FITNESS FULLSTACK APPLICATION-FRONT-END

Components

a) Register

RegisterComponent.html

```
<div class="col-md-12">
  <div class="card card-container">
    <img
      id="profile-img"
      src="//ssl.gstatic.com/accounts/ui/avatar_2x.png"
      class="profile-img-card"
    />
    <form
      *ngIf="!isSuccessful"
      name="form"
      (ngSubmit)="f.form.valid && onSubmit()"
      #f="ngForm"
      novalidate
    >
      <div class="form-group">
        <label for="username">Username</label>
        <input
          type="text"
          class="form-control"
          name="username"
          [(ngModel)]="form.username"
          required
          minlength="3"
          maxlength="20"
          #username="ngModel"
        />
        <div class="alert-danger" *ngIf="username.errors && f.submitted">
          <div *ngIf="username.errors['required']">Username is required</div>
          <div *ngIf="username.errors['minlength']">
            Username must be at least 3 characters
          </div>
          <div *ngIf="username.errors['maxlength']">
            Username must be at most 20 characters
          </div>
        </div>
      </div>
      <div class="form-group">
        <label for="email">Email</label>
        <input
          type="email"
          class="form-control"
          name="email"
          [(ngModel)]="form.email"
          required
          email
          #email="ngModel"
        />
        <div class="alert-danger" *ngIf="email.errors && f.submitted">
          <div *ngIf="email.errors['required']">Email is required</div>
          <div *ngIf="email.errors['email']">
            Email must be a valid email address
          </div>
        </div>
      </div>
      <div class="form-group">
        <label for="password">Password</label>
        <input
          type="password"
          class="form-control"
          name="password"
          [(ngModel)]="form.password"
          required
          minlength="6"
          #password="ngModel"
        />
        <div class="alert-danger" *ngIf="password.errors && f.submitted">
          <div *ngIf="password.errors['required']">Password is required</div>
          <div *ngIf="password.errors['minlength']">
            Password must be at least 6 characters
          </div>
        </div>
      </div>
      <div class="form-group">
        <button class="btn btn-primary btn-block">Sign Up</button>
      </div>

      <div class="alert alert-warning" *ngIf="f.submitted && isSignUpFailed">
        Signup failed!<br />{{ errorMessage }}
      </div>
```

```
    </form>

    <div class="alert alert-info">
      Do you have an account? <a href="/login">Log in</a>
    </div>

    <div class="alert alert-success" *ngIf="isSuccessful">
      Your registration is successful!
    </div>
  </div>
</div>
```

RegisterComponent.css

```
.card-container {
  max-width: 400px;
  margin: 0 auto;
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.1);
  background-color: tomato; /* Change the background color to tomato */
  display: flex;
  flex-direction: column;
  margin-top: 200px;
  justify-content: center;
  align-items: center;
}

.profile-img-card {
  width: 100px;
  height: 100px;
  border-radius: 50%;
  margin: 0 auto 20px;
  display: flex;
  justify-content: center;
  align-items: center;
  background-color: #007bff; /* Blue, adjust as needed */
}

.profile-img-card img {
  width: 80%;
  height: auto;
  border-radius: 50%;
}
```

```css
.form-group {
  margin-bottom: 20px;
}

label {
  font-size: 16px;
  color: #333;
}

.form-control {
  width: 100%;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
  outline: none;
}

.btn-primary {
  background-color: #4CAF50; /* Green, adjust as needed */
  color: white;
  border: none;
  border-radius: 5px;
  padding: 10px;
  cursor: pointer;
}

.alert {
  margin-top: 20px;
  border-radius: 5px;
}

.alert-danger {
  background-color: #F44336; /* Red, adjust as needed */
  color: white;
  border: 1px solid #B71C1C;
}

.alert-success {
  background-color: #2196F3; /* Blue, adjust as needed */
  color: white;
  border: 1px solid #1976D2;
}

.alert-info {
  background-color: #17a2b8; /* Cyan, adjust as needed */
  color: white;
  border: 1px solid #138496;
}
```

RegisterComponent.ts

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import { AuthService } from '../../_services/auth.service';

@Component({
  selector: 'app-register',
  standalone:true,
  imports: [FormsModule, CommonModule ],
  templateUrl: './register.component.html',
  styleUrls: ['./register.component.css']
})
export class RegisterComponent {
    form: any = {
    username: '',
    email: '',
    password: ''
  };
  isSuccessful = false;
  isSignUpFailed = false;
  errorMessage = '';

  constructor(private authservice: AuthService, private router: Router) { }

  ngOnInit(): void {

  }
  onSubmit(): void {
    const { username, email, password } = this.form;

    this.authservice.register(username, email, password).subscribe(
      response => {
        this.isSuccessful = true;
        this.isSignUpFailed = false;
        this.router.navigate(['/home']);
      },
      error => {
        this.errorMessage = error.error.message || 'An error occurred during 
registration.';
        this.isSignUpFailed = true;
      }
    );
  }
}
```

b) Login

LoginComponent.html

```
<div class="container">
  <div class="card">
    <img
      id="profile-img"
      src="//ssl.gstatic.com/accounts/ui/avatar_2x.png"
      class="profile-img-card"
    />
    <form
      *ngIf="!isLoggedIn"
      name="form"
      (ngSubmit)="onSubmit()"
      #f="ngForm"
      novalidate
    >
      <div class="form-group">
        <label for="email">Email</label>
        <input
          type="email"
          class="form-control"
          name="email"
          [(ngModel)]="form.email"
          required
          #email="ngModel"
        />
        <div
          class="alert alert-danger"
          role="alert"
          *ngIf="email.errors && f.submitted"
        >
```

```
          <div *ngIf="email.errors['required']">Email is required</div>
          <div *ngIf="email.errors['email']">Email must be a valid email
address</div>
        </div>
      </div>
      <div class="form-group">
        <label for="password">Password</label>
        <input
          type="password"
          class="form-control"
          name="password"
          [(ngModel)]="form.password"
          required
          minlength="6"
          #password="ngModel"
        />
        <div
          class="alert alert-danger"
          role="alert"
          *ngIf="password.errors && f.submitted"
        >
          <div *ngIf="password.errors['required']">Password is required</div>
          <div *ngIf="password.errors['minlength']">
            Password must be at least 6 characters
          </div>
        </div>
      </div>
      <div class="form-group">
        <button type="submit" class="btn btn-primary btn-block">Login</button>
      </div>
      <div class="form-group">
        <div
          class="alert alert-danger"
          role="alert"
          *ngIf="f.submitted && loginFailed"
        >
          Login failed: {{ errorMessage }}
        </div>
      </div>
    </form>
    <div class="form-group">
      <button class="btn btn-secondary btn-block" (click)="onCreateAccount()">
        Create new account
      </button>
    </div>
    <div class="alert alert-success" *ngIf="isLoggedIn">
      Logged in.
    </div>
  </div>
</div>
```

LoginComponent.css

```
body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  background-color: #f2f2f2;
  margin: 0;
  padding: 0;
}

.container {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
}

.card {
  background-color: tomato; /* Tomato color */
  width: 400px;
  padding: 30px;
  border-radius: 10px;
  box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.3);
}

.profile-img-card {
  width: 96px;
  height: 96px;
  margin: 0 auto 20px;
  border-radius: 50%;
  display: flex;
  justify-content: center;
  align-items: center;
  background-color: rgba(255, 255, 255, 0.8); /* White with transparency */
}

.profile-img-card img {
```

```css
  width: 60%;
  height: auto;
  border-radius: 50%;
}

.form-group {
  margin-bottom: 20px;
}

label {
  font-size: 16px;
  color: #333;
}

.form-control {
  width: 100%;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
  outline: none;
}

.btn-primary {
  background-color: #4CAF50; /* Green */
  color: white;
  border: none;
  border-radius: 5px;
  padding: 10px;
  cursor: pointer;
}

.btn-secondary {
  background-color: #FF9800; /* Orange */
  color: white;
  border: none;
  border-radius: 5px;
  padding: 10px;
  cursor: pointer;
}

.alert {
  margin-top: 20px;
  border-radius: 5px;
}
```

```
.alert-danger {
  background-color: #F44336; /* Red */
  color: white;
  border: 1px solid #B71C1C;
}

.alert-success {
  background-color: #2196F3; /* Blue */
  color: white;
  border: 1px solid #1976D2;
}
```

LoginComponent.ts

```
import { Component, OnInit } from '@angular/core';
import { AuthService } from '../../_services/auth.service';
import { TokenStorageService } from '../../_services/token-storage.service';
import { Router } from '@angular/router';
import { FormsModule } from '@angular/forms';
import { CommonModule } from '@angular/common';
import { HttpClient } from '@angular/common/http';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  standalone:true,
  imports: [FormsModule, CommonModule ],
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {
  form: any = {
    email: '',
    password: ''
  };
  isLoggedIn = false;
  loginFailed = false;
  errorMessage = '';

  constructor(private authService: AuthService, private tokenStorage:
TokenStorageService, private router: Router, private http: HttpClient) { }

  ngOnInit(): void {
  }
  onSubmit(): void {
    const { email, password } = this.form;

    if (!email || !password) {
      this.loginFailed = false;
      this.errorMessage = 'Please enter both email and password.';
      return; // Exit the function early if fields are missing
    }


    this.authService.login(email, password).subscribe(
      response => {this.tokenStorage.setUserId(response.user_id)
        this.tokenStorage.saveToken(response.token);
        this.tokenStorage.saveUser(response.user);
        this.isLoggedIn = true;
        this.loginFailed = false;
        alert(response.message);
        this.router.navigate(['/home']);
      },
      error => {
        if (error && error.error && error.error.error) {
          switch (error.error.error) {
            case 'User does not exist':
              this.errorMessage = 'User not found. Please check your email.';
              break;
            case 'Incorrect password':
              this.errorMessage = 'Incorrect password. Please try again.';
              break;
            default:
              this.errorMessage = 'An error occurred. Please try again later.';
              break;
          }
        } else {
          this.errorMessage = 'An error occurred. Please try again later.';
        }
        this.loginFailed = true;
      }
    );
  }
  onCreateAccount(): void {
    this.router.navigate(['/register']);
  }
}
```

c) Home

HomeComponent.html

```
<div class="container">

<!-- Fitness Program Section -->
<div class="fitness-program-container">
  <h2>Fitness Program</h2>
  <button class="create-program-button" (click)="createFitnessProgram()">Create
Fitness Program</button>
  <button class="edit-programs-button" (click)="loadFitnessPrograms()">View
Programs</button>
  <div *ngIf="creatingProgram">
    <h3>Create New Program</h3>
    <form (submit)="saveFitnessProgram()">
      <label for="programName">Program Name:</label>
      <input type="text" id="programName" name="programName"
[(ngModel)]="newProgramName" required>
      <label for="hoursPerWeek">Hours Per Week:</label>
      <input type="number" id="hoursPerWeek" name="hoursPerWeek"
[(ngModel)]="hoursPerWeek" required>
      <label for="daysPerWeek">Days Per Week:</label>
      <input type="number" id="daysPerWeek" name="daysPerWeek"
[(ngModel)]="daysPerWeek" required>
      <label for="bodyPart">Body Part:</label>
      <select id="bodyPart" name="bodyPart" [(ngModel)]="selectedBodyPart"
required>
        <option *ngFor="let part of bodyParts" [value]="part">{{ part }}</option>
      </select>
      <button type="submit">Save Program</button>
    </form>
  </div>
  <div *ngIf="fitnessPrograms">
    <div *ngFor="let program of fitnessPrograms">
      <p><strong>Body Part:</strong> {{ program.bodyPart }}</p>
      <p><strong>Program Name:</strong> {{ program.programName }}</p>
      <p><strong>Hours Per Week:</strong> {{ program.hoursPerWeek }}</p>
      <p><strong>Days Per Week:</strong> {{ program.daysPerWeek }}</p>
      <button (click)="deleteFitnessProgram(program)">Delete Program</button>
    </div>
  </div>
</div>
```

```
  <!-- Select Body Part Section -->
  <div class="select-body-part-container">
    <h2>Select Body Part</h2>
    <div class="body-part-list">
      <button *ngFor="let part of bodyParts" class="body-part-button"
(click)="changeBodyPart(part)">
        {{ part }}
      </button>
    </div>
  </div>

  <!-- Featured Workouts Section -->
  <div class="featured-workouts-container">
    <h2>Featured Workouts</h2>
    <div class="workout-scroll-container">
      <div *ngFor="let workout of featuredWorkouts" class="workout-card">
        <div class="workout-info">
          <h3>{{ workout.name }}</h3>
          <p><strong>Body Part:</strong> {{ workout.bodyPart }}</p>
          <p><strong>Muscles Worked:</strong> {{ workout.secondaryMuscles.join(',
') }}</p>
        </div>
        <div class="workout-gif">
          <img [src]="workout.gifUrl" alt="{{ workout.name }}">
        </div>
      </div>
    </div>
  </div>

</div>
```

HomeComponent.css

```
/* home.component.css */
.container {
  padding: 20px;
}
```

```css
.featured-workouts-container {
  margin-bottom: 30px;
}

.workout-scroll-container {
  display: flex;
  flex-wrap: wrap;
}

.workout-card {
  flex: 0 0 calc(33.333% - 20px);
  margin: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
  overflow: hidden;
}

.workout-info {
  padding: 10px;
  background-color: #f9f9f9;
}

.workout-info h3 {
  margin-top: 0;
}

.workout-gif {
  overflow: hidden;
}

.workout-gif img {
  width: 100%;
  height: auto;
}

.select-body-part-container {
  text-align: center;
}

.body-part-list {
  display: flex;
  justify-content: center;
  flex-wrap: wrap;
}
```

```css
.body-part-button {
  margin: 5px;
  padding: 10px 20px;
  border: none;
  border-radius: 20px;
  background-color: tomato;
  color: white;
  cursor: pointer;
}

.body-part-button:hover {
  background-color: darkred;
}
```

HomeComponent.ts

```typescript
import { Component, OnInit } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Router } from '@angular/router';
import { TokenStorageService } from '../../_services/token-storage.service';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import { FitnessProgramServiceService } from '../../_services/fitness-program-
service.service';
import { Observable } from 'rxjs';
import { FeaturedWorkoutsService } from '../../_services/featured-workouts-
service.service';

@Component({
  selector: 'app-home',
  standalone: true,
  imports:[CommonModule, FormsModule],
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit {

  featuredWorkouts: any[] = [];
  bodyParts: string[] = [
    'back', 'cardio', 'chest', 'lower arms', 'lower legs',
    'neck', 'shoulders', 'upper arms', 'upper legs', 'waist'
  ];
  fitnessPrograms$!: Observable<any>;
  selectedBodyPart: string = 'back'; // Default body part
  fitnessProgram: any; // Placeholder for fitness program data
  creatingProgram: boolean = false; // Flag to show/hide create program form
  newProgramName: string = ''; // Placeholder for new program name
  hoursPerWeek: number = 0; // Placeholder for hours per week
  daysPerWeek: number = 0; // Placeholder for days per week
  fitnessPrograms: any[] = []; // Placeholder for fitness program list
  selectedProgramId: string | null = null;


  constructor(private http: HttpClient, private fitnessProgramService:
FitnessProgramServiceService, private featuredWorkoutsService:
FeaturedWorkoutsService, private router: Router, private tokenStorage:
TokenStorageService) { }

  ngOnInit(): void {
    this.loadFeaturedWorkouts();
  }



  loadFeaturedWorkouts(): void {
    this.featuredWorkoutsService.getFeaturedWorkouts(this.selectedBodyPart)
      .subscribe(
        (data: any) => {
          this.featuredWorkouts = data;
        },
        (error: any) => {
          console.log('Error fetching featured workouts:', error);
        }
      );
  }

  changeBodyPart(bodyPart: string): void {
    this.selectedBodyPart = bodyPart;
    this.loadFeaturedWorkouts();
  }



  createFitnessProgram(): void {
    this.creatingProgram = true; // Show create program form

    const userId = this.tokenStorage.getUserId(); // Retrieve user ID from
TokenStorageService
```

```
    const programData = {
      userId: userId,
      programName: this.newProgramName,
      bodyPart: this.selectedBodyPart,
      hoursPerWeek: this.hoursPerWeek,
      daysPerWeek: this.daysPerWeek
    };

    this.http.post<any>('http://127.0.0.1:100/fitness_program', programData)
      .subscribe((data: any) => {
        this.fitnessProgram = data;
        this.creatingProgram = true; // Hide create program form
        this.newProgramName = ''; // Clear input field
        this.hoursPerWeek = 0; // Reset hours per week
        this.daysPerWeek = 0; // Reset days per week
      });
  }


  saveFitnessProgram(): void {
    this.fitnessProgramService.createFitnessProgram(this.newProgramName,
this.selectedBodyPart, this.hoursPerWeek, this.daysPerWeek)
      .subscribe((data: any) => {
        this.fitnessProgram = data;
        this.creatingProgram = false; // Hide create program form
        this.newProgramName = ''; // Clear input field
        this.hoursPerWeek = 0; // Reset hours per week
        this.daysPerWeek = 0; // Reset days per week
      });
  }

 viewPrograms(): void {
    this.fitnessProgramService.getFitnessPrograms()
      .subscribe(
        (programs: any[]) => {
          this.fitnessPrograms = programs;
        },
        (error: any) => {
          // Handle error if needed
        }
      );
  }
  loadFitnessPrograms(): void {
    this.router.navigate(['/fitness_program']);

  }






  deleteFitnessProgram(program: any): void {
    const confirmDelete = confirm('Are you sure you want to delete this
program?');
    if (confirmDelete) {
      this.fitnessProgramService.deleteFitnessProgram(program._id)
        .subscribe(
          () => {
            // Remove the program from the local list
            this.fitnessPrograms = this.fitnessPrograms.filter((p) => p._id !==
program._id);
            alert('Program deleted successfully!');
          },
          (error: any) => {
            console.error('Error deleting program:', error);
            alert('Error deleting program. Please try again.');
          }
        );
    }
  }
}
```

d) Fitness Programs

FitnessProgramsComponent.html

```
<div class="fitness-programs-container">
    <h2>Fitness Programs</h2>

    <!-- Display fitness programs -->
    <div *ngIf="items$ | async as items">
      <div *ngFor="let program of items" class="program-item">
        <div *ngIf="editingProgram !== program; else editForm">
          <div>{{ program.program_name }}</div>
          <div>{{ program.body_part }}</div>
          <div>Days Per Week: {{ program.days_per_week }}</div>
          <div>Hours Per Week: {{ program.hours_per_week }}</div>
          <div>Progress: {{ program.progress }}</div>
          <button (click)="editProgram(program)">Edit</button>
          <button (click)="deleteProgram(program)">Delete</button>
        </div>
        <ng-template #editForm>
          <div>
            <label for="programName">Program Name:</label>
            <input id="programName" [(ngModel)]="program.program_name"
placeholder="Program Name">
            <label for="bodyPart">Body Part:</label>
            <input id="bodyPart" [(ngModel)]="program.body_part"
placeholder="Body Part">
            <label for="daysPerWeek">Days Per Week:</label>
            <input id="daysPerWeek" [(ngModel)]="program.days_per_week"
placeholder="Days Per Week">
            <label for="hoursPerWeek">Hours Per Week:</label>
            <input id="hoursPerWeek" [(ngModel)]="program.hours_per_week"
placeholder="Hours Per Week">
            <button (click)="saveChanges(program)">Save</button>
            <button (click)="editProgram(null)">Cancel</button>
          </div>
        </ng-template>
      </div>
    </div>

</div>
```

FitnessProgramsComponent.css

```
/* Style for the fitness programs container */
.fitness-programs-container {
    max-width: 800px;
    margin: 0 auto;
    padding: 20px;
  }

  /* Style for the fitness program items */
  .program-item {
    margin-bottom: 20px;
    border: 1px solid #ccc;
    padding: 10px;
  }

  /* Style for the edit form within each program item */
  .program-item input {
    margin-bottom: 10px;
    padding: 5px;
  }

  /* Style for the buttons */
  .program-item button {
    margin-right: 10px;
    padding: 5px 10px;
    cursor: pointer;
    background-color: #007bff;
    color: #fff;
    border: none;
    border-radius: 4px;
  }

  /* Style for the buttons on hover */
  .program-item button:hover {
    background-color: #0056b3;
  }
```

FitnessProgramsComponent.ts

```
import { CommonModule } from '@angular/common';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { switchMap, catchError, Observable, throwError } from 'rxjs';
import { FitnessProgramServiceService } from '../../_services/fitness-program-
service.service';


@Component({
  selector: 'app-fitness-programs',
  standalone: true,
  imports: [FormsModule, CommonModule],
  templateUrl: './fitness-programs.component.html',
  styleUrl: './fitness-programs.component.css'
})
export class FitnessProgramsComponent implements OnInit {
  fitnessPrograms: any[] = [];
  editingProgram: any = null;
  items$!: Observable<any[]>;
  totalItemsCount$!: Observable<number>;

  constructor(private http: HttpClient,   private fitnessProgramService:
FitnessProgramServiceService
  ) { }

  ngOnInit(): void {
    this.loadFitnessPrograms();
  }

  loadFitnessPrograms(): void {
    const url = 'http://localhost:100/fitness_programs'; // Adjusted port number

    this.items$ = this.http.get<any[]>(url).pipe(
      catchError((error) => {
        console.error('Error loading fitness programs:', error);
        return throwError(error);
      })
    );
  }



  editProgram(program: any): void {
    this.editingProgram = program;
  }

  saveChanges(program: any): void {
    const programId = program._id;
    const url = `http://localhost:100/fitness_program/${programId}`;

    this.http.put<any>(url, program).pipe(
      catchError((error) => {
        console.error('Error updating program:', error);
        return throwError(error);
      })
    ).subscribe(() => {
      console.log('Program updated successfully');
      this.loadFitnessPrograms(); // Reload programs to reflect changes
      this.editingProgram = null; // Reset editing mode
    });
  }
  deleteProgram(program: any): void {
    const programId = program._id;
    const url = `http://localhost:100c/fitness_program/${programId}`;

    this.http.delete<any>(url).pipe(
      catchError((error) => {
        console.error('Error deleting program:', error);
        return throwError(error);
      })
    ).subscribe(() => {
      console.log('Program deleted successfully');
      this.loadFitnessPrograms(); // Reload programs after deletion
    });
    }

  }
```

Services

Auth Service

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Observable } from 'rxjs';

const AUTH_API = 'http://localhost:100/';

const httpOptions = {
  headers: new HttpHeaders({ 'Content-Type': 'application/json' })
};

@Injectable({
  providedIn: 'root'
})
export class AuthService {
  constructor(private http: HttpClient, ) { }

  login(email: string, password: string): Observable<any> {

    return this.http.post(`${AUTH_API}/login`, { email, password });
  }

  register(username: string, email: string, password: string) {
    const data = { username, email, password };
    return this.http.post<any>(`${AUTH_API}/register`, data);
  }
}
```

Fitness Program Service

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { catchError, Observable, throwError } from 'rxjs';

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

  constructor(private http: HttpClient) { }

  createFitnessProgram(programName: string, bodyPart: string, hoursPerWeek:
number, daysPerWeek: number): Observable<any> {
    return this.http.post<any>('http://127.0.0.1:100/fitness_program', {
programName, bodyPart, hoursPerWeek, daysPerWeek });
  }


  getFitnessPrograms(): Observable<any[]> {
    return this.http.get<any[]>('http://127.0.0.1:100/fitness_program').pipe(
      catchError((error) => {
        console.error('Error loading fitness programs:', error);
        return throwError(error);
      })
    );
  }



  deleteFitnessProgram(programId: string): Observable<any> {
    return
this.http.delete<any>(`http://127.0.0.1:100/fitness_program/${programId}`);
  }
}
```

Token Storage Service

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

const TOKEN_KEY = 'auth-token';
const USER_KEY = 'auth-user';

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

   // Function to set user ID in sessionStorage
   setUserId(userId: string): void {
    localStorage.setItem('userId', userId);
  }

  // Function to get user ID from sessionStorage
  getUserId(): string | null {
    return localStorage.getItem('userId');
  }

  // Function to clear user ID from sessionStorage (e.g., on logout)
  clearUserId(): void {
    localStorage.removeItem('userId');
  }

  signOut(): void {
    this.clearUserId();
  }

  public saveToken(token: string): void {
    window.sessionStorage.removeItem(TOKEN_KEY);
    window.sessionStorage.setItem(TOKEN_KEY, token);
  }

  public getToken(): string | null {
    return window.sessionStorage.getItem(TOKEN_KEY);
  }

  public saveUser(user: any): void {
    window.sessionStorage.removeItem(USER_KEY);
    window.sessionStorage.setItem(USER_KEY, JSON.stringify(user));
  }

  public getUser(): any {
    const user = window.sessionStorage.getItem(USER_KEY);
    if (user) {
      return JSON.parse(user);
    }

    return {};
  }
}
```

Featured Workout Service

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';

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



  private apiUrl = 'http://localhost:100/exercise/';

  constructor(private http: HttpClient, ) { }

  getFeaturedWorkouts(bodyPart: string): Observable<any> {
    return this.http.get<any>(`${this.apiUrl}${bodyPart}?limit=10`);
  }
}
```

Proxy.config

```
{
    "/api/*": {
        "target": "https://127.0.0.1:100",
        "secure": "false",
        "loglevel": "debug"
    }
}
```