';
import { SocAnswer } from '../models/soc-answer.model';
import { SocQuestion } from '../models/soc-question.model';
import { AuthService } from 'src/app/services/auth.service';
import { HttpClient } from '@angular/common/http';
import { switchMap, take, tap, map } from 'rxjs/operators';
import { BehaviorSubject } from 'rxjs';
import { SocAnswerService } from './soc-answer.service';

interface SocQuestionData {
  answers: SocAnswer[];
  name: string;
}

@Injectable({
  providedIn: 'root'
})
export class SocQuestionService {
  private _socQuestions = new BehaviorSubject<SocQuestion[]>([]);

  constructor(
    private authService: AuthService,
    private http: HttpClient,
    private socAnswersService: SocAnswerService
  ) { }
```

```
/**
 * Returns questions from back-end.
 *
 * @returns Questions
 */
get socQuestions() {
  return this._socQuestions.asObservable();
}

/**
 * Fetches questions from a specified SOC from the back-end.
 *
 * @param string socId
 * @returns Subscribable
 */
fetchQuestions(socId: string) {
  return this.http
    .get<{[key: string]: SocQuestionData}>(
      `https://fyp-wnolan.firebaseio.com/soc/${socId}/questions.json`
    )
    .pipe(map(resData => {
      const socQuestions = [];
      for (const key in resData) {
        if (resData.hasOwnProperty(key)) {
          socQuestions.push(new SocQuestion(
            key,
            resData[key].name,
            resData[key].answers
          )
          );
        }
      }
      return socQuestions;
    }),
    tap(socQuestions => {
      this._socQuestions.next(socQuestions);
    })
  );
}

/**
 * Gets a specified question from the back-end.
 *
 * @param string socId
```

```
   * @param string questionId
   * @returns Subscribable
   */
  getQuestion(socId: string, questionId: string) {
    return this.http
    .get<SocQuestionData>(
      `https://fyp-
wnolan.firebaseio.com/soc/${socId}/questions/${questionId}.json`
    )
    .pipe(
      map(questionData => {
        return new SocQuestion(
          questionId,
          questionData.name,
          questionData.answers
        );
      })
    );
  }

  /**
   * Creates a question and sends it to the back-end.
   *
   * @param string socId
   * @param string name
   * @param any[] answers
   * @returns Subscribable
   */
  createQuestion(socId: string, name: string, answers: any[]) {
    console.log(name);
    let generatedId: string;
    let isAnswer: boolean;
    const newSocQuestion = new SocQuestion(
      Math.random().toString(),
      name,
      []
    );
    return this.http
      .post<{name: string}>(`https://fyp-
wnolan.firebaseio.com/soc/${socId}/questions.json`, {
        ...newSocQuestion,
        id: null
      })
      .pipe(
        switchMap(resData => {
          generatedId = resData.name;
          answers.forEach(answer => {
            if (answers.indexOf(answer) === 0) {
              isAnswer = true;
            } else {
              isAnswer = false;
            }
            this.socAnswersService
                .createAnswer(socId, generatedId, answer.answerName, isAns
wer)
                .subscribe();
          });
          return this.socQuestions;
        }),
        take(1),
        tap(socQuestions => {
          newSocQuestion.id = generatedId;
          this._socQuestions.next(socQuestions.concat(newSocQuestion));
        })
      );
  }

  /**
   * Updates a question.
   *
   * @param string socId
   * @param string questionId
   * @param string name
   * @param any[] answers
   * @returns Subscribable
   */
  updateQuestion(socId: string, questionId: string, name: string, answers:
 any[]) {
    let generatedId: string;
    let isAnswer: boolean;
    const newSocQuestion = new SocQuestion(
      Math.random().toString(),
      name,
      []
    );
    return this.http
      .put<{name: string}>(`https://fyp-
wnolan.firebaseio.com/soc/${socId}/questions/${questionId}.json`, {
        ...newSocQuestion,
        id: null
      })
      .pipe(
        switchMap(resData => {
          generatedId = resData.name;
          answers.forEach(answer => {
            if (answers.indexOf(answer) === 0) {
              isAnswer = true;
            } else {
              isAnswer = false;
            }
            if (answer.answerId === null) {
              this.socAnswersService
              .createAnswer(socId, questionId, answer.answerName, isAnswer
)
              .subscribe();
            } else {
              this.socAnswersService
                .updateAnswer(socId, questionId, answer.answerId, answer.a
nswerName, isAnswer)
                .subscribe();
            }
          });
          return this.socQuestions;
        }),
        take(1),
        tap(socQuestions => {
          newSocQuestion.id = generatedId;
          this._socQuestions.next(socQuestions.concat(newSocQuestion));
        })
      );
  }

  /**
   * Deletes a question from the back-end.
   *
   * @param string socId
   * @param string questionId
   * @returns Subscribable
   */
  deleteQuestion(socId: string, questionId: string) {
    return this.http.delete(`https://fyp-
wnolan.firebaseio.com/soc/${socId}/questions/${questionId}.json`)
```

```
      .pipe(switchMap(() => {
        return this.socQuestions;
      }),
      take(1),
      tap(questions => {
        this._socQuestions.next(questions.filter(b => b.id !== questionId)
);
        this.fetchQuestions(socId).subscribe();
      }));
  }
}
```

socs.service.ts

```
/**
 * Name:         Wiliam Nolan
 * Student ID:   C00216986
 * Description: This service handles all the access
 *              to the back-end for all SOC actions.
 */
import { Injectable } from '@angular/core';
import { take, map, tap, delay, switchMap, filter } from 'rxjs/operators';

import { Soc } from 'src/app/models/soc.model';
import { BehaviorSubject } from 'rxjs';
import { AuthService } from './auth.service';
import { HttpClient } from '@angular/common/http';
import { SocQuestion } from 'src/app/models/soc-question.model';
import { SocQuestionService } from './soc-question.service';
import { Feedback } from '../models/feedback.model';
import { Result } from '../models/result.model';

interface SocData {
  description: string;
  questions: SocQuestion[];
  name: string;
  percent: number;
}
interface ResultData {
  id: string;
  result: number;
  total: number;
  incorrect: string[];
  feedback: Feedback[];
  date: Date;
```

```
}

@Injectable({
  providedIn: 'root'
})
export class SocsService {
  private _socs = new BehaviorSubject<Soc[]>([]);
  private _pendingSocs = new BehaviorSubject<Soc[]>([]);
  socIds: string[];
  allResultIds: string[] = [];
  dates: Date[];
  noResults = false;

  constructor(
    private authService: AuthService,
    private http: HttpClient,
    private socQuestionServive: SocQuestionService
  ) { }

  /**
   * Returns SOCs from back-end.
   *
   * @returns SOCs
   */
  get socs() {
    return this._socs.asObservable();
  }

  /**
   * Returns pending SOCs.
   *
   * @returns Pending SOCs
   */
  get pendingSocs() {
    return this._pendingSocs.asObservable();
  }

  /**
   * Fetches SOCs from back-end.
   *
   * @returns Subscribable
   */
  fetchSocs() {
    return this.http
    .get<{[key: string]: SocData}>(
      'https://fyp-wnolan.firebaseio.com/soc.json'
    )
    .pipe(map(resData => {
      const socs = [];
      for (const key in resData) {
        if (resData.hasOwnProperty(key)) {
          socs.push(new Soc(
            key,
            resData[key].name,
            resData[key].description,
            resData[key].percent,
            []
          )
          );
        }
      }
      return socs;
    }),
    tap(socs => {
      this._socs.next(socs);
    })
  );
}

/**
 * Gets a specified SOC from the back-end.
 *
 * @param string id
 * @returns Subscribable
 */
getSoc(id: string) {
  return this.http
  .get<SocData>(
    `https://fyp-wnolan.firebaseio.com/soc/${id}.json`
  )
  .pipe(
    map(socData => {
      return new Soc(
        id,
        socData.name,
        socData.description,
        socData.percent,
        socData.questions
        );
      })
    );
  }

  /**
   * Creates a new SOC and sends it to the back-end.
   *
   * @param string name
   * @param string description
   * @param number percent
   * @param any[] questions
   * @returns Subscribable
   */
  createSoc(name: string, description: string, percent: number, questions:
 any[]) {
    let generatedId: string;
    const newSoc = new Soc(
      Math.random().toString(),
      name,
      description,
      percent,
      []
    );
    return this.http
      .post<{name: string}>('https://fyp-
wnolan.firebaseio.com/soc.json', {
        ...newSoc,
        id: null
      })
      .pipe(
        switchMap(resData => {
          generatedId = resData.name;
          questions.forEach(question => {
              this.socQuestionServive
                .createQuestion(generatedId, question.questionName, questi
on.answers)
                .subscribe();
            }
          );
          return this.socs;
        }),
        take(1),
        tap(socs => {
          newSoc.id = generatedId;
          this._socs.next(socs.concat(newSoc));
        })
      );
  }

  /**
   * Updates an SOC.
   *
   * @param string socId
   * @param string name
   * @param string description
   * @param number percent
   * @param any[] questions
   * @returns Subscribable
   */
  updateSoc(socId: string, name: string, description: string, percent: num
ber, questions: any[]) {
    let generatedId: string;
    const newSoc = new Soc(
      Math.random().toString(),
      name,
      description,
      percent,
      []
    );
    return this.http
      .put<{name: string}>(`https://fyp-
wnolan.firebaseio.com/soc/${socId}.json`, {
        ...newSoc,
        id: null
      })
      .pipe(
        switchMap(resData => {
          generatedId = resData.name;
          questions.forEach(question => {
              if (question.questionId === null) {
                this.socQuestionServive
                .createQuestion(socId, question.questionName, question.ans
wers)
                .subscribe();
              } else {
                this.socQuestionServive
                .updateQuestion(socId, question.questionId, question.quest
ionName, question.answers)
                .subscribe();
              }
            }
          );
          return this.socs;
        }),
        take(1),
        tap(socs => {
          newSoc.id = generatedId;
          this._socs.next(socs.concat(newSoc));
          this.fetchSocs().subscribe();
        })
      );
  }

  /**
   * Deletes a specified SOC from the back-end.
   *
   * @param string socId
   * @returns Subscribable
   */
  deleteSOC(socId: string) {
    return this.http
      .delete(`https://fyp-wnolan.firebaseio.com/soc/${socId}.json`)
      .pipe(
        switchMap(() => {
          return this.socs;
        }),
        take(1),
        tap(socs => {
          this._socs.next(socs.filter(b => b.id !== socId));
        })
      );
  }

  /**
   * Gets pending SOCs, i.e. SOCs that haven't been completed within the l
ast 6 months.
   *
   * @param string userId
   * @returns Subscribable
   */
  getPendingSocs(userId: string) {
    this.getPendingSocIds(userId).subscribe();
    return this.http
    .get<{[key: string]: SocData}>(
      'https://fyp-wnolan.firebaseio.com/soc.json'
    )
    .pipe(map(resData => {
      const socs = [];
      for (const key in resData) {
        if (resData.hasOwnProperty(key)) {
          if (this.socIds.indexOf(key) !== -
1 || this.allResultIds.indexOf(key) === -1) {
            socs.push(new Soc(
              key,
              resData[key].name,
              resData[key].description,
              resData[key].percent,
              []
            )
            );
          }
        }
      }
      return socs;
    }),
    tap(socs => {
      this._pendingSocs.next(socs);
    })
  );
  }

  /**
   * Gets the IDs of the pending SOCs, i.e. SOCs that haven't been complet
ed within the last 6 months.
   *
   * @param string userId
   * @returns Subscribable
   */
  getPendingSocIds(userId: string) {
    return this.http
      .get<{[key: string]: any}>(
        `https://fyp-wnolan.firebaseio.com/result/${userId}.json`
      )
      .pipe(map(resData => {
        const socs = [];
        var today = new Date();
        var sixMonths = new Date(today);
        sixMonths.setMonth(today.getMonth() - 6);
        for (const key in resData) {
          if (resData.hasOwnProperty(key)) {
            this.dates = [];
            // tslint:disable-next-line: forin
            for (const key2 in resData[key]) {
                this.dates.push(new Date(resData[key][key2].date));
            }
            this.allResultIds.push(key);
            this.dates = this.dates.sort((a, b) => new Date(b).getTime() -
 new Date(a).getTime());
            if (this.dates[0].getTime() < sixMonths.getTime()) {
              socs.push(key);
            }
          }
        }
        return socs;
      }),
      tap(socs => {
        this.socIds = socs;
      }));
  }
}
```