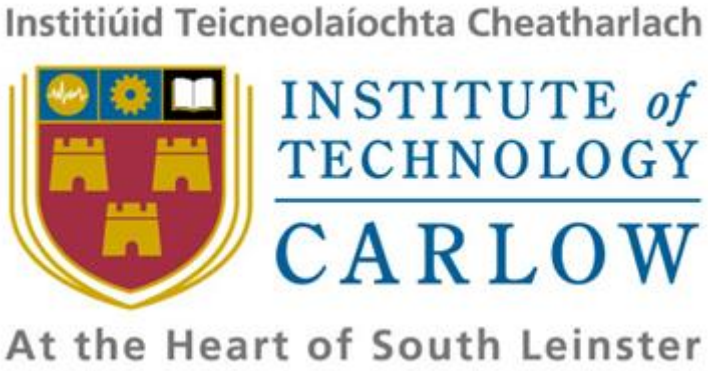


Final Year Project,
Technical Manual,
Gym and Analytical App/Dashboard.

Author: Kevin Quinn.

Student No: C00216607.

Supervisor: Greg Doyle.

Contents

1. Models

1.1. Recorded_workout.model.ts

```
export class RecordedWorkout {
    constructor(
        public id: string,
        public title: string,
        public date: Date,
        public notes: string,
        public exercises: string[],
        public weights: number[],
    ) {}
}
```

1.2. User.model.ts

```
export class User {
    constructor(
        public id: string,
        public name: string,
        public email: string,
        public userType: string,
        public bodyWeigth: number,
        public height: number,
    ) {}
}
```

1.3. Workout.model.ts

```
export class Workout {
    constructor(
        public id: string,
        public title: string,
        public description: string,
        public exercises: string[],
        public sets: number[],
        public reps: number[]
    ) {}
}
```

2. Private

2.1. Create-workout

2.1.1. Create-workout.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start"><ion-back-button defaultHref="/view-
workouts"></ion-back-button></ion-buttons>
    <ion-title>Create Workout</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <form [formGroup]="form" (ngSubmit)="createWorkout()">

    <ion-row>
      <ion-col size-sm="6" offset-sm="3">
        <ion-item>
          <ion-label position="floating">Name</ion-label>
          <ion-input type="text" clear-
input required="true" autocomplete autocorrect formControlName="title"></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" clear-
input required="true" autocomplete autocorrect formControlName="description"><
/ion-input>
        </ion-item>
      </ion-col>
    </ion-row>

    <ion-row>
      <ion-col size-sm="6" offset-sm="3">
        <ion-row>
        <ion-col size="6">
        <div formArrayName="exercises" *ngFor="let exercise of exercises.contr
ols; let i=index">
          <ion-item>
            <ion-label position="floating">Exercises</ion-label>
            <ion-
select [formControlName]="i" okText="Okay" cancelText="Cancel" required="true"
>
```

```
              <ion-select-option value="Bench Press">Bench Press</ion-select-
option>
              <ion-select-option value="Squat">Squat</ion-select-option>
              <ion-select-option value="Deadlift">Deadlift</ion-select-option>
              <ion-select-option value="Pull Ups">Pull Ups</ion-select-option>
              <ion-select-option value="Shoulder Press">Shoulder Press</ion-
select-option>
              <ion-select-option value="Dumbell Curls">Dumbell Curls</ion-
select-option>
              <ion-select-option value="Barbell Curls">Barbell Curls</ion-
select-option>
              <ion-select-option value="Leg Curls">Leg Curls</ion-select-
option>
              <ion-select-option value="Pec Flies">Pec Flies</ion-select-
option>
              <ion-select-option value="Chest Press">Chest Press</ion-select-
option>
              <ion-select-
option value="Incline Barbell Press">Incline Barbell Press</ion-select-option>
              <ion-select-option value="Barbell Row">Barbell Row</ion-select-
option>
              <ion-select-option value="Penlay Row">Penlay Row</ion-select-
option>
              <ion-select-option value="Leg Extensions">Leg Extensions</ion-
select-option>
              <ion-select-
option value="Block Pull Deadlifts">Block Pull Deadlifts</ion-select-option>
              <ion-select-
option value="Deficit Deadlifts">Deficit Deadlifts</ion-select-option>
              <ion-select-option value="Chin Ups">Chin Ups</ion-select-option>
              <ion-select-option value="Push Ups">Push Ups</ion-select-option>
              <ion-select-option value="Sit Ups">Sit Ups</ion-select-option>
              <ion-select-
option value="Lateral Pull Down">Lateral Pull Down</ion-select-option>
              <ion-select-
option value="Tricep Extensions">Tricep Extensions</ion-select-option>
              <ion-select-option value="Hammer Curls">Hammer Curls</ion-
select-option>
            </ion-select>
          </ion-item>
        </div>
        </ion-col>
        <ion-col>
        <div formArrayName="sets" *ngFor="let set of sets.controls; let i=inde
x">
          <ion-item>
            <ion-label position="floating">Sets</ion-label>
```

```
            <ion-input type="number" clear-
input [formControlName]="i" expand="full" required="true"></ion-input>
          </ion-item>
        </div>
        </ion-col>
        <ion-col>
        <div formArrayName="reps" *ngFor="let rep of reps.controls; let i=inde
x">
          <ion-item>
            <ion-label position="floating">Reps</ion-label>
            <ion-input type="number" clear-
input [formControlName]="i" required="true"></ion-input>
          </ion-item>
        </div>
        </ion-col>
        </ion-row>
      </ion-col>
    </ion-row>
  </form>
</ion-content>
<ion-footer>
  <div padding>
    <div>
      <ion-
button color="dark"  size="large" expand="full" [disabled]="exercises.length =
== 1"  (click)="removeExercise()">Remove Exercise</ion-button>
    </div>
    <div>
      <ion-
button color="dark"  size="large" expand="full" [disabled]="this.form.value.ex
ercises[this.form.value.exercises.length - 1] === '' || this.form.value.sets[t
his.form.value.sets.length - 1] === '' || this.form.value.reps[this.form.value
.reps.length - 1] === ''" (click)="addExercise()">Add Exercise</ion-button>
    </div>
    <div>
      <ion-
button color="primary"  size="large" (click)="createWorkout()" [disabled]="!fo
rm.valid" expand="full">Create</ion-button>
    </div>
  </div>
</ion-footer>
```

2.1.2. Create-workout.page.ts

```
import { Component, OnInit } from '@angular/core';
import { FormGroup, FormControl, Validators, FormArray, FormBuilder} from '@an
gular/forms';
import { AngularFirestoreCollection } from '@angular/fire/firestore';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Workout } from '../../models/workout.model';
import { Injectable } from '@angular/core';
import { BehaviorSubject } from 'rxjs';
import { take, tap, delay, switchMap, map } from 'rxjs/operators';
import { WorkoutService } from '../../services/workout.service'
import { Router } from '@angular/router';
import { LoadingController } from '@ionic/angular';


@Component({
  selector: 'app-record-workout',
  templateUrl: './create-workout.page.html',
  styleUrls: ['./create-workout.page.scss'],
})
export class CreateWorkoutPage implements OnInit {

  form: FormGroup;
  isLoading = false;

  constructor(
    private fb: FormBuilder,
    public workoutServ: WorkoutService,
    public router: Router,
    private loadingCtrl: LoadingController
    ) { }

  ngOnInit() {
    this.form = new FormGroup({
      title: new FormControl(null, {
        updateOn: 'blur',
        validators: [Validators.required]
      }),
      description: new FormControl(null, {
        updateOn: 'blur',
        validators: [Validators.required]
      }),
      exercises: new FormControl(null, {
        updateOn: 'blur',
        validators: [Validators.required]
      }),
      sets: new FormControl(null, {
        updateOn: 'blur',
        validators: [Validators.required]
```

```
    }),
    reps: new FormControl(null, {
      updateOn: 'blur',
      validators: [Validators.required]
    }),
  });
  this.form = this.fb.group({
    title: [''],
    description: [''],
    exercise: [''],
    exercises: this.fb.array(['']),
    set: [''],
    sets: this.fb.array(['']),
    rep: [''],
    reps: this.fb.array([''])
  });
}

get exercises() {
  return this.form.get('exercises') as FormArray;
}

get sets() {
  return this.form.get('sets') as FormArray;
}

get reps() {
  return this.form.get('reps') as FormArray;
}

addExercise() {
  this.exercises.push(this.fb.control(''));
  this.sets.push(this.fb.control(''));
  this.reps.push(this.fb.control(''));
}

removeExercise() {
  if(this.exercises.length > 1){
    this.exercises.removeAt(this.exercises.length - 1);
    this.sets.removeAt(this.sets.length - 1);
    this.reps.removeAt(this.reps.length - 1);
  } else {
    console.log("cant remove last exercise");
  }
}

createWorkout() {
  const newWorkout = new Workout(
```

```
      "",
      this.form.value.title,
      this.form.value.description,
      this.form.value.exercises,
      this.form.value.sets,
      this.form.value.reps,
    );
    console.log(newWorkout);
    this.loadingCtrl
        .create({ keyboardClose: true, message: 'Creating Workout...' })
        .then(loadingEl => {
          loadingEl.present();
          this.workoutServ.createWorkout(
            newWorkout
          ).subscribe(() => {
            this.isLoading = false;
            loadingEl.dismiss();
            this.router.navigateByUrl('/view-workouts');
          });
        });
  }
}
```

2.2. Dashboard

2.2.1. Dashboard.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-button slot="start" fill="clear" (click)="logout()"><ion-
icon color="primary" name="log-out"></ion-icon></ion-button>
    <ion-title>Revolute Fitness</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-grid>
    <ion-row>
      <ion-col size-sm="6" offset-sm="3" text-center>
        <h1>
           Welcome {{ this.userName }} !
        </h1>
      </ion-col>
    </ion-row>
    <ion-row>
      <ion-col size-sm="6" offset-sm="3">

          <ion-
button color="dark"  size="large" expand="full" routerLink='/view-
workouts'>View Workouts</ion-button>

          <ion-
button color="dark"  size="large" expand="full" routerLink='/record-workouts-
view'>Record Workout</ion-button>

          <ion-
button color="dark"  size="large" expand="full" routerLink='/view-recorded-
workouts'>View Recorded Workouts</ion-button>

          <ion-
button *ngIf="userType == 'personal_trainer'" color="dark"  size="large" expan
d="full" routerLink='/manage-clients'>Manage Clients</ion-button>

          <ion-
button color="dark"  size="large" expand="full" routerLink='/user-
stats'>View Stats</ion-button>

          <ion-
button color="dark"  size="large" expand="full" (click)="openSite()">Exercise
Analysis</ion-button>

        </ion-col>
    </ion-row>
```

```
  </ion-grid>
</ion-content>
```

2.2.2. Dashboard.page.ts

```
import { Component, OnInit } from '@angular/core';
import { AngularFireAuth } from '@angular/fire/auth';
import { Router } from '@angular/router';
import { NgForm } from '@angular/forms';
import { LoadingController } from '@ionic/angular';
import { UserService } from '../../services/user.service';
import { User } from 'src/app/models/user.model';
import { InAppBrowser } from '@ionic-native/in-app-browser/ngx';

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

  isLoading = false;
  isLogin = true;
  userType: string;
  userName: string;

  constructor(private authService: UserService,
              private router: Router,
              private loadingCtrl: LoadingController,
              private loginServ: UserService,
              private iab: InAppBrowser) { }

  ngOnInit() {
    this.userType = this.loginServ.currentUser.userType;
    this.userName = this.loginServ.currentUserName;
  }

  openSite() {
    this.iab.create('http://kquinn1998.pythonanywhere.com/','_blank');
  }

  logout() {
    this.isLoading = true;
    this.authService.logout();
    this.loadingCtrl
      .create({ keyboardClose: true, message: 'Logging out...' })
      .then(loadingEl => {
        loadingEl.present();
        setTimeout(() => {
          this.isLoading = false;
          loadingEl.dismiss();
          this.router.navigateByUrl('/login');
```

```
        }, 1500);
      });
  }

}
```

2.3. Manage-client-dashboard

2.3.1. Manage-client-dashboard.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start"><ion-back-button defaultHref="/manage-
clients"></ion-back-button></ion-buttons>
    <ion-title>Manage Clients</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-grid>
    <ion-row>
      <ion-col size-sm="6" offset-sm="3">

          <ion-
button color="dark"  size="large" expand="full" routerLink='/view-
workouts'>View Workouts</ion-button>

          <ion-
button color="dark"  size="large" expand="full" routerLink='/view-recorded-
workouts'>View Recorded Workouts</ion-button>

          <ion-
button color="dark"  size="large" expand="full" routerLink='/user-
stats'>View Stats</ion-button>

        </ion-col>
    </ion-row>
  </ion-grid>
</ion-content>
```

2.4. Manage-clients

2.4.1. Manage-clients.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start"><ion-back-button defaultHref="/dashboard"></ion-
back-button></ion-buttons>
    <ion-title>Manage Clients</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <form #add_client_form="ngForm" (ngSubmit)="addClient(add_client_form)">
    <ion-grid>
      <ion-row color="dark" justify-content-center>
        <ion-col align-self-center size-md="8" size-lg="7" size-xs="12">
          <div text-center>
            <h2>Add Client</h2>
          </div>
          <div padding>
            <ion-item no-padding>
            <ion-input type="email" placeholder="yourclient@email.com" clear-
input autocomplete ngModel name="email" required email></ion-input>
            </ion-item>
          </div>
          <div *ngIf="errorMessage">
            <ion-label color="danger">{{ errorMessage }}</ion-label>
          </div>
          <div padding-top>
            <ion-
button color="dark"  size="large" type="submit" expand="full">Add</ion-button>
          </div>
        </ion-col>
      </ion-row>
    </ion-grid>
  </form>

  <p *ngIf="isLoading" class="ion-text-center"><ion-
spinner color="primary"></ion-spinner></p>
  <ion-grid *ngIf="!isLoading">
    <ion-row>
      <ion-col size-sm="6" offset-sm="3">
        <ion-list>
          <div *ngFor="let client of clients">
            <ion-item-sliding padding-bottom padding-end>
              <ion-item (click)="manageClient(client.id)">
                  <ion-label class="ion-text-wrap">
                    <h1>{{ client.name }}</h1>
                  </ion-label>
```

```
                <ion-icon slot="end" md="md-arrow-back"></ion-icon>
              </ion-item>

              <ion-item-options side="end" padding-bottom padding-end>
                <ion-item-
option color="danger" (click)="deleteClient(client.id)" >Delete</ion-item-
option>
              </ion-item-options>
            </ion-item-sliding>
          </div>
        </ion-list>
      </ion-col>
    </ion-row>
  </ion-grid>

</ion-content>
```

2.4.2. Manage-clients.page.ts

```
import { Component, OnInit } from '@angular/core';
import { UserService } from 'src/app/services/user.service';
import { User } from 'src/app/models/user.model';
import { Subscription } from 'rxjs';
import { NgForm } from '@angular/forms';
import { Router } from '@angular/router';

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

  private userSub: Subscription;
  private clientSub: Subscription;

  users: User[] = [ ];
  clients: User[] = [ ];
  isLoading = false;
  errorMessage: string;

  constructor(private userServ: UserService,
              public router: Router) { }

  ngOnInit() {
    this.errorMessage = '';
    this.isLoading = true;
    this.userSub = this.userServ.users.subscribe(users => {
      this.users = users;
    });
    this.clientSub = this.userServ.clientUsers.subscribe(clients => {
      this.clients = clients;
      this.isLoading = false;
    });
  }

  ionViewWillEnter() {
    this.userServ.ptMode = false;
    this.isLoading = true;
    this.userServ.fetchUsers().subscribe(() => {
    });
    this.userServ.fetchClientUsers().subscribe(() => {
      this.isLoading = false;
    });
  }
```

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

  async addClient(form: NgForm) {
    this.isLoading = true;
    this.errorMessage = 'No Client Found';
    for (const user of this.users) {
      if (user.email === form.value.email) {
        this.errorMessage = '';
        this.userServ.addClient(user).subscribe(() => {
          this.userServ.fetchClientUsers().subscribe(() => {
            this.isLoading = false;
          });
        });
        break;
      }
    }
  }

  async deleteClient(id: string) {
    this.isLoading = true;
    this.userServ.deleteClient(id).subscribe(() => {
      this.userServ.fetchClientUsers().subscribe(() => {
        this.isLoading = false;
      });
    });
  }

  manageClient(id: string){
    this.userServ._currentClient = id;
    this.userServ.ptMode = true;
    this.router.navigateByUrl('/manage-client-dashboard');
  }
}
```

2.5. Record-workout

2.5.1. Record-workout.page.html

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

<ion-content>
  <div *ngIf="!isLoading">
  <form [formGroup]="form" (ngSubmit)="recordWorkout()">

    <ion-row>
      <ion-col size-sm="6" offset-sm="3">
        <ion-item>
          <ion-label position="floating">Notes</ion-label>
          <ion-
input type="text" autocomplete autocorrect formControlName="notes"></ion-
input>
        </ion-item>
      </ion-col>
    </ion-row>

    <ion-row>
      <ion-col size-sm="6" offset-sm="3">
        <ion-row>
          <ion-col>
            <div formArrayName="exercises" *ngFor="let exercise of exercises.c
ontrols; let i=index">
              <ion-item>
                <ion-label position="floating">Exercises</ion-label>
                <ion-
input [readonly]='true' type="text" [formControlName]="i"></ion-input>
              </ion-item>
            </div>
          </ion-col>
          <ion-col>
            <div formArrayName="weights" *ngFor="let weight of weights.control
s; let i=index">
              <ion-item>
                <ion-label position="floating">Weights (kg)</ion-label>
                <ion-input type="number" clear-
input [formControlName]="i"></ion-input>
              </ion-item>
            </div>
          </ion-col>
```

```
        </ion-row>
      </ion-col>
    </ion-row>
  </form>
  </div>
</ion-content>
<ion-footer>

  <ion-row>
    <ion-col size-sm="6" offset-sm="3">
      <ion-
button color="primary" (click)='recordWorkout()' size="large" expand="full">Re
cord</ion-button>
    </ion-col>
  </ion-row>

</ion-footer>
```

2.5.2. Record-workout.page.ts

```
import { Component, OnInit } from '@angular/core';
import { FormGroup, FormControl, Validators, FormArray, FormBuilder } from '@a
ngular/forms';
import { AngularFirestoreCollection } from '@angular/fire/firestore';
import { HttpClient } from '@angular/common/http';
import { Workout } from '../../models/workout.model';
import { RecordedWorkout } from '../../models/recorded_workout.model';
import { Injectable } from '@angular/core';
import { BehaviorSubject, Subscription } from 'rxjs';
import { take, tap, delay, switchMap, map } from 'rxjs/operators';
import { WorkoutService } from 'src/app/services/workout.service';
import { ActivatedRoute, Router } from '@angular/router';
import { LoadingController } from '@ionic/angular';

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

  workout: Workout;
  recordedWorkout: RecordedWorkout;
  workoutSub: Subscription;
  isLoading = false;
  editMode = false;
  form: FormGroup;


  constructor(private route: ActivatedRoute,
              private router: Router,
              private workoutServ: WorkoutService,
              private http: HttpClient,
              private fb: FormBuilder,
              private loadingCtrl: LoadingController) { }

  ngOnInit() {
    this.route.paramMap.subscribe(paramMap => {
      if(!paramMap.has('workoutId')){
        this.router.navigateByUrl('/view-workouts');
        return;
      }
      this.isLoading = true;
      this.workoutSub = this.workoutServ.getWorkout(paramMap.get('workoutId'))
.subscribe(workout => {
        this.workout = workout;
        this.setForm();
        this.isLoading = false;
```

```
    });
  });
}

setForm() {
  this.form = new FormGroup({
    notes: new FormControl(null, {
      updateOn: 'blur',
      validators: [Validators.required]
    }),
    exercises: new FormControl(null, {
      updateOn: 'blur',
      validators: [Validators.required]
    }),
    weights: new FormControl(null, {
      updateOn: 'blur',
      validators: [Validators.required]
    }),
  });

  this.form = this.fb.group({
    notes: [''],
    exercise: [''],
    exercises: this.fb.array(this.workout.exercises),
    weight: [''],
    weights: this.fb.array(['']),
  });

  for (let i = 0; i < this.workout.exercises.length - 1; i++) {
    this.weights.push(this.fb.control(''));
  }
}


get exercises() {
  return this.form.get('exercises') as FormArray;
}

get weights() {
  return this.form.get('weights') as FormArray;
}

recordWorkout() {
  const newRecordWorkout = new RecordedWorkout(
    '',
    this.workout.title,
    new Date(),
    this.form.value.notes,
    this.form.value.exercises,
```

```
      this.form.value.weights,
    );
    this.loadingCtrl
        .create({ keyboardClose: true, message: 'Recording Workout...' })
        .then(loadingEl => {
          loadingEl.present();
          this.workoutServ.recordWorkout(
            newRecordWorkout
          ).subscribe(() => {
            this.isLoading = false;
            loadingEl.dismiss();
            this.router.navigateByUrl('/view-recorded-workouts');
          });
        });
  }

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

2.6. Record-workouts-view

2.6.1. Record-workouts-view.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start"><ion-back-button defaultHref="/dashboard"></ion-
back-button></ion-buttons>
    <ion-title>Select A Workout</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <p *ngIf="isLoading" class="ion-text-center"><ion-
spinner color="primary"></ion-spinner></p>
  <ion-grid *ngIf="!isLoading">
    <ion-row>
      <ion-col size-sm="6" offset-sm="3">
        <div *ngIf="workouts.length === 0" text-center>
          <ion-item padding-bottom padding-end>
            <ion-label text-center class="ion-text-wrap">
              <h1>Add a Workout</h1>
            </ion-label>
          </ion-item>
        </div>
        <ion-list>
          <div *ngFor="let workout of workouts">
            <ion-item-sliding padding-bottom padding-end>
              <ion-item [routerLink]="['/','record-workouts-
view', workout.id]">
                  <ion-label text-center class="ion-text-wrap">
                    <h1>{{ workout.title }}</h1>
                  </ion-label>
              </ion-item>
            </ion-item-sliding>
          </div>
        </ion-list>
      </ion-col>
    </ion-row>
  </ion-grid>
</ion-content>
```

2.6.2. Record-workouts-view.page.ts

```
import { Component, OnInit } from '@angular/core';
import { Subscription } from 'rxjs';
import { FormGroup } from '@angular/forms';
import { Workout } from 'src/app/models/workout.model';
import { WorkoutService } from 'src/app/services/workout.service';

@Component({
  selector: 'app-record-workouts-view',
  templateUrl: './record-workouts-view.page.html',
  styleUrls: ['./record-workouts-view.page.scss'],
})
export class RecordWorkoutsViewPage implements OnInit {

  private workoutSub: Subscription;
  form: FormGroup;
  isLoading = false;
  workouts: Workout[];

  constructor(private workoutServ: WorkoutService) { }

  ngOnInit() {
    this.isLoading=true;
    this.workoutSub = this.workoutServ.workouts.subscribe(workouts => {
      this.workouts = workouts;
      this.isLoading=false;
    });
  }

  ionViewWillEnter() {
    this.isLoading=true;
    this.workoutServ.fetchWorkouts().subscribe(() => {
      this.isLoading=false;
    });
  }

  ngOnDestroy(): void {
    if(this.workoutSub){
      this.workoutSub.unsubscribe();
    }
  }

}
```

2.7. User-stats

2.7.1. User-stats.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start"><ion-back-button defaultHref="/dashboard"></ion-
back-button></ion-buttons>
    <ion-title>User Statisics</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>

  <canvas
  baseChart [datasets]="chartData1"
  [labels]="chartLabels1"
  [chartType]="chartType1"
  [colors]="chartColors1"
  [options]="chartOptions1">
  </canvas>

</ion-content>
```

2.7.2. User-stats.page.ts

```
import { Component, OnInit } from '@angular/core';
import { ChartDataSets } from 'chart.js';
import { Label, Color } from 'ng2-charts';
import { WorkoutService } from 'src/app/services/workout.service';
import { Subscription } from 'rxjs';
import { RecordedWorkout } from 'src/app/models/recorded_workout.model';
import { NONE_TYPE } from '@angular/compiler/src/output/output_ast';

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

  // Normal Page Variables
  private workoutSub: Subscription;
  workouts: RecordedWorkout[] = [ ];
  isLoading = false;

  // Chart1Variables
  chartData1: ChartDataSets[] = [{data: [], label: 'Total Workout Weight'}];
  chartLabels1: Label[];
  chartOptions1 = {
    responsive: true,
    maintainAspectRatio: true,
    aspectRatio: 1,
    scales: {
      xAxes: [{
        ticks: {
          display: true,
          autoSkip: true,
          maxRotation: 90,
          minRotation: 90,
          maxTicksLimit: 10
        }
      }],
      yAxes: [{
        ticks: {
          beginAtZero: true,
          callback(value: string) {
                  return value + 'kg';
          }
      },
      }]
    },
    title: {
```

```
    display: true,
    text: 'Total Weight'
  },
  elements: {
    line: {
      tension: 0
    }
  }
};
chartColors1: Color[] = [
  {
    borderColor: '#000000',
    backgroundColor: '#4287f5'
  }
];
chartType1 = 'line';

// Chart2 Variables
chartData2: ChartDataSets[] = [{data: [], label: 'Total Weight'}];
chartLabels2: Label[];
chartOptions2 = {
  responsive: true,
  maintainAspectRatio: true,
  aspectRatio: 1.5,
  scales: {
    xAxes: [{
      ticks: {
        autoSkip: false,
        maxRotation: 90,
        minRotation: 90
      }
    }]
  },
  title: {
    display: true,
    text: 'Past Workouts'
  },
  elements: {
    line: {
      tension: 0
    }
  }
};
chartColors2: Color[] = [
  {
    borderColor: '#000000',
    backgroundColor: '#4287f5'
  }
```

```
  ];
  chartType2 = 'line';

  constructor(private workoutServ: WorkoutService) { }

  ngOnInit() {
    this.isLoading = true;
    this.workoutSub = this.workoutServ.recordedWorkouts.subscribe(recordedWork
outs => {
      this.workouts = recordedWorkouts;
      this.isLoading = false;
      this.setData();
    });
  }

  ionViewWillEnter() {
    this.isLoading = true;
    this.workoutServ.fetchRecordedWorkouts().subscribe(() => {
      this.isLoading = false;
    });
  }

  setData() {

    this.chartData1[0].data = [];
    this.chartLabels1 = [];
    let i = 1;

    for(const entry of this.workouts){
      const date: Date = new Date(entry.date);
      this.chartLabels1.push(entry.title + ' ' + date.getDate() + '/' + date.g
etMonth() + '/' + date.getFullYear());
      i = i + 1;
      let totalWeight = 0;
      for(const weight of entry.weights){
        totalWeight = totalWeight + Number(weight);
      }
      this.chartData1[0].data.push(totalWeight);
    }

  }

}
```

2.8. View-recorded-workout-details

2.8.1. View-recorded-workouts-details.page.html

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

<ion-content>
  <div *ngIf="!isLoading">
  <form [formGroup]="form">

    <ion-row>
      <ion-col size-sm="6" offset-sm="3">
        <ion-item>
          <ion-label position="floating">Notes</ion-label>
          <ion-
input type="text" autocomplete autocorrect formControlName="notes"></ion-
input>
        </ion-item>
      </ion-col>
    </ion-row>

    <ion-row>
      <ion-col size-sm="6" offset-sm="3">
        <ion-row>
          <ion-col>
            <div formArrayName="exercises" *ngFor="let exercise of exercises.c
ontrols; let i=index">
              <ion-item>
                <ion-label position="floating">Exercises</ion-label>
                <ion-
input [readonly]='true' type="text" [formControlName]="i"></ion-input>
              </ion-item>
            </div>
          </ion-col>
          <ion-col>
            <div formArrayName="weights" *ngFor="let weight of weights.control
s; let i=index">
              <ion-item>
                <ion-label position="floating">Weights (kg)</ion-label>
                <ion-
input [readonly]='true' type="number" [formControlName]="i"></ion-input>
              </ion-item>
            </div>
```

```
          </ion-col>
        </ion-row>
      </ion-col>
    </ion-row>
  </form>
  </div>
</ion-content>
```

2.8.2. View-recorded-workouts-details.page.ts

```
import { Component, OnInit } from '@angular/core';
import { FormArray, FormGroup, FormControl, Validators, FormBuilder } from '@a
ngular/forms';
import { Subscription } from 'rxjs';
import { RecordedWorkout } from 'src/app/models/recorded_workout.model';
import { ActivatedRoute, Router } from '@angular/router';
import { WorkoutService } from 'src/app/services/workout.service';
import { HttpClient } from '@angular/common/http';
import { LoadingController } from '@ionic/angular';

@Component({
  selector: 'app-view-recorded-workout-details',
  templateUrl: './view-recorded-workout-details.page.html',
  styleUrls: ['./view-recorded-workout-details.page.scss'],
})
export class ViewRecordedWorkoutDetailsPage implements OnInit {

  recordedWorkout: RecordedWorkout;
  recordedWorkoutSub: Subscription;
  isLoading = false;
  form: FormGroup;


  constructor(private route: ActivatedRoute,
              private router: Router,
              private workoutServ: WorkoutService,
              private http: HttpClient,
              private fb: FormBuilder,
              private loadingCtrl: LoadingController) { }

  ngOnInit() {
    this.route.paramMap.subscribe(paramMap => {
      if(!paramMap.has('recordedWorkoutId')){
        this.router.navigateByUrl('/view-recorded-workouts');
        return;
      }
      this.isLoading = true;
      this.recordedWorkoutSub = this.workoutServ.getRecordedWorkout(paramMap.g
et('recordedWorkoutId')).subscribe(recordedWorkout => {
        this.recordedWorkout = recordedWorkout;
        this.setForm();
        this.isLoading = false;
      });
    });
  }

  setForm() {
    this.form = new FormGroup({
```

```
      notes: new FormControl(null, {
        updateOn: 'blur',
        validators: [Validators.required]
      }),
      exercises: new FormControl(null, {
        updateOn: 'blur',
        validators: [Validators.required]
      }),
      weights: new FormControl(null, {
        updateOn: 'blur',
        validators: [Validators.required]
      }),
    });

    this.form = this.fb.group({
      notes: [this.recordedWorkout.notes],
      exercise: [''],
      exercises: this.fb.array(this.recordedWorkout.exercises),
      weight: [''],
      weights: this.fb.array(this.recordedWorkout.weights),
    });
  }


  get exercises() {
    return this.form.get('exercises') as FormArray;
  }

  get weights() {
    return this.form.get('weights') as FormArray;
  }

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

}
```

2.9. View-recorded-workouts

2.9.1. View-recorded-workouts.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start"><ion-back-button defaultHref="/dashboard"></ion-
back-button></ion-buttons>
    <ion-title>Recorded Workouts</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <p *ngIf="isLoading" class="ion-text-center"><ion-
spinner color="primary"></ion-spinner></p>
  <ion-grid *ngIf="!isLoading">
    <ion-row>
      <ion-col size-sm="6" offset-sm="3">
        <div *ngIf="recordedWorkouts.length === 0" text-center>
          <ion-item padding-bottom padding-end>
            <ion-label text-center class="ion-text-wrap">
              <h1>Record a Workout</h1>
            </ion-label>
          </ion-item>
        </div>
        <ion-list>
          <div *ngFor="let recordedWorkout of recordedWorkouts">
            <ion-item-sliding padding-bottom padding-end>
              <ion-item [routerLink]="['/','view-recorded-
workouts', recordedWorkout.id]" >
                  <ion-label class="ion-text-wrap">
                    <h1>{{ recordedWorkout.title }}</h1>
                  </ion-label>
                  <ion-label class="ion-text-wrap">
                    <h2>{{ recordedWorkout.date | date:'MM/dd/yyyy HH:mm'}}</h
2>
                  </ion-label>
                  <ion-icon slot="end" md="md-arrow-back"></ion-icon>
              </ion-item>

              <ion-item-options side="end" padding-bottom padding-end>
                <ion-item-
option color="danger" (click)="deleteRecordedWorkout(recordedWorkout.id)">Dele
te</ion-item-option>
              </ion-item-options>
            </ion-item-sliding>
          </div>
        </ion-list>
      </ion-col>
    </ion-row>
```

```
  </ion-grid>
</ion-content>
```

2.9.2. View-recorded-workouts.page.ts

```
import { Component, OnInit, OnDestroy } from '@angular/core';
import { RecordedWorkout } from '../../models/recorded_workout.model';
import { Subscription } from 'rxjs';
import { WorkoutService } from '../../services/workout.service';
import { UserService } from 'src/app/services/user.service';
@Component({
  selector: 'app-view-recorded-workouts',
  templateUrl: './view-recorded-workouts.page.html',
  styleUrls: ['./view-recorded-workouts.page.scss'],
})
export class ViewRecordedWorkoutsPage implements OnInit {

  private recordedWorkoutSub: Subscription;

  recordedWorkouts: RecordedWorkout[] = [ ];
  isLoading = false;
  ptMode = false;

  constructor(private workoutServ: WorkoutService,
              private userServ: UserService) { }

  ngOnInit() {
    this.isLoading=true;
    this.recordedWorkoutSub = this.workoutServ.recordedWorkouts.subscribe(reco
rdedWorkouts => {
      this.recordedWorkouts = recordedWorkouts;
      this.isLoading=false;
    });
    this.ptMode = this.userServ.ptMode;
  }

  ionViewWillEnter() {
    this.isLoading=true;
    this.workoutServ.fetchRecordedWorkouts().subscribe(() => {
      this.isLoading=false;
    });
  }

  ngOnDestroy(): void {
    if(this.recordedWorkoutSub){
      this.recordedWorkoutSub.unsubscribe();
    }
  }

  deleteRecordedWorkout(id: string){
    this.workoutServ.deleteRecordedWorkout(id).subscribe();
  }
```

```
}
```

2.10. View-workout-details

2.10.1. View-workout-details.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start"><ion-back-button defaultHref="/view-
workouts"></ion-back-button></ion-buttons>
    <ion-title>{{ isLoading ? 'Loading...' : workout.title }}</ion-title>
    <ion-button slot="end" fill="clear" (click)="toggleEdit()"><ion-
icon color="primary" name="create"></ion-icon></ion-button>
  </ion-toolbar>
</ion-header>

<ion-content>
  <div *ngIf="!isLoading">
  <form [formGroup]="form" (ngSubmit)="editWorkout()">

    <ion-row *ngIf="editMode">
      <ion-col size-sm="6" offset-sm="3">
        <ion-item>
          <ion-label position="floating">Name</ion-label>
          <ion-
input [readonly]="!editMode" type="text" required="true" autocomplete autocorr
ect formControlName="title"></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 [readonly]="!editMode" type="text" required="true" autocomplete autocorr
ect formControlName="description"></ion-input>
        </ion-item>
      </ion-col>
    </ion-row>

    <ion-row>
      <ion-col size-sm="6" offset-sm="3">
        <ion-row>
          <ion-col size="6">
            <div formArrayName="exercises" *ngFor="let exercise of exercises.c
ontrols; let i=index">
              <ion-item>
                <ion-label position="floating">Exercises</ion-label>
```

```
                <ion-
select [formControlName]="i" okText="Okay" cancelText="Cancel" required="true"
>
                  <ion-select-option value="Bench Press">Bench Press</ion-
select-option>
                  <ion-select-option value="Squat">Squat</ion-select-option>
                  <ion-select-option value="Deadlift">Deadlift</ion-select-
option>
                  <ion-select-option value="Pull Ups">Pull Ups</ion-select-
option>
                  <ion-select-
option value="Shoulder Press">Shoulder Press</ion-select-option>
                  <ion-select-option value="Dumbell Curls">Dumbell Curls</ion-
select-option>
                  <ion-select-option value="Barbell Curls">Barbell Curls</ion-
select-option>
                  <ion-select-option value="Leg Curls">Leg Curls</ion-select-
option>
                  <ion-select-option value="Pec Flies">Pec Flies</ion-select-
option>
                  <ion-select-option value="Chest Press">Chest Press</ion-
select-option>
                  <ion-select-
option value="Incline Barbell Press">Incline Barbell Press</ion-select-option>
                  <ion-select-option value="Barbell Row">Barbell Row</ion-
select-option>
                  <ion-select-option value="Penlay Row">Penlay Row</ion-
select-option>
                  <ion-select-
option value="Leg Extensions">Leg Extensions</ion-select-option>
                  <ion-select-
option value="Block Pull Deadlifts">Block Pull Deadlifts</ion-select-option>
                  <ion-select-
option value="Deficit Deadlifts">Deficit Deadlifts</ion-select-option>
                  <ion-select-option value="Chin Ups">Chin Ups</ion-select-
option>
                  <ion-select-option value="Push Ups">Push Ups</ion-select-
option>
                  <ion-select-option value="Sit Ups">Sit Ups</ion-select-
option>
                  <ion-select-
option value="Lateral Pull Down">Lateral Pull Down</ion-select-option>
                  <ion-select-
option value="Tricep Extensions">Tricep Extensions</ion-select-option>
                  <ion-select-option value="Hammer Curls">Hammer Curls</ion-
select-option>
                </ion-select>
              </ion-item>
```

```
            </div>
            </ion-col>
            <ion-col>
              <div formArrayName="sets" *ngFor="let set of sets.controls; let
i=index">
                <ion-item>
                  <ion-label position="floating">Sets</ion-label>
                  <ion-
input [readonly]="!editMode" type="number" required="true" clear-
input [formControlName]="i" expand="full"></ion-input>
                </ion-item>
              </div>
            </ion-col>
            <ion-col>
              <div formArrayName="reps" *ngFor="let rep of reps.controls; let
i=index">
                <ion-item>
                  <ion-label position="floating">Reps</ion-label>
                  <ion-
input [readonly]="!editMode" type="number" required="true" clear-
input [formControlName]="i"></ion-input>
                </ion-item>
              </div>
            </ion-col>
          </ion-row>
        </ion-col>
      </ion-row>
    </form>
  </div>
</ion-content>
<ion-footer *ngIf="editMode">
  <div padding>
    <div>
      <ion-
button color="dark"  size="large" expand="full" [disabled]="exercises.length =
== 1" (click)="removeExercise()">Remove Exercise</ion-button>
    </div>
    <div>
      <ion-
button color="dark"  size="large" expand="full" [disabled]="this.form.value.ex
ercises[this.form.value.exercises.length - 1] === '' || this.form.value.sets[t
his.form.value.sets.length - 1] === '' || this.form.value.reps[this.form.value
.reps.length - 1] === ''" (click)="addExercise()">Add Exercise</ion-button>
    </div>
    <div>
      <ion-
button color="primary"  size="large" (click)="editWorkout()" [disabled]="!form
.valid" expand="full">Update</ion-button>
```

```
    </div>
  </div>
</ion-footer>
```

2.10.2. View-workout-details.page.ts

```
import { Component, OnInit, OnDestroy } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { WorkoutService } from 'src/app/services/workout.service';
import { Workout } from 'src/app/models/workout.model';
import { Subscription } from 'rxjs';
import { FormGroup, FormControl, Validators, FormArray, FormBuilder } from '@a
ngular/forms';
import { HttpClient } from '@angular/common/http';
import { LoadingController } from '@ionic/angular';


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

  workout: Workout;
  workoutSub: Subscription;
  isLoading = false;
  editMode = false;
  form: FormGroup;


  constructor(private route: ActivatedRoute,
              private router: Router,
              private workoutServ: WorkoutService,
              private http: HttpClient,
              private fb: FormBuilder,
              private loadingCtrl: LoadingController) { }

  ngOnInit() {
    this.route.paramMap.subscribe(paramMap => {
      if(!paramMap.has('workoutId')){
        this.router.navigateByUrl('/view-workouts');
        return;
      }
      this.isLoading = true;
      this.workoutSub = this.workoutServ.getWorkout(paramMap.get('workoutId'))
.subscribe(workout => {
        this.workout = workout;
        this.setForm();
        this.isLoading = false;
      });
    });
  }
```

```
setForm() {
  this.form = new FormGroup({
    title: new FormControl(null, {
      updateOn: 'blur',
      validators: [Validators.required]
    }),
    description: new FormControl(null, {
      updateOn: 'blur',
      validators: [Validators.required]
    }),
    exercises: new FormControl(null, {
      updateOn: 'blur',
      validators: [Validators.required]
    }),
    sets: new FormControl(null, {
      updateOn: 'blur',
      validators: [Validators.required]
    }),
    reps: new FormControl(null, {
      updateOn: 'blur',
      validators: [Validators.required]
    }),
  });

  this.form = this.fb.group({
    title: this.workout.title,
    description: this.workout.description,
    exercise: [''],
    exercises: this.fb.array(this.workout.exercises),
    set: [''],
    sets: this.fb.array(this.workout.sets),
    rep: [''],
    reps: this.fb.array(this.workout.reps)
  });
}

toggleEdit() {
  if(this.editMode){
    this.editMode=false;
  } else {
    this.editMode=true;
  }
}


get exercises() {
  return this.form.get('exercises') as FormArray;
}
```

```
get sets() {
  return this.form.get('sets') as FormArray;
}

get reps() {
  return this.form.get('reps') as FormArray;
}

addExercise() {
  this.exercises.push(this.fb.control(''));
  this.sets.push(this.fb.control(''));
  this.reps.push(this.fb.control(''));
}

removeExercise() {
  if(this.exercises.length > 1){
    this.exercises.removeAt(this.exercises.length - 1);
    this.sets.removeAt(this.sets.length - 1);
    this.reps.removeAt(this.reps.length - 1);
  } else {
    console.log("cant remove last exercise");
  }
}

editWorkout() {
  const newWorkout = new Workout(
    this.workout.id,
    this.form.value.title,
    this.form.value.description,
    this.form.value.exercises,
    this.form.value.sets,
    this.form.value.reps,
  );
  console.log(newWorkout);
  this.loadingCtrl
      .create({ keyboardClose: true, message: 'Updating Workout...' })
      .then(loadingEl => {
        loadingEl.present();
        this.workoutServ.editWorkout(
          newWorkout
        ).subscribe(() => {
          this.isLoading = false;
          loadingEl.dismiss();
          this.router.navigateByUrl('/view-workouts');
        });
      });
}
```

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

}
```

2.11. View-workouts

2.11.1. View-workouts.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-buttons slot="start"><ion-back-button defaultHref="/dashboard"></ion-
back-button></ion-buttons>
    <ion-title>Workouts</ion-title>
    <ion-button slot="end" fill="clear" routerLink="/create-workout"><ion-
icon color="primary" name="add"></ion-icon></ion-button>
  </ion-toolbar>
</ion-header>

<ion-content>
  <p *ngIf="isLoading" class="ion-text-center"><ion-
spinner color="primary"></ion-spinner></p>
  <ion-grid *ngIf="!isLoading">
    <ion-row>
      <ion-col size-sm="6" offset-sm="3">
        <div *ngIf="workouts.length === 0" text-center>
          <ion-item padding-bottom padding-end>
            <ion-label text-center class="ion-text-wrap">
              <h1>Add a Workout</h1>
            </ion-label>
          </ion-item>
        </div>
        <ion-list>
          <div *ngFor="let workout of workouts">
            <ion-item-sliding padding-bottom padding-end>
              <ion-item [routerLink]="['/','view-workouts', workout.id]">
                  <ion-label class="ion-text-wrap">
                    <h1>{{ workout.title }}</h1>
                  </ion-label>
                  <ion-icon slot="end" md="md-arrow-back"></ion-icon>
              </ion-item>

              <ion-item-options side="end" padding-bottom padding-end>
                <ion-item-
option color="danger" (click)="deleteWorkout(workout.id)">Delete</ion-item-
option>
              </ion-item-options>
            </ion-item-sliding>
          </div>
        </ion-list>
      </ion-col>
    </ion-row>
  </ion-grid>
</ion-content>
```

2.11.2. View-workouts.page.ts

```
import { Component, OnInit, OnDestroy } from '@angular/core';
import { Workout } from '../../models/workout.model';
import { Subscription } from 'rxjs';
import { WorkoutService } from '../../services/workout.service';

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

  private workoutSub: Subscription;

  workouts: Workout[] = [ ];
  isLoading = false;

  constructor(private workoutServ: WorkoutService) { }

  ngOnInit() {
    this.isLoading=true;
    this.workoutSub = this.workoutServ.workouts.subscribe(workouts => {
      this.workouts = workouts;
      this.isLoading=false;
    });
  }

  ionViewWillEnter() {
    this.isLoading=true;
    this.workoutServ.fetchWorkouts().subscribe(() => {
      this.isLoading=false;
    });
  }

  ngOnDestroy(): void {
    if(this.workoutSub){
      this.workoutSub.unsubscribe();
    }
  }

  deleteWorkout(id: string){
    this.workoutServ.deleteWorkout(id).subscribe();
  }

}
```

3. Public

3.1. Forgot-password

3.1.1. Forgot-password.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-title>Revolute Fitness</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <form [formGroup]="form" (ngSubmit)="changePassword()">
    <ion-grid padding-top>
      <ion-row justify-content-center>
        <ion-col align-self-center size-md="8" size-lg="7" size-xs="12">
          <div text-center>
            <h1>Forgot Password</h1>
          </div>
          <div padding>
            <ion-item no-padding>
              <ion-label position="floating">Accounts Email</ion-label>
            <ion-input type="email" placeholder="your@email.com" clear-
input autocomplete="off" autofocus formControlName="email" required email></io
n-input>
            </ion-item>
          </div>
        </ion-col>
      </ion-row>
    </ion-grid>
  </form>
</ion-content>
<ion-footer>
  <div padding>
    <div>
      <ion-
button color="dark"  size="large" (click)="changePassword()" [disabled]="!form
.valid" expand="full">Submit</ion-button>
    </div>
    <div>
      <ion-button size="large" routerLink='/login' expand="full">Back</ion-
button>
    </div>
  </div>
</ion-footer>
```

3.1.2. Forgot-password.page.ts

```
import { Component, OnInit } from '@angular/core';
import { UserService } from 'src/app/services/user.service';
import { FormBuilder, FormGroup, FormControl, Validators } from '@angular/form
s';
import { Router } from '@angular/router';

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

  constructor(
    private authService: UserService,
    private fb: FormBuilder,
    private router: Router) { }

  form: FormGroup;
  isLoading = false;

  ngOnInit() {
    this.form = new FormGroup({
      email: new FormControl(null, {
        updateOn: 'blur',
        validators: [Validators.required]
      })
    });
    this.form = this.fb.group({
      email: ['']
    });
  }

  async changePassword() {
    await this.authService.changePasswordRequest(this.form.value.email);
    this.router.navigateByUrl('/login');
  }
}
```

3.2. Login

3.2.1. Login.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-title>Revolute Fitness</ion-title>
  </ion-toolbar>
```

```
</ion-header>

<ion-content>
  <form [formGroup]="form" (ngSubmit)="login()">
    <ion-grid padding-top>
      <ion-row justify-content-center>
        <ion-col align-self-center size-md="8" size-lg="7" size-xs="12">
          <div text-center>
            <h1>Login</h1>
          </div>
          <div padding>
            <ion-item no-padding>
              <ion-label position="floating">Email</ion-label>
            <ion-input type="email" placeholder="your@email.com" clear-
input autocomplete="off" autofocus formControlName="email" required email></io
n-input>
            </ion-item>
            <ion-item no-padding>
              <ion-label position="floating">Password</ion-label>
              <ion-input type="password" placeholder="Password" clear-
input clear-on-
edit formControlName="password" required password minLength="6"></ion-input>
            </ion-item>
          </div>
          <div text-center>
            <a href="/forgot-password">Forgot Password</a>
          </div>
          <div padding>
            <ion-label color="danger">{{ errorMessage }}</ion-label>
          </div>
        </ion-col>
      </ion-row>
    </ion-grid>
  </form>
</ion-content>
<ion-footer>
        <div padding>
          <div>
            <ion-
button color="dark"  size="large" (click)="login()" [disabled]="!form.valid" e
xpand="full">Login</ion-button>
          </div>
          <div>
            <ion-
button size="large" routerLink='/register' expand="full">Register</ion-button>
          </div>
        </div>
</ion-footer>
```

3.2.2. Login.page.ts

```
import { Component, OnInit } from '@angular/core';
import { AngularFireAuth } from '@angular/fire/auth';
import { Router } from '@angular/router';
import { NgForm, FormGroup, FormBuilder, FormControl, Validators } from '@angu
lar/forms';
import { LoadingController } from '@ionic/angular';
import { UserService } from '../../services/user.service';


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

  constructor(
    private authService: UserService,
    private fb: FormBuilder,) { }

  form: FormGroup;
  isLoading = false;
  errorMessage;
  isKeyboardHide: boolean;

  ngOnInit() {
    this.form = new FormGroup({
      email: new FormControl(null, {
        updateOn: 'blur',
        validators: [Validators.required]
      }),
      password: new FormControl(null, {
        updateOn: 'blur',
        validators: [Validators.required]
      }),
    });
    this.form = this.fb.group({
      email: [''],
      password: ['']
    });
  }

  async login() {
    await this.authService.login(this.form.value.email, this.form.value.passwo
rd);
    this.errorMessage = this.authService.loginErrorMessage;
  }
```

```
}
```

3.3. Register

3.3.1. Register.page.html

```
<ion-header>
  <ion-toolbar>
    <ion-title>Revolute Fitness</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <form [formGroup]="form" (ngSubmit)="register()">
    <ion-grid>
      <ion-row color="dark" justify-content-center>
        <ion-col align-self-center size-md="8" size-lg="7" size-xs="12">
          <div text-center>
            <h3>Register Your Account</h3>
          </div>
          <div padding>
            <ion-item no-padding>
              <ion-label position="floating">Full Name</ion-label>
              <ion-
input  name="name" type="text" placeholder="John Doe" clear-
input autofocus autocapitalize="on" formControlName="name" required="true"></i
on-input>
            </ion-item>
            <ion-item no-padding>
              <ion-label position="floating">Email</ion-label>
              <ion-
input name="email" type="email" placeholder="your@email.com" clear-
input formControlName="email" required="true"></ion-input>
            </ion-item>
            <ion-item no-padding>
              <ion-label position="floating">Password</ion-label>
              <ion-
input name="password" type="password" placeholder="*********" clear-on-
edit clear-input formControlName="password" required ="true"></ion-input>
            </ion-item>
            <ion-item no-padding>
              <ion-label position="floating">Height</ion-label>
              <ion-
input name="height" type="number" placeholder="180cm" clear-
input formControlName="height" required="true"></ion-input>
            </ion-item>
            <ion-item no-padding>
              <ion-label position="floating">Weight</ion-label>
              <ion-input name="weight" type="number" placeholder="70kg" clear-
input formControlName="weight" required="true"></ion-input>
            </ion-item>
            <ion-item no-padding>
```

```
              <ion-label position="floating">User Type</ion-label>
              <ion-
select value="general_user" formControlName="userType" required="true" okText=
"Okay" cancelText="Cancel" >
                <ion-select-option value="general_user">General User</ion-
select-option>
                <ion-select-
option value="personal_trainer">Personal Trainer</ion-select-option>
                <ion-select-option value="client">Client</ion-select-option>
              </ion-select>
            </ion-item>
          </div>
          <div padding>
            <ion-label color="danger">{{ errorMessage }}</ion-label>
          </div>
        </ion-col>
      </ion-row>
    </ion-grid>
  </form>
</ion-content>
<ion-footer>
  <div padding>
    <div>
      <ion-
button color="dark"  size="large" (click)="register()" [disabled]="!form.valid
" expand="full">Register</ion-button>
    </div>
    <div>
      <ion-button size="large" routerLink='/login' expand="full">Back</ion-
button>
    </div>
  </div>
</ion-footer>
```

3.3.2. Register.page.ts

```
import { Component, OnInit } from '@angular/core';
import { AngularFireAuth } from '@angular/fire/auth';
import { auth } from 'firebase/app';
import { NavController } from '@ionic/angular';
import { UserService } from '../../services/user.service';
import { NgForm, FormGroup, FormBuilder, FormControl, Validators } from '@angu
lar/forms';
import { User } from '../../models/user.model';
@Component({
  selector: 'app-register',
  templateUrl: './register.page.html',
  styleUrls: ['./register.page.scss'],
})
export class RegisterPage implements OnInit {

  public errorMessage: string;
  form: FormGroup;
  isLoading = false;

  constructor(public afAuth: AngularFireAuth,
              public nav: NavController,
              private regService: UserService,
              private fb: FormBuilder,) { }

  ngOnInit() {
    this.form = new FormGroup({
      name: new FormControl(null, {
        updateOn: 'blur',
        validators: [Validators.required]
      }),
      email: new FormControl(null, {
        updateOn: 'blur',
        validators: [Validators.required]
      }),
      password: new FormControl(null, {
        updateOn: 'blur',
        validators: [Validators.required]
      }),
      height: new FormControl(null, {
        updateOn: 'blur',
        validators: [Validators.required]
      }),
      weight: new FormControl(null, {
        updateOn: 'blur',
        validators: [Validators.required]
      }),
      userType: new FormControl(null, {
```

```
        updateOn: 'blur',
        validators: [Validators.required]
      }),
    });
    this.form = this.fb.group({
      name: [''],
      email: [''],
      password: [''],
      height: [''],
      weight: [''],
      userType: ['general_user'],
    });
  }

  async register() {
    const user = new User(
      '',
      this.form.value.name,
      this.form.value.email,
      this.form.value.userType,
      this.form.value.weight,
      this.form.value.height,
    );
    await this.regService.register(user, this.form.value.password);
    this.errorMessage = this.regService.registerErrorMessage;
  }



}
```

4. Services

4.1. Login.gaurd.ts

```
import { Injectable } from '@angular/core';
import { CanLoad, Route, UrlSegment, Router } from '@angular/router';
import { Observable } from 'rxjs';

import { UserService } from './user.service';

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

  canLoad(
    route: Route,
    segments: UrlSegment[]
  ): Observable<boolean> | Promise<boolean> | boolean {
    if (!this.authService.userIsAuthenticated) {
      this.router.navigateByUrl('/login');
    }
    return this.authService.userIsAuthenticated;
  }
}
```

4.2. User.service.ts

```
import { Injectable } from '@angular/core';
import { NavController, LoadingController } from '@ionic/angular';
import { AngularFireAuth } from '@angular/fire/auth';
import { Router } from '@angular/router';
import { Message } from '@angular/compiler/src/i18n/i18n_ast';
import { User } from '../models/user.model';
import { HttpClient } from '@angular/common/http';
import { map, tap, take } from 'rxjs/operators';
import { BehaviorSubject, Subscription } from 'rxjs';
import * as firebase from 'firebase';

interface UserDataInt {
  name: string;
  email: string;
  userType: string;
  bodyWeigth: number;
  height: number;
}

@Injectable({
  providedIn: 'root'
})
export class UserService {

  // Login Stuff
  private userSub: Subscription;
  private user: firebase.User;
  private _userIsAuthenticated = false;
  private _userId: string;
  private _currentUser: User;
  public ptMode = false;
  public _currentClient;
  private isLoading = false;
  private isLogin = true;
  public loginErrorMessage: string;
  public registerErrorMessage: string;

  get userIsAuthenticated() {
    return this._userIsAuthenticated;
  }

  get userId() {
    if (this.ptMode) {
      return this._currentClient;
    } else {
      return this._userId;
    }
```

```
  }

  get currentUser() {
    return this._currentUser;
  }

  get currentUserName() {
    return this._currentUser.name;
  }

  // PT STUFF
  private _users = new BehaviorSubject<User[]>([]);
  private _clientUsers = new BehaviorSubject<User[]>([]);

  get users() {
    return this._users.asObservable();
  }

  get clientUsers() {
    return this._clientUsers.asObservable();
  }

  constructor(public nav: NavController,
              private afAuth: AngularFireAuth,
              private router: Router,
              private loadingCtrl: LoadingController,
              private http: HttpClient) {
                afAuth.authState.subscribe(user => {
                  this.user = user;
                });
              }

  async login(email: string, password: string) {
    this.isLoading = true;
    try {

      firebase.auth().setPersistence(firebase.auth.Auth.Persistence.LOCAL);
      const res = await this.afAuth.auth.signInWithEmailAndPassword(email, pas
sword);

      if (this.user.emailVerified){
        this._userId = this.user.uid;

        this.userSub = this.getUserRecord().subscribe(userReturned => {
          this._currentUser = userReturned;
        });


        this.loadingCtrl
```

```
          .create({ keyboardClose: true, message: 'Logging in...' })
          .then(loadingEl => {
            loadingEl.present();
            setTimeout(() => {
              this.isLoading = false;
              loadingEl.dismiss();
              this.router.navigateByUrl('/dashboard');
            }, 1500);
          });
        this._userIsAuthenticated = true;
      } else {
        this.loginErrorMessage = 'Users email needs to be verified, check your
 inbox !';
      }
    } catch (err) {
      this.loginErrorMessage = err.message;
      this.loadingCtrl
        .create({ keyboardClose: true, message: 'Logging in...' })
        .then(loadingEl => {
          loadingEl.present();
          setTimeout(() => {
            this.isLoading = false;
            loadingEl.dismiss();
          }, 1500);
        });
    }
  }

  async logout() {
    this._userIsAuthenticated = false;
    this._userId = null;
    this._currentUser = null;
    await this.afAuth.auth.signOut();
  }

  async register(userObj: User, password: string) {
    this.isLoading = true;
    try {
      const res = await this.afAuth.auth.createUserWithEmailAndPassword(userOb
j.email, password);
      const user = this.afAuth.auth.currentUser;
      user.updateProfile({
        displayName: name,
      }).then(function() {

      }, function(error) {

      });
```

```
      this.createUserRecord(userObj, user.uid).subscribe();

      user.sendEmailVerification().then( resp => {
        // Email sent.
      }).catch(error => {
        // An error happened.
      });

      this.logout();
      this.loadingCtrl
        .create({ keyboardClose: true, message: 'Registering...' })
        .then(loadingEl => {
          loadingEl.present();
          setTimeout(() => {
            this.isLoading = false;
            this.nav.navigateForward('/login');
            loadingEl.dismiss();
          }, 1500);
        });
    } catch (err) {
      this.registerErrorMessage = err.message;
    }
  }

  createUserRecord(user: User, uid: string) {
    return this.http.put(`https://revolutefitness-
a92df.firebaseio.com/users/${uid}.json` , {
      ...user,
      id: null
    });
  }

  getUserRecord() {
    return this.http
      .get<UserDataInt>(
        `https://revolutefitness-
a92df.firebaseio.com/users/${this._userId}.json`
      )
      .pipe(
        map(userData => {
          return new User(
            this.userId,
            userData.name,
            userData.email,
            userData.userType,
            userData.bodyWeigth,
            userData.height,
```

```
          )
          }
        )
      );
  }

  async changePasswordRequest(email: string) {
    await this.afAuth.auth.sendPasswordResetEmail(email);
  }

  // PT Client Stuff
  fetchUsers() {
    return this.http
      .get<{ [key: string]: UserDataInt }>(
        `https://revolutefitness-a92df.firebaseio.com/users.json`
      )
      .pipe(
        map(UserDataInt => {
          const users = [];
          for (const key in UserDataInt) {
            if (UserDataInt.hasOwnProperty(key) && UserDataInt[key].userType =
== 'client') {
              users.push(
                new User(
                  key,
                  UserDataInt[key].name,
                  UserDataInt[key].email,
                  UserDataInt[key].userType,
                  UserDataInt[key].bodyWeigth,
                  UserDataInt[key].height
                )
              );

            }
          }
          return users;
        }),
        tap(users => {
          this._users.next(users);
        })
      );
  }

  fetchClientUsers() {
    return this.http
      .get<{ [key: string]: UserDataInt }>(
        `https://revolutefitness-
a92df.firebaseio.com/personal_trainers/${this.currentUser.id}/clients.json`
```

```
      )
      .pipe(
        map(UserDataInt => {
          const users = [];
          for (const key in UserDataInt) {
            if (UserDataInt.hasOwnProperty(key)) {
              users.push(
                new User(
                  key,
                  UserDataInt[key].name,
                  UserDataInt[key].email,
                  UserDataInt[key].userType,
                  UserDataInt[key].bodyWeigth,
                  UserDataInt[key].height
                )
              );

            }
          }
          return users;
        }),
        tap(users => {
          this._clientUsers.next(users);
        })
      );
  }

  addClient(user: User) {
    return this.http.put(`https://revolutefitness-
a92df.firebaseio.com/personal_trainers/${this.userId}/clients/${user.id}.json`
 , {
      ...user,
    });
  }

  deleteClient(id: string) {
    return this.http
      .delete(
        `https://revolutefitness-
a92df.firebaseio.com/personal_trainers/${this.userId}/clients/${id}.json`
      );
  }
}
```

4.3. Workout.service.ts

```
import { Injectable } from '@angular/core';
import { NavController, LoadingController } from '@ionic/angular';
import { AngularFireAuth } from '@angular/fire/auth';
import { Router } from '@angular/router';
import { Message } from '@angular/compiler/src/i18n/i18n_ast';
import { Workout } from '../models/workout.model';
import { RecordedWorkout } from '../models/recorded_workout.model';
import { HttpClient } from '@angular/common/http';
import { BehaviorSubject } from 'rxjs';
import { map, tap, switchMap, take } from 'rxjs/operators';
import { UserService } from './user.service';

interface WorkoutDataInt {
  title: string;
  description: string;
  id: string;
  exercises: string[];
  sets: number[];
  reps: number[];
}

interface RecordedWorkoutDataInt {
  id: string;
  title: string;
  date: Date;
  notes: string;
  exercises: string[];
  weights: number[];
}

@Injectable({
  providedIn: 'root'
})
export class WorkoutService {

  private _workouts = new BehaviorSubject<Workout[]>([]);
  private _recordedWorkouts = new BehaviorSubject<RecordedWorkout[]>([]);

  constructor(private http: HttpClient, private loginServ: UserService) {}

  get workouts() {
    return this._workouts.asObservable();
  }

  getWorkout(id: string) {
    return this.http
      .get<WorkoutDataInt>(
```

```
        `https://revolutefitness-
a92df.firebaseio.com/workouts/${this.loginServ.userId}/${id}.json`
      )
      .pipe(
        map(workoutData => {
          return new Workout(
            id,
            workoutData.title,
            workoutData.description,
            workoutData.exercises,
            workoutData.sets,
            workoutData.reps
          );
        })
      );
  }

  editWorkout(workout: Workout){
    let generatedId: string;
    return this.http.put<{name: string}>(`https://revolutefitness-
a92df.firebaseio.com/workouts/${this.loginServ.userId}/${workout.id}.json` ,{
      ...workout,
      id: null
    })
    .pipe(
      switchMap(resData => {
        generatedId = resData.name;
        return this.workouts;
      }),
      take(1),
      tap(workouts => {
        workout.id = generatedId;
        this._workouts.next(workouts.concat(workout));
      })
    )
  }

  createWorkout(workout: Workout){
    let generatedId: string;
    return this.http.post<{name: string}>(`https://revolutefitness-
a92df.firebaseio.com/workouts/${this.loginServ.userId}.json`, {
      ...workout,
      id: null
    })
    .pipe(
      switchMap(resData => {
        generatedId = resData.name;
        return this.workouts;
```

```
      }),
      take(1),
      tap(workouts => {
        workout.id = generatedId;
        this._workouts.next(workouts.concat(workout));
      })
    );
  }

  fetchWorkouts(){
    return this.http
      .get<{ [key: string]: WorkoutDataInt }>(
        `https://revolutefitness-
a92df.firebaseio.com/workouts/${this.loginServ.userId}.json`
      )
      .pipe(
        map(WorkoutDataInt => {
          const workouts = [];
          for (const key in WorkoutDataInt) {
            if (WorkoutDataInt.hasOwnProperty(key)) {
              workouts.push(
                new Workout(
                  key,
                  WorkoutDataInt[key].title,
                  WorkoutDataInt[key].description,
                  WorkoutDataInt[key].exercises,
                  WorkoutDataInt[key].sets,
                  WorkoutDataInt[key].reps
                )
              );

            }
          }
          return workouts;
        }),
        tap(workouts => {
          this._workouts.next(workouts);
        })
      );
  }

  deleteWorkout(id: string) {
    return this.http
      .delete(
        `https://revolutefitness-
a92df.firebaseio.com/workouts/${this.loginServ.userId}/${id}.json`
      )
      .pipe(
```

```
        switchMap(() => {
          return this.workouts;
        }),
        take(1),
        tap(workouts => {
          this._workouts.next(workouts.filter(b => b.id !== id));
        })
      );
  }

  // RECORDED WORKOUTS

  get recordedWorkouts() {
    return this._recordedWorkouts.asObservable();
  }

  recordWorkout(recordedWorkout: RecordedWorkout) {
    let generatedId: string;
    return this.http.post<{name: string}>(`https://revolutefitness-
a92df.firebaseio.com/recorded_workouts/${this.loginServ.userId}.json` ,{
      ...recordedWorkout,
      id: null
    })
    .pipe(
      switchMap(resData => {
        generatedId = resData.name;
        return this.recordedWorkouts;
      }),
      take(1),
      tap(recordedWorkouts => {
        recordedWorkout.id = generatedId;
        this._recordedWorkouts.next(recordedWorkouts.concat(recordedWorkout));
      })
    )
  }

  getRecordedWorkout(id: string) {
    return this.http
      .get<RecordedWorkoutDataInt>(
        `https://revolutefitness-
a92df.firebaseio.com/recorded_workouts/${this.loginServ.userId}/${id}.json`
      )
      .pipe(
        map(recordedWorkoutData => {
          return new RecordedWorkout(
            id,
            recordedWorkoutData.title,
            recordedWorkoutData.date,
```

```
            recordedWorkoutData.notes,
            recordedWorkoutData.exercises,
            recordedWorkoutData.weights
          );
        }
        )
      );
  }

  fetchRecordedWorkouts() {
    return this.http
      .get<{ [key: string]: RecordedWorkoutDataInt }>(
        `https://revolutefitness-
a92df.firebaseio.com/recorded_workouts/${this.loginServ.userId}.json`
      )
      .pipe(
        map(recordedWorkoutData => {
          const recordedWorkouts = [];
          for (const key in recordedWorkoutData) {
            if (recordedWorkoutData.hasOwnProperty(key)) {
              recordedWorkouts.push(
                new RecordedWorkout(
                  key,
                  recordedWorkoutData[key].title,
                  recordedWorkoutData[key].date,
                  recordedWorkoutData[key].notes,
                  recordedWorkoutData[key].exercises,
                  recordedWorkoutData[key].weights
                )
              );

            }
          }
          return recordedWorkouts;
        }),
        tap(recordedWorkouts => {
          this._recordedWorkouts.next(recordedWorkouts);
        })
      );
  }

  deleteRecordedWorkout(id: string) {
    return this.http
      .delete(
        `https://revolutefitness-
a92df.firebaseio.com/recorded_workouts/${this.loginServ.userId}/${id}.json`
      )
      .pipe(
```

```
        switchMap(() => {
          return this.recordedWorkouts;
        }),
        take(1),
        tap(recordedWorkouts => {
          this._recordedWorkouts.next(recordedWorkouts.filter(b => b.id !== id
));
        })
      );
  }
}
```

4.4. App.routing.module.ts

```
import { NgModule } from '@angular/core';
import { PreloadAllModules, RouterModule, Routes } from '@angular/router';
import { AuthGuard } from './services/login.guard';

const routes: Routes = [
  { path: '', redirectTo: 'login', pathMatch: 'full' },
  {
    path: 'login',
    loadChildren: () => import('./public/login/login.module').then( m => m.Log
inPageModule)
  },
  {
    path: 'register',
    loadChildren: () => import('./public/register/register.module').then( m =>
 m.RegisterPageModule)
  },
  {
    path: 'dashboard',
    loadChildren: () => import('./private/dashboard/dashboard.module').then( m
 => m.DashboardPageModule),
    canLoad: [AuthGuard]
  },
  {
    path: 'view-workouts',
    loadChildren: () => import('./private/view-workouts/view-
workouts.module').then( m => m.ViewWorkoutsPageModule),
    canLoad: [AuthGuard]
  },
  {
    path: 'record-workouts-view/:workoutId',
    loadChildren: () => import('./private/record-workout/record-
workout.module').then( m => m.RecordWorkoutPageModule),
    canLoad: [AuthGuard]
  },
  {
    path: 'create-workout',
    loadChildren: () => import('./private/create-workout/create-
workout.module').then( m => m.CreateWorkoutPageModule),
    canLoad: [AuthGuard]
  },
  {
    path: 'view-workouts/:workoutId',
    loadChildren: () => import('./private/view-workout-details/view-workout-
details.module').then( m => m.ViewWorkoutDetailsPageModule),
    canLoad: [AuthGuard]
  },
  {
```

```
    path: 'manage-clients',
    loadChildren: () => import('./private/manage-clients/manage-
clients.module').then( m => m.ManageClientsPageModule),
    canLoad: [AuthGuard]
  },
  {
    path: 'record-workouts-view',
    loadChildren: () => import('./private/record-workouts-view/record-
workouts-view.module').then( m => m.RecordWorkoutsViewPageModule),
    canLoad: [AuthGuard]
  },
  {
    path: 'exercise-analysis',
    loadChildren: () => import('./private/exercise-analysis/exercise-
analysis.module').then( m => m.ExerciseAnalysisPageModule),
    canLoad: [AuthGuard]
  },
  {
    path: 'view-recorded-workouts',
    loadChildren: () => import('./private/view-recorded-workouts/view-
recorded-workouts.module').then( m => m.ViewRecordedWorkoutsPageModule),
    canLoad: [AuthGuard]
  },
  {
    path: 'view-recorded-workouts/:recordedWorkoutId',
    loadChildren: () => import('./private/view-recorded-workout-details/view-
recorded-workout-
details.module').then( m => m.ViewRecordedWorkoutDetailsPageModule),
    canLoad: [AuthGuard]
  },
  {
    path: 'user-stats',
    loadChildren: () => import('./private/user-stats/user-
stats.module').then( m => m.UserStatsPageModule),
    canLoad: [AuthGuard]
  },
  {
    path: 'manage-client-dashboard',
    loadChildren: () => import('./private/manage-client-dashboard/manage-
client-dashboard.module').then( m => m.ManageClientDashboardPageModule),
    canLoad: [AuthGuard]
  },
  {
    path: 'forgot-password',
    loadChildren: () => import('./public/forgot-password/forgot-
password.module').then( m => m.ForgotPasswordPageModule)
  }
```

```
];

@NgModule({
  imports: [
    RouterModule.forRoot(routes, { preloadingStrategy: PreloadAllModules })
  ],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

5. Machine Learning

5.1. Model Creation

```
from __future__ import absolute_import, division, print_function, unicode_literals

# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D
import pickle

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt
import cv2
import os

print(tf.__version__)


DATADIR = "C:/Users/kevin/Documents/Ionic/FYP_Applications/ML_Model_Creation/exercises"
CATEGORIES = ["Squat", "Bench"]

for category in CATEGORIES:
    path = os.path.join(DATADIR, category)
    for img in os.listdir(path):
        img_array = cv2.imread(os.path.join(path,img), cv2.IMREAD_GRAYSCALE)
        plt.imshow(img_array, cmap="gray")
        plt.show()
        break
    break


IMG_SIZE = 150

new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
plt.imshow(new_array, cmap = 'gray')
plt.show()


training_data = []

def create_training_data():
    for category in CATEGORIES:
        path = os.path.join(DATADIR, category)
        class_num = CATEGORIES.index(category)
```

```
        for img in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path,img), cv2.IMREAD_GRAY
SCALE)
                new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
                training_data.append([new_array, class_num])
            except Exception as e:
                pass

create_training_data()

import random

random.shuffle(training_data)


X = []
y = []


for features, label in training_data:
    X.append(features)
    y.append(label)

X = np.array(X).reshape(-1, IMG_SIZE, IMG_SIZE, 1)
y = np.array(y)


import pickle

pickle_out = open("X.pickle", "wb")
pickle.dump(X, pickle_out)
pickle_out.close()

pickle_out = open("y.pickle", "wb")
pickle.dump(y, pickle_out)
pickle_out.close()


pickle_in = open("X.pickle", "rb")
X = pickle.load(pickle_in)
pickle_in.close()

pickle_in = open("y.pickle", "rb")
y = pickle.load(pickle_in)
pickle_in.close()


X = X / 255.0

model = Sequential()
```

```
model.add(Conv2D(64, (3,3), input_shape = X.shape[1:]))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(64, (3,3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())
model.add(Dense(64))
model.add(Activation("relu"))

model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy
"])

model.fit(X, y, batch_size=1,epochs=10, validation_split=0.05)


def prepare(filepath):
    img_array = cv2.imread(filepath, cv2.IMREAD_GRAYSCALE)
    new_array = cv2.resize(img_array, (IMG_SIZE,IMG_SIZE))
    return new_array.reshape(-1,IMG_SIZE,IMG_SIZE,1)

prediction = model.predict([prepare('C:/Users/kevin/Documents/Ionic/FYP_Applic
ations/ML_Model_Creation/exercises/predict4.jpg')])

print(CATEGORIES[int(prediction[0][0])])


model.save('my_model.h5')

DATADIR = "C:/Users/kevin/Documents/Ionic/exercises"
CATEGORIES2 = ["Squat_TEST", "Bench_TEST"]

for category in CATEGORIES2:
    path = os.path.join(DATADIR, category)
    for img in os.listdir(path):
        img_array = cv2.imread(os.path.join(path,img), cv2.IMREAD_GRAYSCALE)
        plt.imshow(img_array, cmap="gray")
        plt.show()
        break
    break
IMG_SIZE = 150

new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
```

```
plt.imshow(new_array, cmap = 'gray')
plt.show()

testing_data = []

def create_testing_data():
    for category in CATEGORIES2:
        path = os.path.join(DATADIR, category)
        class_num = CATEGORIES2.index(category)
        for img in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path,img), cv2.IMREAD_GRAY
SCALE)
                new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
                testing_data.append([new_array, class_num])
            except Exception as e:
                pass

create_testing_data()

random.shuffle(testing_data)

X_test = []
y_test = []
for features, label in training_data:
    X_test.append(features)
    y_test.append(label)

X_test = np.array(X).reshape(-1, IMG_SIZE, IMG_SIZE, 1)
y_test = np.array(y)

X_test = X_test / 255.0

from sklearn.metrics import confusion_matrix
y_pred = model.predict_classes(X_test)

confusion_matrix(y_test, y_pred)
```

5.2. Flask App

5.2.1. App.py

```
from flask import Flask, render_template, url_for, flash, redirect, request

# TensorFlow and tf.keras
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2
D, MaxPooling2D
import pickle

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt
import cv2
import os


app = Flask(__name__)
app.debug = True

@app.route("/", methods=['POST','GET'])
@app.route("/home", methods=['POST','GET'])
def home(name=None):
    if request.args:
        result = request.args['result']
    else:
        result = None
    return render_template('home.html', predictions=result)


app.config['IMAGE_UPLOADS'] = "C:/Users/kevin/Documents/Ionic/Revolute_Fitness
/ML_Flask_App/static/img"


@app.route("/predict/", methods=['POST','GET'])
def runPrediction():
    if request.method == 'POST':

        if request.files:

            image = request.files['image']

            image.save(os.path.join(app.config['IMAGE_UPLOADS'], 'predict.jpg'
))
```

```
            result = predict(os.path.join(app.config['IMAGE_UPLOADS'], 'predic
t.jpg'))

            os.remove(os.path.join(app.config['IMAGE_UPLOADS'], 'predict.jpg')
)

            return redirect(url_for('.home', result=result))

    return redirect('/home')




#ML STUFF
def prepare(filepath):
    IMG_SIZE = 150
    img_array = cv2.imread(filepath, cv2.IMREAD_GRAYSCALE)
    new_array = cv2.resize(img_array, (IMG_SIZE,IMG_SIZE))
    new_array = new_array.astype(np.float64)
    return new_array.reshape(-1,IMG_SIZE,IMG_SIZE,1)



def predict(filepath):
    model = tf.keras.models.load_model('my_model.h5')
    prediction = model.predict([prepare(filepath)])
    CATEGORIES = ["Barbell Squat", "Barbell Bench Press"]
    result = CATEGORIES[int(prediction[0][0])]
    print(result)
    return result



if __name__ == '__main__':
    app.run(debug=True)
```

5.2.2. Home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1
/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.j
s"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.mi
n.js"></script>
  <script>
    function readURL(input) {
        if (input.files && input.files[0]) {
            var reader = new FileReader();

            reader.onload = function (e) {
                $('#blah')
                    .attr('src', e.target.result)
                    .width(150)
                    .height(200);
            };

            reader.readAsDataURL(input.files[0]);
        }
    }
  </script>
</head>
<body>

<div class="jumbotron text-center">
  <h1>Exercise Image Analysis</h1>
  <p>Select an image of a bench press or a squat and the application will clas
sify the image</p>
</div>

<div class="container">
  <form action="/predict/" method="post" enctype="multipart/form-data">
    <div class="row">
      <div class="col-sm-4">
        <h3>Select Prediction Image</h3>
        <input name='image' id='image' class="form-control border-
0" type="file" onchange="readURL(this)">
        <button type="submit" class="btn btn-primary">Predict</button>
      </div>
      <div class="col-sm-4">
```

```
        <h3>Selected Image</h3>
        <img id="blah" src="#"/>
      </div>
      <div class="col-sm-4">
        <h3>Prediction</h3>
        <h4><b>{{ predictions }}</b></h4>
      </div>
    </div>
  </form>
</div>

</body>
</html>
```

6. Appendix

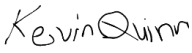
6.1. Declaration

- I declare that all material in this submission e.g. thesis/essay/project/assignment is entirely my/our own work except where duly acknowledged.
- I have cited the sources of all quotations, paraphrases, summaries of information, tables, diagrams or other material; including software and other electronic media in which intellectual property rights may reside.
- I have provided a complete bibliography of all works and sources used in the preparation of this submission.
- I understand that failure to comply with the Institute's regulations governing plagiarism constitutes a serious offense.

Student Name: Kevin Quinn

Student Number: C00216607

Signature:



Date: 20/04/2020