



## **Faculty of Automation and Computer Science**



# Industrial Informatics Semester Project

2022 – 2023

Link: [github.com/NTimea302/UpFit](https://github.com/NTimea302/UpFit)

### **Team members:**

Nagy Timea | Team Leader

Nemes Raluca | Software Tester

Oprea Florin Octavian | Software developer

Szekely Rolland Iulian | Database developer

### **Coordinating teacher:**

Eng. Teodora Sanislav

## INTRODUCTION (GOAL, OBJECTIVES, FUNCTIONS)

UpFit is a fitness application designed to help individuals track their meals, monitor their calorie intake, and maintain a healthy lifestyle. The application caters to four different types of users: admin, coach, basic user, and premium user. Each user type has specific functionalities and privileges within the application.

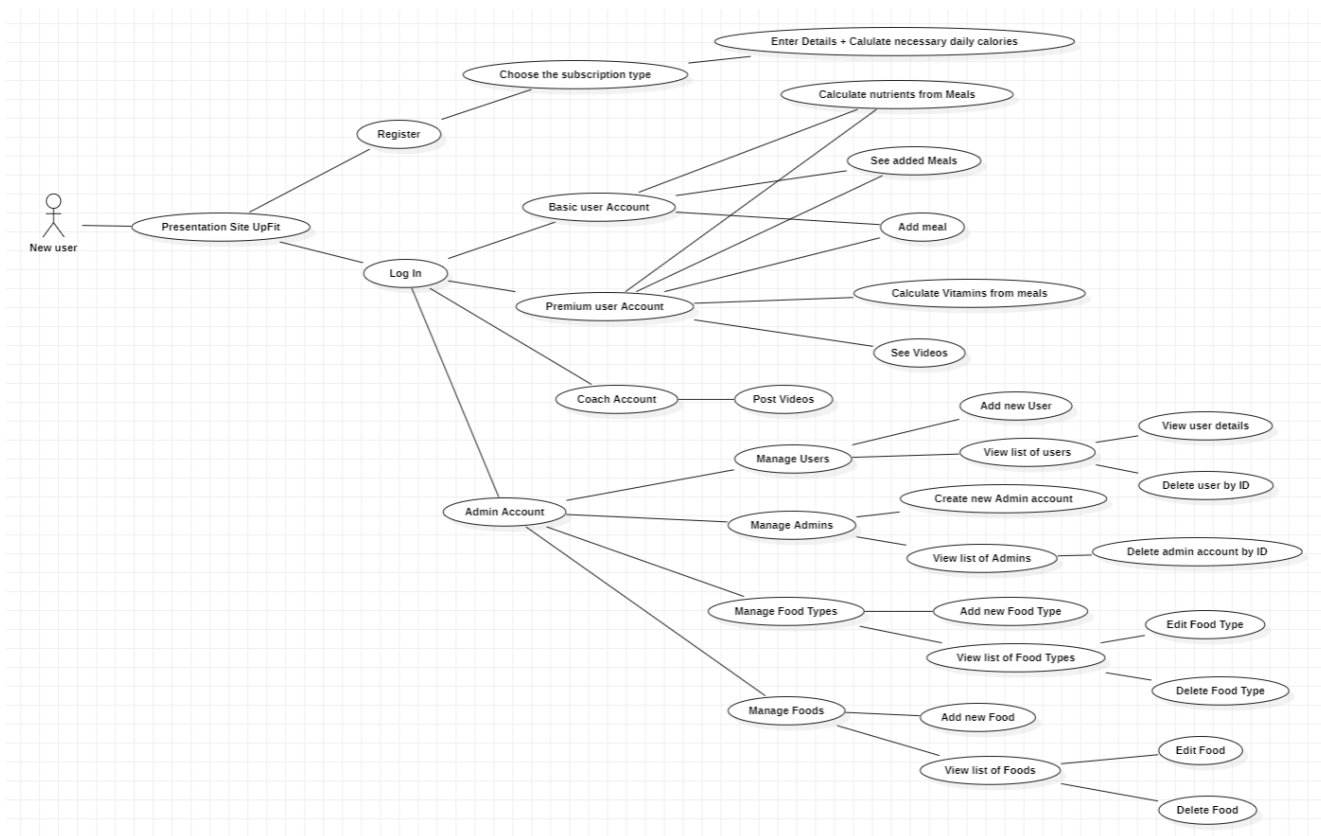
The primary purpose of UpFit is to provide a platform where users can conveniently manage their nutrition and monitor their progress towards their fitness goals. The application offers features such as user registration, login functionality, meal tracking, meal history, and personalized calorie calculations based on user data such as weight, height, and activity level. UpFit calculates the recommended daily calorie intake.

Admins have full access and control over the application, including user management, food and food types management. Coaches have the ability to post videos that can be viewed by premium users. Basic users have access to essential features such as meal tracking, meal history, and personalized calorie calculations. Premium users enjoy additional benefits and advanced features within the application such as viewing the vitamins they consumed.

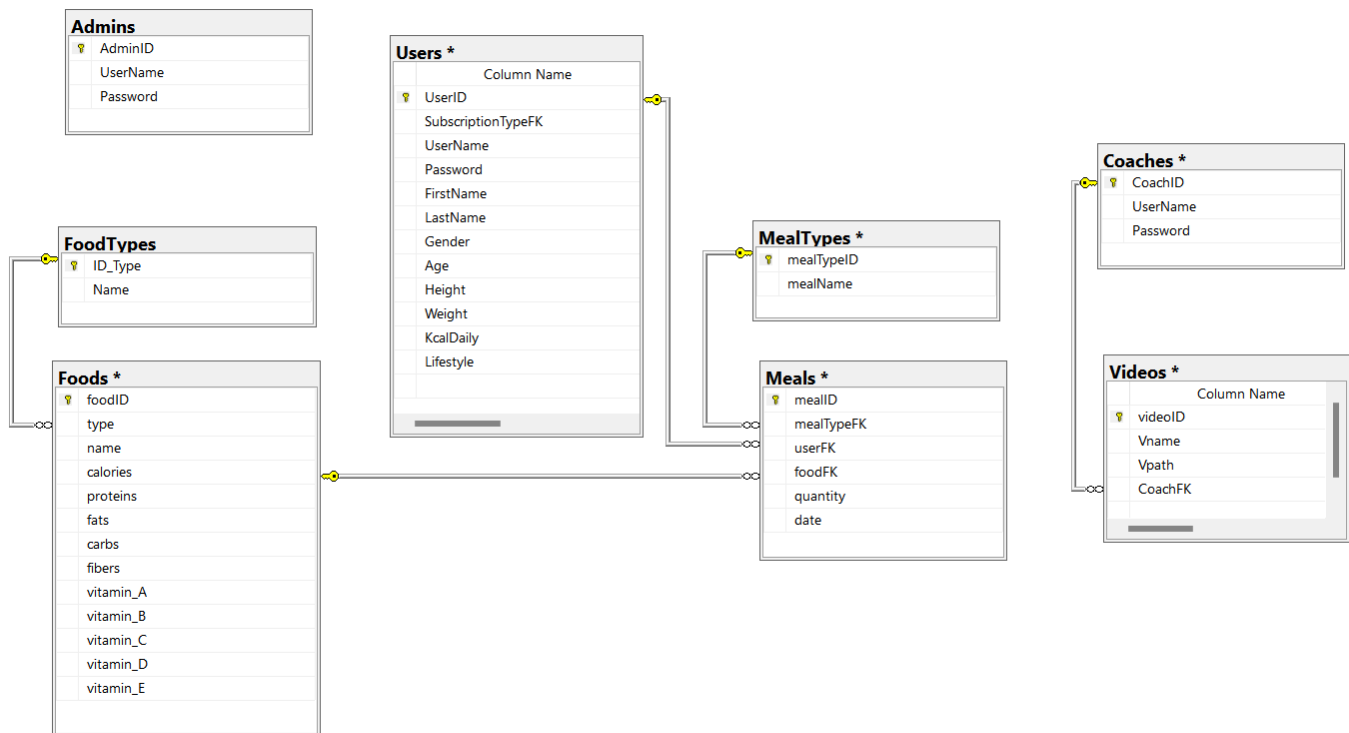
UpFit enables users to add meals to their daily intake, allowing them to monitor their calorie consumption accurately. Users can select from a pre-defined list of foods from a datatable, making it easier to track the nutritional content of their meals.

## APPLICATION DESIGN

### UML DIAGRAMS – USE CASE



## DATABASE DESIGN



The application has a well-structured database design to effectively manage data and support its various functionalities. The database consists of eight tables, each serving a specific purpose:

- Admins Table:**
  - Stores the credentials of administrators.
  - Contains columns such as ID (identifier), Username, and Password.
- Coaches Table:**
  - Stores the credentials of coaches.
  - Contains columns such as ID, Username, and Password.
- Videos Table:**
  - Stores information about uploaded videos.
  - Includes columns such as ID, Name (video title), Path (file path), and CoachFK (foreign key)
- FoodTypes Table:**
  - Stores different types of food categories.
  - Contains columns such as ID and Name (e.g., fruits, vegetables, meats, nuts, seeds).
- Foods Table:**
  - Stores detailed information about individual food items.
  - Includes columns such as ID, TypeFK (foreign key referencing the corresponding food type), Name, Calories, Proteins, Fats, Carbs, Fibers, and various vitamin fields (A, B, C, D, E).
- MealTypes Table:**
  - Stores different meal types, such as breakfast, lunch, and dinner.
  - Contains columns such as ID and Name.
- Meals Table:**
  - Represents individual meal instances recorded by users.
  - Includes columns such as ID, MealtypeFK (foreign key referencing the meal type), UserFK (foreign key referencing the associated user), FoodFK (foreign key referencing the consumed food), Quantity, and Date.

## GRAPHICAL USER INTERFACE DESIGN AND FUNCTIONALITIES

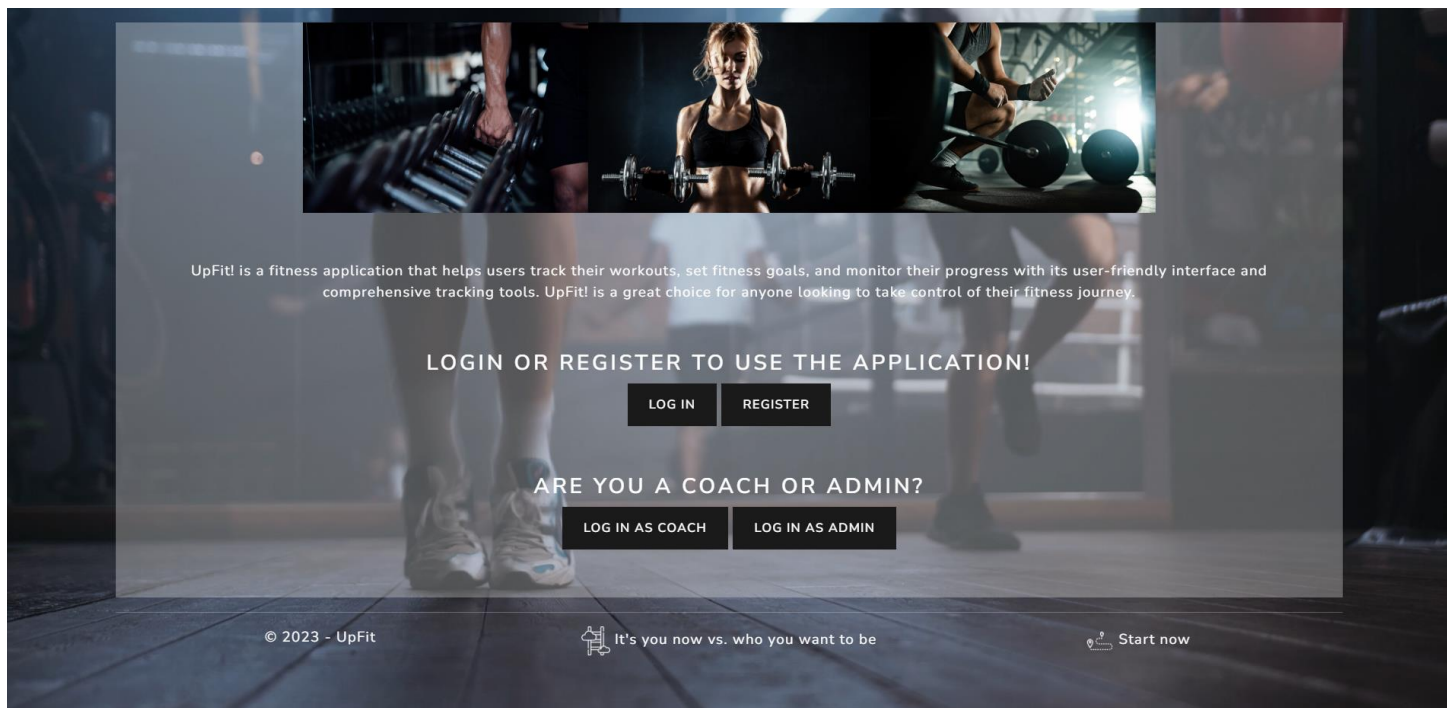
Upon launching the UpFit application, users are presented with an informative home screen, providing a general overview of the application's features and functionalities. The home screen serves as the starting point for users to access the login and registration functionalities.

Users have the option to log in to their existing accounts by entering their credentials, such as their username and password. Once logged in, users can view a table of their consumed nutrients for the day and have the option to add new meals by accessing the 'Add Meal' button and selecting a food type, a food from the database and entering the consumed amount.

For new users who have not yet registered, the application offers a registration process. Users can create a new account by choosing their preferred subscription type option and by providing necessary personal details such as their name, username, password and other details about their lifestyle and other relevant data that helps in determining personalized fitness goals and recommendations, such as height, weight and age. Upon successful registration, users can proceed to log in.

In addition to regular users, the application offers access to admins and coaches too. These type of users have their own login functionality. Admins have comprehensive access and control over the system, including user and food management. Coaches can upload videos for users.

### STARTING PAGE OF UPFIT!




## USERS GUI

The graphical user interface (GUI) for both basic users and premium users in the UpFit application shares a similar design and layout. However, premium users are granted additional options and features compared to basic users such as accessing videos related to exercises and tutorials, as well as being able to view detailed information about the vitamin content in their meals.

The differentiation between basic and premium users lies primarily in the availability of specific features and access to premium content, while the core GUI remains unchanged.

### LOG IN AND REGISTER OF USER



LOG IN WITH USER ACCOUNT


UserName


Password

LOGIN

[Back](#) [Mail](#)

© 2023 - UpFit

 It's you now vs. who you want to be

 Start now



REGISTER

Subscription Type

UserName

Password

FirstName

LastName

Gender

Age

Height

Weight

How phisically active are you?

© 2023 - UpFit

 It's you now vs. who you want to be

 Start now

## Model View Controller Application

### PREMIUM AND BASIC USER ACCOUNT

After logging in, the user is directed to either their premium or basic user account. In this section, the user is presented with essential information regarding their daily necessary intake for maintenance.

The user interface displays the following details:

- A message indicating that their daily necessary intake has been calculated based on the provided information.
- A goal recommendation for the user, indicating the target calorie intake per day, displayed as "@kcaldaily kcal".
- Information on the remaining calories for the user, presented as "You have left: @remainedKcal kcal".

Furthermore, the user has the option to add a new meal. By selecting this option, the user can input the details of the food and quantity they consumed. The added meal is then included in the user's daily nutrient list, which is presented in a tabular format.

The daily nutrient list contains the following columns:

- Nutrient: Displays the name of the nutrient, including "Calories consumed", "Protein", "Carbohydrates", "Fats", and "Fiber".
- Total Amount: Indicates the respective total amount of each nutrient consumed, retrieved from the "@macrodaily" variable.

The user can refer to this nutrient table to track their daily intake and monitor their nutritional values accordingly.

The image displays two side-by-side screenshots of a fitness application interface, showing user profiles and daily nutrient intake tables.

**Left Screenshot (User: RALUCA):**

- Header: HELLO, RALUCA
- Message: Based on your info we calculated your daily necessary intake for maintenance.
- Goal: Your goal should be: 1223 kcal
- Remaining: You have left: 982 kcal
- Button: ADD MEAL
- Table: MY DAILY NUTRIENTS

NUTRIENT	TOTAL AMOUNT
Calories consumed	241
Protein	23.545
Carbohydrates	4.05
Fats	15.175
Fiber	1.035

**Right Screenshot (User: TIMEA):**

- Header: HELLO, TIMEA
- Message: Based on your info we calculated your daily necessary intake for maintenance.
- Goal: Your goal should be: 907 kcal
- Remaining: You have left: 633 kcal
- Button: ADD MEAL
- Table: MY DAILY NUTRIENTS

NUTRIENT	TOTAL AMOUNT
Calories consumed	274
Protein	15.55
Carbohydrates	22.9
Fats	13.25
Fiber	0.85
Vitamin A	1.89
Vitamin B	0.1725
Vitamin C	0.1209
Vitamin D	1.6085

## ADMIN GUI

Upon successful login, the admin has access to four distinct management sections in the UpFit application to perform various administrative tasks and maintain the system efficiently. The four management sections available to the admin are as follows:

1. User Management:

- In the "Manage Users" section, the admin can add new users to the system, view the existing list of users, view the details of each user, and delete user accounts as necessary.

2. Admin Management:

- The "Manage Admins" section allows the admin to create new admin accounts or delete existing admin accounts as required.


3. Food Type Management:

- In the "Manage Food Types" section, the admin has the ability to create new food types, edit existing food types, or remove food types from the system.

4. Food Management:

- Within the "Manage Foods" section, the admin can create new food entries, edit the details of existing food items, or remove food items from the application.
- This feature allows the admin to curate a database of foods that users can choose from when tracking their meals, ensuring accurate nutritional information and a diverse selection.

## LOG IN AND STARTING PAGE OF ADMIN



### LOG IN AS ADMIN

UserName

Password

LOG IN


[Back to Index](#)

---

© 2023 - UpFit

It's you now vs. who you want to be

Start now



### LOGGED IN AS ADMIN

Welcome to your admin account! Select the operation you want to perform!

- HOME
- MANAGE USERS
- MANAGE ADMINS
- MANAGE FOODS
- MANAGE FOOD TYPES
- LOG OUT

---

© 2023 - UpFit

It's you now vs. who you want to be

Start now

## MANAGE USERS

BACK

CREATE NEW

SUBSCRIPTION TYPE	USERNAME	FIRSTNAME	LASTNAME	
0	Clemi33	Clementina	Teodor	<a href="#">Delete</a>   <a href="#">Details</a>
0	Raluca	Raluca	Nemes	<a href="#">Delete</a>   <a href="#">Details</a>
1	Timea	Timea	Nagy	<a href="#">Delete</a>   <a href="#">Details</a>
1	Adela90	Adela	Ionescu	<a href="#">Delete</a>   <a href="#">Details</a>
1	Radu78	Radu	Popescu	<a href="#">Delete</a>   <a href="#">Details</a>
0	IoanaDumitrescu	Ioana	Dumitrescu	<a href="#">Delete</a>   <a href="#">Details</a>
1	Lucian55	Lucian	Stanescu	<a href="#">Delete</a>   <a href="#">Details</a>
0	AnaPetrescu2	Ana	Petrescu	<a href="#">Delete</a>   <a href="#">Details</a>
0	Paul23	Paul	Mihai	<a href="#">Delete</a>   <a href="#">Details</a>
1	Cristina87	Cristina	Rad	<a href="#">Delete</a>   <a href="#">Details</a>
1	Andreiii3	Andrei	Popa	<a href="#">Delete</a>   <a href="#">Details</a>



## USER DETAILS AND DELETION INTERFACE FOR ADMIN

DETAILS  
USER

Subscription Type  
1  
UserName  
Cristina87  
Password  
upfitaplicatiecristina  
FirstName  
Cristina  
LastName  
Rad  
Gender  
F  
Age  
34  
Height  
163  
Weight  
55  
KcalDaily  
992

[Back to List](#)

© 2023 - UpFit

## DELETE USER

ARE YOU SURE YOU WANT TO DELETE  
THIS USER?

Subscription Type  
1  
UserName  
Timea  
Password  
timeaa  
FirstName  
Timea  
LastName  
Nagy  
Gender  
F  
Age  
22  
Height  
172  
Weight  
51  
KcalDaily  
907

DELETE | [Back to List](#)

© 2023 - UpFit

It's you now  
vs. who you want  
to be

Start now

The interface for managing admins (viewing admins list and deleting admins) and for managing users (viewing users list, user details, and deleting users) shares a unified design, making it easy for admins to perform tasks such as viewing lists and deleting accounts. The consistent interface ensures a seamless user experience and simplifies administrative management.

## FOOD AND FOOD TYPES MANAGEMENT USER INTERFACES

## MANAGE FOOD TYPES

BACK

CREATE NEW

NAME

Fruits

[Edit](#) | [Delete](#)

Vegetables

[Edit](#) | [Delete](#)

Meat

[Edit](#) | [Delete](#)

Seafood

[Edit](#) | [Delete](#)

Dairy Product

[Edit](#) | [Delete](#)

Grains

[Edit](#) | [Delete](#)

Nuts and Seeds

[Edit](#) | [Delete](#)

Bakery

[Edit](#) | [Delete](#)

## MANAGE FOODS

BACK

CREATE NEW

TYPE

NAME

Fruits

Apple

[Edit](#) |  
[Delete](#)

Fruits

Orange

[Edit](#) |  
[Delete](#)

Fruits

Banana

[Edit](#) |  
[Delete](#)

Fruits

Strawberry

[Edit](#) |  
[Delete](#)

Vegetables

Carrot

[Edit](#) |  
[Delete](#)

Vegetables

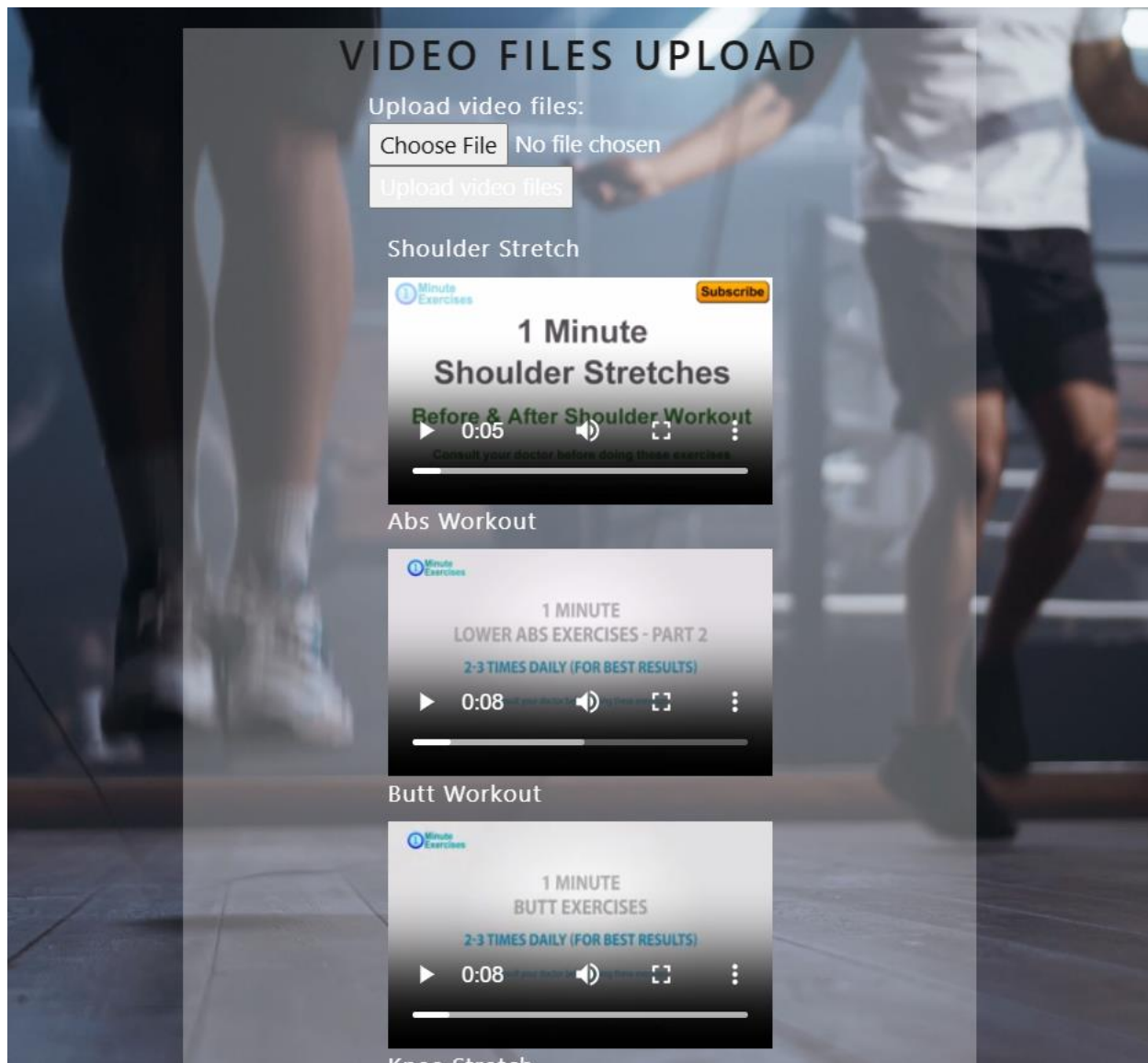
Broccoli

[Edit](#) |  
[Delete](#)

The graphical user interface (GUI) designed for coaches in the UpFit application provides convenient access to uploaded videos and allows coaches to upload new videos directly from their devices.

Within the coach interface, coaches can view a list or gallery displaying the videos that have been uploaded so far. This provides coaches with a visual overview of the available video content for their users.

By offering these features within the coach GUI, UpFit empowers coaches to manage and expand the available video content, ensuring a rich and diverse library of resources for users.



## APPLICATION IMPLEMENTATION

The implementation of the UpFit application involved transforming the design and the ideas into an actual code and integrating the mentioned functionalities. The application was developed in Visual Studio 2022 using the ASP.net Core Web App (Model-View-Controller) template. We started by defining the necessary classes/models. We used Entity Framework and code-first approach. After we defined the models, we generated a database schema based on these classes. We implemented the controllers to handle user requests and actions, and designed the views to provide a user-friendly interface.

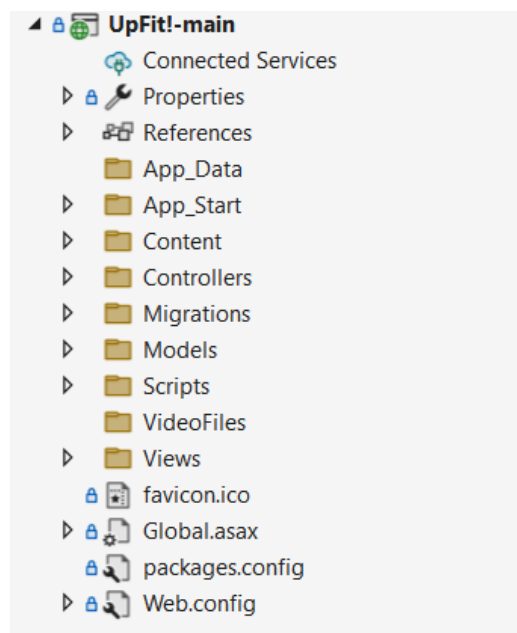
## STRUCTURE OF APPLICATION

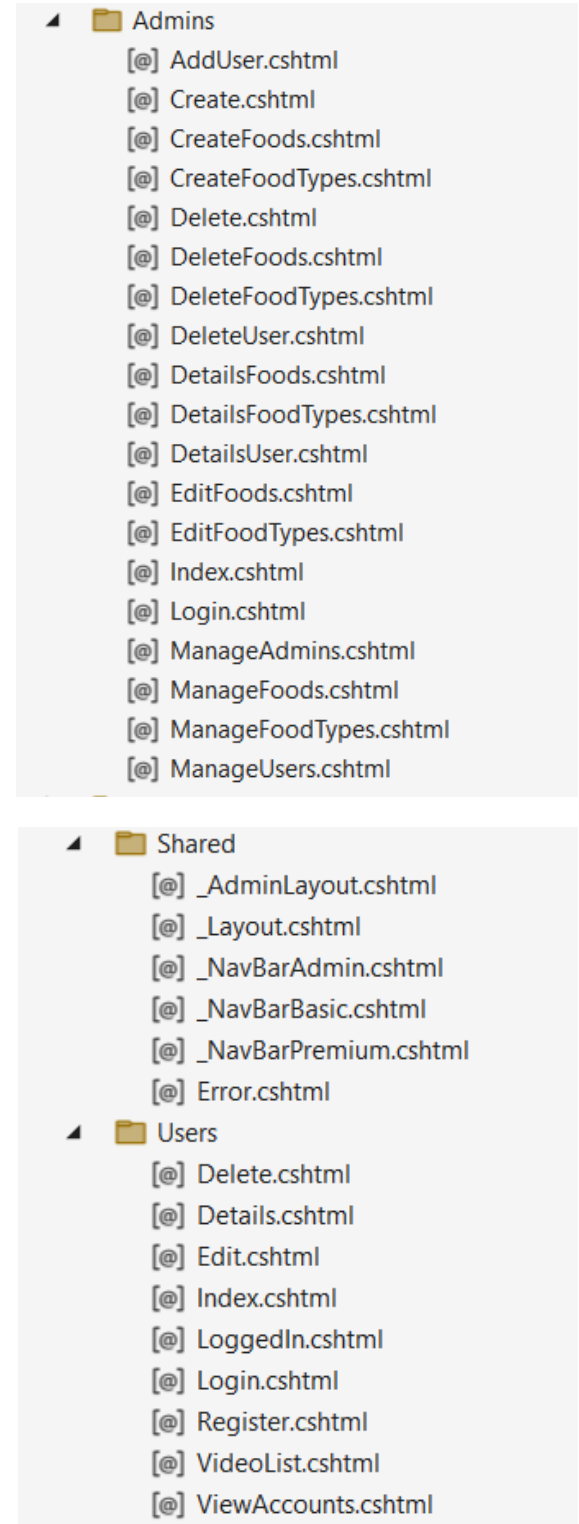
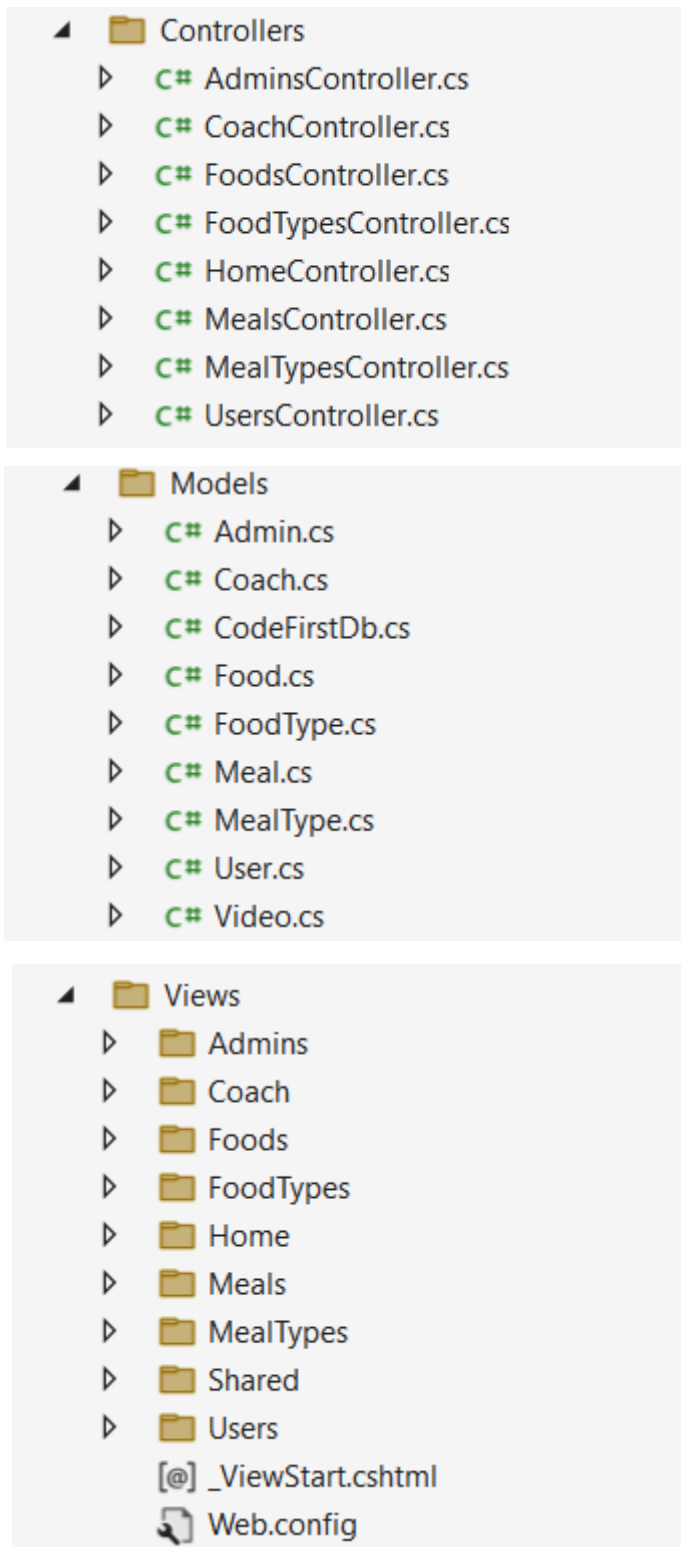
The UpFit application follows the Model-View-Controller (MVC) architectural pattern, which provides a structured approach to organizing and separating different components. The application consists of several folders, each serving a specific purpose:

1. **Controllers:** This folder contains the controllers that handle user requests, process data, and control the flow of the application. Controllers interact with the models and views to perform various actions.
2. **Migrations:** The Migrations folder stores database migration files generated by Entity Framework. These files track changes to the database schema over time, allowing for easy database updates and version control.
3. **Models:** The Models folder holds the model classes that represent the entities and data structures within the application. These classes define properties, relationships, and business logic related to the data.
4. **Views:** The Views folder contains the user interface templates that display information to the users. Views are responsible for rendering data and presenting it in a visually appealing and interactive manner.
5. **VideoFiles:** This folder is specifically dedicated to storing video files used within the application. It will contain videos related to exercises, tutorials, or any other multimedia content.

In the provided pictures, the model classes are illustrated. These classes represent the various entities and their relationships, providing a visual representation of the application's data structure.

By organizing the application into these distinct folders and adopting the MVC pattern, UpFit ensures modularity, separation of concerns, and a clear structure that facilitates maintenance and extensibility.





## IMPLEMENTATION FOR DAILY CALORIE INTAKE CALCULATOR

The code snippet provided represents the implementation for the daily calorie intake calculator within the Register action of the Users Controller.

```

106 [HttpPost]
107 [ValidateAntiForgeryToken]
108 public ActionResult Register([Bind(Include = "UserID,SubscriptionTypeFK,UserName,Password,FirstName,LastName,Gender,Age,Height,Weight
109 {
110     if (user.Gender == "M")
111     {
112         user.KcalDaily = (int)Math.Round(user.Lifestyle * (66 + (13.7 * user.Weight) + (5 * user.Height) - (6.8 * user.Age)));
113     }
114     else
115     {
116         user.KcalDaily = (int)Math.Round(user.Lifestyle * (65 + (9.5 * user.Weight) + (1.8 * user.Height) - (4.7 * user.Age)));
117     }
118     if (ModelState.IsValid)
119     {
120         db.users.Add(user);
121         db.SaveChanges();
122         return RedirectToAction("Login");
123     }
124
125     return View(user);
126 }

```

To enhance personalization, the application allows users to select their lifestyle level, which can be sedentary, medium, active, or very active at registration. These lifestyle levels are mapped to corresponding numeric values in the code, with sedentary being assigned a value of 1.2, medium as 1.375, active as 1.55, and very active as 1.725.

Based on the user's gender and selected lifestyle level, the code performs the necessary calculations using the Harris-Benedict equation. For male users (indicated by a Gender value of "M"), the equation for men is utilized. Conversely, for female users, the equation for women is applied. The resulting calorie intake is then assigned to the user's KcalDaily property.

By incorporating this code, the UpFit application accurately calculates each user's recommended daily calorie intake based on their gender and selected lifestyle level. This personalized approach ensures that users receive tailored guidance to support their health and fitness goals.

## CALCULATING NUTRIENTS FROM MEALS

```

1  @model UpFit_main.Models.User
2  @{
3      ViewBag.Title = "LoggedIn";
4      var alo = Session["SubscriptionType"].ToString();
5      var userID = Session["UserID"].ToString();
6      var kcaldaily = Session["KcalDaily"];
7  }
8
9
10 @using (Html.BeginForm("MacroCalculations"))
11 {
12     List<double> macrodaily = ViewBag.Macrodaily;
13     var remainedKcal = (int)kcaldaily - (int)Math.Round(macrodaily[0]);
14     if (alo == "0")
15     {
16         @Html.Partial("_NavBarBasic")
17
18         @Html.AntiForgeryToken()
19         <div class="d-flex justify-content-center align-items-center flex-column" style="background-color: rgba(255,255,255,0.3); ">
20             <h2> Hello, @Model.FirstName </h2>
21             <div>
22                 <p>
23                     Based on your info we calculated your daily necessary intake for maintenance.
24                 </p>
25                 <p>
26                     Your goal should be: @kcaldaily kcal
27                 </p>
28                 <p>
29                     You have left: @remainedKcal kcal
30                 </p>
31             </div>
32         </div>

```

## Model View Controller Application

The below provided code snippet represents the implementation of the "MacroCalculations" action within the Users Controller. This action is accessed via an HTTP GET request and takes a user ID as a parameter.

Inside the action, the code retrieves the current date and initializes variables for storing the calculated nutrient values. It then queries the database to iterate through each meal entry associated with the specified user ID. For each meal, the code retrieves the corresponding food item from the "foods" table and calculates the nutrient values based on the meal's quantity.

The calculated nutrient values, such as calories, carbohydrates, proteins, fats, fibers, and various vitamins, are accumulated and stored in a list called "totalmacro." This list is then passed to the "LoggedIn" view using the ViewBag.

The "LoggedIn" view can access these nutrient values and display them accordingly using the code snippet "@using (Html.BeginForm("MacroCalculations"))".

In summary, the "MacroCalculations" action calculates the nutrient values from the user's meals and makes them available in the "LoggedIn" view, allowing users to view their daily nutrient intake.

```
214 [HttpGet]
215 public List<double> MacroCalculations(int userID)
216 {
217     DateTime day = DateTime.Today;
218     double calories = 0;
219     double carbo = 0.00, proteins = 0.00, fats = 0.00, fibers = 0.00, vitamin_A = 0.00, vitamin_B = 0.00, vitamin_C = 0, vitamin_D = 0, vitamin_E = 0;
220     List<double> totalmacro = new List<double>();
221     using (CodeFirstDb db = new CodeFirstDb())
222     {
223         foreach (Meal meal in db.meals)
224         {
225             if (meal.userFK == userID)
226             {
227                 if (meal.date == day)
228                 {
229                     Food aliment = db.foods.SingleOrDefault(x => x.foodID == meal.foodFK);
230                     calories = calories + (aliment.calories)*meal.quantity/100;
231                     carbo = carbo + (double)aliment.carbs*meal.quantity/100;
232                     proteins = proteins + (double)aliment.proteins* meal.quantity/100;
233                     fats=fats+ (double)aliment.fats * meal.quantity / 100;
234                     fibers=fibers +(double)aliment.fibers * meal.quantity / 100;
235                     vitamin_A = vitamin_A + (double)aliment.vitamin_A * meal.quantity / 100;
236                     vitamin_B = vitamin_B + (double)aliment.vitamin_B * meal.quantity / 100;
237                     vitamin_C=vitamin_C + (double)aliment.vitamin_C * meal.quantity / 100;
238                     vitamin_D = vitamin_D + (double)aliment.vitamin_D * meal.quantity / 100;
239                     vitamin_E = vitamin_E + (double)aliment.vitamin_E * meal.quantity / 100;
240                 }
241             }
242         }
243         totalmacro.AddRange(new double[] { calories, carbo, proteins, fats, fibers, vitamin_A, vitamin_B, vitamin_C, vitamin_D, vitamin_E });
244         ViewBag.MacroDaily = totalmacro;
245     }
246     return totalmacro;
247 }
```

## USER MODEL

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace UpFit__main.Models
{
    public class User
    {
        [Key]
        public int UserID { get; set; }

        [Required(ErrorMessage = "Subscription Type is required.")]
        [DisplayName("Subscription Type")]
        public int SubscriptionTypeFK { get; set; }

        [Required(ErrorMessage = "Username is required.")]
        [RegularExpression(@"^[a-zA-Z0-9_]+$", ErrorMessage = "Username can only contain letters, numbers, and underscores.")]
        public string UserName { get; set; }

        [DataType(DataType.Password)]
        [Required(ErrorMessage = "Password is required.")]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.", MinimumLength = 6)]
        public string Password { get; set; }

        [Required(ErrorMessage = "First Name is required.")]
        [RegularExpression(@"^[a-zA-Z]+$", ErrorMessage = "First Name can only contain letters.")]
        public string FirstName { get; set; }

        [Required(ErrorMessage = "Last Name is required.")]
        [RegularExpression(@"^[a-zA-Z]+$", ErrorMessage = "Last Name can only contain letters.")]
        public string LastName { get; set; }

        [RegularExpression(@"^[FM]$", ErrorMessage = "Gender must be 'F' or 'M'.")]
        [Required(ErrorMessage = "Gender is required.")]
        public string Gender { get; set; }

        [Range(1, 150, ErrorMessage = "Age must be between 1 and 150.")]
        public int Age { get; set; }

        [Range(50, 300, ErrorMessage = "Height must be between 50 and 300.")]
        public int Height { get; set; }

        [Range(10, 300, ErrorMessage = "Weight must be between 10 and 300.")]
        public int Weight { get; set; }

        [Range(0, int.MaxValue, ErrorMessage = "Kcal Daily must be a positive value.")]
        public int KcalDaily { get; set; }

        public double Lifestyle { get; set; }
    }
}

```



## FOOD MODEL

```
namespace UpFit__main.Models
{
    public class Food
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int foodID { get; set; }

        [Display(Name = "What type of food is?")]
        [Required(ErrorMessage = "Type Required")]
        public int type { get; set; }

        [Display(Name = "Name")]
        [Required(ErrorMessage = "Name Required")]
        public string name { get; set; }

        [Display(Name = "Calories")]
        [Required(ErrorMessage = "Calories Required")]
        public int calories { get; set; }

        [Display(Name = "Proteins")]
        [Required(ErrorMessage = "Proteins Required")]
        public double? proteins { get; set; }

        [Display(Name = "Fats")]
        [Required(ErrorMessage = "Fats Required")]
        public double? fats { get; set; }

        [Display(Name = "Carbohydrates")]
        [Required(ErrorMessage = "Carbs Required")]
        public double? carbs { get; set; }

        [Display(Name = "Fibers")]
        [Required(ErrorMessage = "Fibers Required")]
        public double? fibers { get; set; }

        [Display(Name = "Vitamin A")]
        [Required(ErrorMessage = "Vitamin A Required")]
        public double? vitamin_A { get; set; }

        [Display(Name = "Vitamin B")]
        [Required(ErrorMessage = "Vitamin B Required")]
        public double? vitamin_B { get; set; }

        [Display(Name = "Vitamin C")]
        [Required(ErrorMessage = "Vitamin C Required")]
        public double? vitamin_C { get; set; }

        [Display(Name = "Vitamin D")]
        [Required(ErrorMessage = "Vitamin D Required")]
        public double? vitamin_D { get; set; }

        [Display(Name = "Vitamin E")]
        [Required(ErrorMessage = "Vitamin E Required")]
        public double? vitamin_E { get; set; }
    }
}
```



## INDEX PAGE FRONT-END

```

<main aria-labelledby="title">
  <div class="d-flex justify-content-center align-items-center flex-column"
style="background-color: rgba(255,255,255,0.3); ">
    <div id="myCarousel" class="carousel slide" data-ride="carousel">
      <!-- Indicators -->
      <ol class="carousel-indicators">
        <li data-target="#myCarousel" data-slide-to="0" class="active"></li>
        <li data-target="#myCarousel" data-slide-to="1"></li>
        <li data-target="#myCarousel" data-slide-to="2"></li>
      </ol>

      <!-- Wrapper for slides -->
      <div class="carousel-inner">
        <div class="item active">
          <div class="slider">
            
            
            
          </div>
        </div>
      </div>

    </div>
    <div class="px-md-5" style="padding: 50px; text-align: center;">
      UpFit! is a fitness application that helps users track their workouts, set
      fitness goals, and monitor their progress with its user-friendly interface and comprehensive
      tracking tools. UpFit! is a great choice for anyone looking to take control of their fitness
      journey.

    </div>
    <div class="login-register">
      <h3 style="color: white;">Login or Register to use the application!</h3>
      <div>
        @Html.ActionLink("Log In", "Login", "Users", new { area = "" }, new { @class
= "btn btn-primary" })
        @Html.ActionLink("Register", "Register", "Users", new { area = "" }, new {
@class = "btn btn-primary" })
      </div>
      <div style="padding: 50px; ">
        <h3 style="color: white;">Are you a coach or admin?</h3>
        <div>
          @Html.ActionLink("Log In as Coach", "Login", "Coach", new { area = "" },
new { @class = "btn btn-primary" })
          @Html.ActionLink("Log In as Admin", "Login", "Admins", new { area = "" },
new { @class = "btn btn-primary" })
        </div>
      </div>
    </div>
  </div>
</main>

```

## POSTING VIDEOS

The provided code snippet showcases two methods within the **Coach Controller**: "AddVideo" and "VideoList". The "AddVideo" method is an HTTP POST action that handles the **uploading of a video file**. The "VideoList" method is an HTTP GET action that **retrieves a list of videos from the database**. It also creates a new instance of the "CodeFirstDb" context. These methods work together to handle video uploads and display the list of uploaded videos for the coach interface.

```
[HttpPost]
public ActionResult AddVideo(HttpPostedFileBase video)
{
    using (CodeFirstDb db = new CodeFirstDb())
    {
        if (video.FileName != null)
        {
            video.SaveAs(Server.MapPath("/Videofiles/" + video.FileName));
            Video uploadVideo = new Video();
            {
                uploadVideo.Vname = video.FileName.Split('.')[0];
                uploadVideo.Vpath = "/Videofiles/" + video.FileName;
            };
            db.videos.Add(uploadVideo);
            db.SaveChanges();
            return RedirectToAction("VideoList");
        }
    }
    return View("VideoList");
}

[HttpGet]
public ActionResult VideoList()
{
    using (CodeFirstDb db = new CodeFirstDb())
    {
        List<Video> Videolist = new List<Video>();
        foreach (Video video in db.videos)
        {
            Video dbVideo = new Video();
            {
                dbVideo.Vname = video.Vname;
                dbVideo.Vpath = video.Vpath.ToString();
            };
            Videolist.Add(dbVideo);
        }
        return View(Videolist);
    }
}
```

## IMPORTANT SECTIONS FROM ADMIN CONTROLLER

```

// POST: Admins/Create
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create([Bind(Include = "AdminID,UserName,Password")] Admin admin)
{
    if (ModelState.IsValid)
    {
        db.admins.Add(admin);
        db.SaveChanges();
        return RedirectToAction("Index");
    }

    return View(admin);
}

// GET: Admins/Delete/5
public ActionResult Delete(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Admin admin = db.admins.Find(id);
    if (admin == null)
    {
        return HttpNotFound();
    }
    return View(admin);
}

// POST: Admins/Edit/5
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit([Bind(Include = "AdminID, UserName, Password")] Admin admin)
{
    if (ModelState.IsValid)
    {
        db.Entry(admin).State = EntityState.Modified;
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    return View(admin);
}

// POST: Admins/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public ActionResult DeleteConfirmed(int id)
{
    Admin admin = db.admins.Find(id);
    db.admins.Remove(admin);
    db.SaveChanges();
    return RedirectToAction("Index");
}

```

```

// POST: Admins/AddUser
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult AddUser(User user)
{
    if (ModelState.IsValid)
    {
        db.users.Add(user);
        db.SaveChanges();
        return RedirectToAction("ManageUsers");
    }

    return View(user);
}

// GET: Admins/DeleteUser/5
public ActionResult DeleteUser(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    User user = db.users.Find(id);
    if (user == null)
    {
        return HttpNotFound();
    }
    return View(user);
}

[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult CreateFoods(Food food)
{
    if (ModelState.IsValid)
    {
        var foodType = db.foodTypes.FirstOrDefault(ft => ft.ID_Type == food.type);
        if (foodType != null)
        {
            food.type = foodType.ID_Type;

            db.foods.Add(food);
            db.SaveChanges();
            return RedirectToAction("ManageFoods");
        }
        else
        {
            ModelState.AddModelError("", "Invalid food type.");
        }
    }

    ViewBag.FoodTypes = db.foodTypes.ToList();

    return View(food);
}

```

## IMPORTANT SECTIONS FROM USERS CONTROLLER

```

[HttpPost]
public ActionResult Login(User user)
{
    using (CodeFirstDb db = new CodeFirstDb())
    {
        var usr = db.users.SingleOrDefault(u => u.UserName == user.UserName &&
            u.Password == user.Password);
        if (usr != null)
        {
            Session["UserID"] = usr.UserID.ToString();
            Session["UserName"] = usr.UserName.ToString();
            Session["SubscriptionType"] = usr.SubscriptionTypeFK.ToString();
            Session["KcalDaily"] = usr.KcalDaily;
            return RedirectToAction("LoggedIn");
        }
        else
        {
            ModelState.AddModelError("", "Incorrect username or password.");
        }
    }
    return View();
}

public ActionResult LoggedIn()
{
    if (Session["UserID"] != null)
    {
        int userId = Convert.ToInt32(Session["UserID"]);
        MacroCalculations(userId);
        // Retrieve the user from the database based on the UserID
        User user = db.users.Find(userId);

        if (user != null)
        {
            return View(user);
        }

        // If the user is not logged in or not found in the database, redirect to the login
        return RedirectToAction("Login");
    }
}

```

```

// POST: Users/Register
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Register([Bind(Include =
"UserID,SubscriptionTypeFK,UserName,Password,FirstName,LastName,Gender,Age,Height,Weight,KcalDaily,Lifestyle")] User user)
{
    if (user.Gender == "M")
    {
        user.KcalDaily = (int)Math.Round(user.Lifestyle * (66 + (13.7 * user.Weight) + (5
* user.Height) - (6.8 * user.Age)));
    }
    else
    {
        user.KcalDaily = (int)Math.Round(user.Lifestyle * (65 + (9.5 * user.Weight) +
(1.8 * user.Height) - (4.7 * user.Age)));
    }
    if (ModelState.IsValid)
    {
        db.users.Add(user);
        db.SaveChanges();
        return RedirectToAction("Login");
    }

    return View(user);
}

[HttpGet]
public ActionResult VideoList()
{
    using (CodeFirstDb db = new CodeFirstDb())
    {
        List<Video> Videolist = new List<Video>();
        foreach (Video video in db.videos)
        {
            Video dbVideo = new Video();
            {
                dbVideo.Vname = video.Vname;
                dbVideo.Vpath = video.Vpath.ToString();
            };
            Videolist.Add(dbVideo);
        }
        return View(Videolist);
    }
}

```

## DB CONTEXT

The provided code snippet represents the definition of a class named "CodeFirstDb" within the "UpFit\_\_main.Models" namespace. This class inherits from the "DbContext" class, which is a part of Entity Framework.

Within the "CodeFirstDb" class, there are several properties defined, each corresponding to a table in the database. These properties use the "DbSet<T>" class to represent database tables, where "T" is the corresponding model class for each table.

The properties include "users" for storing user information, "admins" for storing admin credentials, "foodTypes" for storing types of food, "foods" for storing detailed information about specific foods, "mealTypes" for storing types of meals (e.g., breakfast, lunch, dinner), "meals" for storing individual meal records, "coaches" for storing coach information, and "videos" for storing video details.

By defining these properties within the "CodeFirstDb" class, it allows the application to interact with the corresponding database tables using Entity Framework's capabilities, such as querying, inserting, updating, and deleting data.

Overall, the "CodeFirstDb" class acts as a bridge between the application and the underlying database, providing access to the necessary tables and facilitating data operations.

```
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Web;

namespace UpFit__main.Models
{
    public class CodeFirstDb : DbContext
    {
        public DbSet<User> users { get; set; }
        public DbSet<Admin> admins { get; set; }
        public DbSet<FoodType> foodTypes { get; set; }
        public DbSet<Food> foods { get; set; }
        public DbSet<MealType> mealTypes { get; set; }
        public DbSet<Meal> meals { get; set; }
        public DbSet<Coach> coaches { get; set; }
        public DbSet<Video> videos { get; set; }
    }
}
```

## DATABASE IMPLEMENTATION

The database was automatically generated using migrations, provided by **Entity Framework**. We set the “**Automatic Migrations**” property to true. With automatic migrations enabled, whenever there are changes to the model classes, such as adding new entities, modifying properties, or updating relationships, Entity Framework automatically generates the necessary migration scripts. This process allows for seamless updates and changes to the database schema as the application evolves. To ensure the integrity and accuracy of the database, it was accessed and verified using **SQL Server Management Studio (SSMS)**.

It's important to note that the **Connection string** used to establish a connection between the application and the database needs to be updated on each computer where the application is deployed or run. The connection string contains information such as the server name, authentication details, and database name. By updating the connection string, the application can establish the necessary connection to the correct database instance.

## Model View Controller Application

### FOODS TABLE

TIMI\MSSQLSERVER0...oject - dbo.Foods		TIMI\MSSQLSERVER...bo.UploadClasses					TIMI\MSSQLSERVER...Project - Diagram			TIMI\MSSQLSERVER0...oject - dbo.Meals			
	foodID	type	name	calories	proteins	fats	carbs	fibers	vitamin_A	vitamin_B	vitamin_C	vitamin_D	vitamin_E
	1	1	Applee	52	0.3	0.2	14	2.4	0.001	0.0001	0.05	0.0001	0.02
	2	1	Orange	43	0.9	0.1	9	2.3	0.00225	1E-05	53.2	0.00012	0.02
	3	1	Banana	96	1.2	0.2	23	2.6	0.0064	0.00012	0.087	1E-05	0.01
	4	1	Strawberry	32	0.7	0.3	7.7	2	0.0012	1E-05	0.588	0.0001	0.03
	5	2	Carrot	41	0.9	0.2	9.6	2.8	0.00167	0.0001	0.059	0.0001	0.07
	6	2	Broccoli	55	3.7	0.6	11.2	2.6	0.0567	0.0001	0.0892	0.0002	0.083
	7	2	Potato	77	2	0.1	17	2.2	0.003	0.03	0.001	0.436	0.056
	8	2	Tomato	18	0.9	0.2	3.9	1.2	0.004	0.04	0.0006	12.3	11.1
	9	2	Cauliflower	25	1.9	0.3	4.9	2	0.458	0.006	9.7	9.4	0.0003
	10	3	Beef	250	26	17	0	0	0.945	1.12	0.43	0.0001	0.01
	11	3	Chicken	165	31	3.6	0	0	0.54	1.23	0.003	0.0001	0.3247
	12	3	Pork	143	25	4	0	0	0.45	0.005	0.2384	0.437	0.3243
	13	3	Lamb	258	25	17	0	0	0.005	9.23	0.0056	0.478	0.92374
	14	3	Turkey	189	29	3.6	0	0	0.067	0.3247	0.04385	0.87345	0.0034
	15	4	Salmon	206	22	13	0	0	0.33	0.045	0.3874	0.034	1.47
	16	4	Shrimp	85	20	1	0	0	0.045	0.0304	0.001	0.45	3.38924
	17	4	Tuna	144	30	1	0	0	0.034	0.056	3.84375	0.056	1.34
	18	4	Cod	82	18	0.6	0	0	0.001	2.4895	7.34	0.340004	4.93284
	19	4	Lobster	90	19	1	1	0	4.2958	0.0001	0.0566	0.0001	0.00065
	20	5	Milk	60	3.2	3.6	0	0	0.046	1.47	3.48	2.4837	0.987384
	21	5	Yoghurt	59	3.5	3.3	3.6	0	8.23847	0.1284	1E-05	0.056	0.8745
	22	5	Cheese	403	24.9	33.1	1.3	0	4.23847	0.33	0.032874	4.32	0.9843
	23	5	Butter	717	0.9	81.1	0.1	0.045	0.003	0.003432	3.984	1E-05	0.0274
	24	6	Rice	130	2.7	0.3	28.2	0.4	8.4	0.068	0.324	9.293	0.03
	25	6	Oats	389	16.9	6.9	65.4	10.6	1E-05	0.003	2.1	0.381279	0.067
	26	6	Wheat	340	14.7	2.2	71.2	10.7	3.33	0.4897	0.0943003	1.1	0.274
	27	7	Almonds	579	21.1	49.9	21.7	12.1	0.934	0.0045	0.374	0.03874	0.045
	28	7	Walnuts	654	15.2	65.2	13.7	6.8	1.2	0.035	0.005	0.0773	1E-05
	29	8	Bread	265	9.7	2.9	51.2	2.4	0.056	0.045	1E-05	0.03942	0.97457834
	30	8	Croissant	406	6.1	22.5	45.8	1.7	3.33	NULL	0.0034	2.78	0.83475

### USERS TABLE

TIMI\MSSQLSERVER0...oject - dbo.Users												
	UserID	Subscriptio...	UserName	Password	FirstName	LastName	Gender	Age	Height	Weight	KcalDaily	Lifestyle
▶	11	0	Clemi33	clementina	Clementina	Teodor	F	29	177	56	1208	1.55
	12	0	Raluca	raluca	Raluca	Nemes	F	23	172	55	1223	1.55
	13	1	Timea	timeaa	Timea	Nagy	F	22	172	51	907	1.2
	14	1	Adela90	parolasiteu...	Adela	Ionescu	F	30	165	58	1265	1.375
	15	1	Radu78	Secur3P@ss	Radu	Popescu	M	43	180	75	2934	1.725
	16	0	IoanaDumit...	ioanadumitr...	Ioana	Dumitrescu	F	25	170	60	1132	1.375
	17	1	Lucian55	galaxys12	Lucian	Stanescu	M	55	175	80	1996	1.2
	18	0	AnaPetrescu2	SunshIn3GlrI	Ana	Petrescu	F	26	168	62	1293	1.55
	19	0	Paul23	B00kL0v3r!	Paul	Mihai	M	23	182	70	3068	1.725
	20	1	Cristina87	upfitaplicati...	Cristina	Rad	F	34	163	55	992	1.375
	21	1	Andrei3	stradaJupiter	Andrei	Popa	M	24	193	94	3718	1.725
	22	0	ElenaStef	stefanescu2	Elena	Stefanescu	F	27	167	58	1086	1.375
	23	0	Vasile67	Coffelover	Vasile	Iancu	M	56	172	78	2219	1.375
	24	0	AnaMaria19	anamaria19	AnaMaria	Gheorghe	F	19	170	57	1132	1.375
	25	1	Roxyy	Roxanaroxy	Roxana	Stefanescu	F	22	171	55	950	1.2
	26	1	FlorinN	florinneagu	Florin	Neagu	M	24	182	67	2683	1.55
	27	1	Larisa	dancingque...	Larisa	Ionita	F	31	169	58	1200	1.55
	28	0	ElenaLazar1	elenalazar1	Elena	Lazar	F	22	172	51	907	1.2
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



## APPLICATION TESTING

The testing phase of the UpFit application begins with thorough documentation to ensure a structured and organized approach. The testing objectives are clearly defined, encompassing the verification of functionality, performance, and reliability. The scope of testing outlines the specific areas of the application that will be tested, including key features, user roles, and critical workflows.

For this manual testing approach, a comprehensive set of test cases is prepared, covering various scenarios and functionalities. Each test case includes detailed steps, expected results, and any necessary test data or prerequisites. Additionally, comments are provided for reporting and tracking bugs or issues encountered during testing.

Throughout the testing phase, the assigned testers follow the defined test execution plan, adhering to the timeline and documenting any identified issues. The testing process aims to deliver a comprehensive test report summarizing the results, highlighting any issues discovered, and offering recommendations for improvement.

Test Scenario ID	Test Scenario Name	No. Test Cases	Test Cases	PRE-CONDITION	TEST STEPS	POST-CONDITION	EXPECTED RESULT	ACTUAL RESULT	STATUS(PASS/FAIL)
TS_Login	Login Functionality Basic User	4	Enter <b>valid</b> User Name and <b>valid</b> Password	Need a unique username	1.Enter UserName 2.Enter Password 3.Click'Login'button	The user is on the Basic User Page	Successful login! Message "Hello!" is shown	Successful login! Message "Hello!" is shown	Pass
			Enter <b>valid</b> User Name and <b>invalid</b> Password				Message "The Password must be at least 6 characters long." is shown	"Login failed due to incorrect password or password"	Pass
			Enter <b>invalid with special characters</b> User Name and <b>valid</b> Password				Message "Username can only contain letters, numbers, and underscores." is shown	"Login failed due to incorrect password or password"	Pass
			Enter <b>invalid</b> User Name and <b>invalid</b> Password				Message "Incorrect username or password" is shown	"Login failed due to incorrect password or password"	Pass
	Login Functionality Premium User	4	Enter <b>valid</b> User Name and <b>valid</b> Password	Need a unique username	1.Enter UserName 2.Enter Password 3.Click'Login'button	The user is on the Premium User Page	Successful login! Message "Hello!" is shown	Successful login! Message "Hello!" is shown	Pass
			Enter <b>valid</b> User Name and <b>invalid</b> Password				Message "The Password must be at least 6 characters long." is shown	"Login failed due to incorrect password or password"	Pass
			Enter <b>invalid with special characters</b> User Name and <b>valid</b> Password				Message "Username can only contain letters, numbers, and underscores." is shown	"Login failed due to incorrect password or password"	Pass
			Enter <b>invalid</b> User Name and <b>invalid</b> Password				Message "Incorrect username or password" is shown	"Login failed due to incorrect password or password"	Pass
Basic User Page Functionality	Basic User Page	3	<b>Click</b> Add Meal button; <b>valid</b> : Meal type; Food type; Quantity ; Click Create; Click Back	Need an account	1.Click Add Meal 2.Select Meal 3.Select Food 4.Choose Quantity 5.Click Create 6.Click Back	The user is on Basic user page	Message "Hello, username"! User can see his DAILY NUTRIENTS	Successful created meal	Pass
			<b>Click</b> Add Meal button; <b>invalid</b> : Meal type; Food type; Quantity ; Click Create			The user is on the Add meal Page	Messages: "The Meal Type field is required." The Food field is required." The Quantity field is required." is shown.	Failed to create meal	Pass
			<b>Click</b> Add Meal button; <b>valid</b> : Meal type; Food type; <b>Negative</b> : Quantity ; Click Create			The user is on the Add meal Page	Message: "Quantity must be a positive value."	Failed to create meal	Pass
Premium User Page Functionality	Premium User Page	4	<b>Click</b> Add Meal button; <b>valid</b> : Meal type; Food type; Quantity ; Click Create; Click Back	Need an account	1.Click Add Meal 2.Select Meal 3.Select Food 4.Choose Quantity 5.Click Create 6.Click Back	The user is on Premium user page	Message "Hello, username"! User can see his DAILY NUTRIENTS	Successful created meal	Pass
			<b>Click</b> Add Meal button; <b>invalid</b> : Meal type; Food type; Quantity ; Click Create			The user is on the Add meal Page	Messages: "The Meal Type field is required." The Food field is required." The Quantity field is required." is shown.	Failed to create meal	Pass
			<b>Click</b> Add Meal button; <b>valid</b> : Meal type; Food type; <b>Negative</b> : Quantity ; Click Create			The user is on the Add meal Page	Message: "Quantity must be a positive value."	Failed to create meal	Pass
			<b>Click</b> on Workouts button			The user is on the Videolist Page	See videos from coach	Videos from coach are shown	Pass

## Model View Controller Application

TS_Register	Verify Register Functionality	11	Choose Subscription Type; Enter valid: UserName;Password;FirstName; LastName; Gender;Age; Height; Weight; Choose LifeStyle	Internet access	1.Subscription Type 2.UserName 3.Password 4.FirstName 5.LastName 6.Gender 7.Age 8.Height 9.Weight 10.Select 11.Choose LifeStyle	The user is on the Login page	Successful created account!	Successful created account!	Pass
			Choose Subscription Type; Enter invalid with special characters UserName; valid: Password; FirstName; LastName; Gender;Age; Height; Weight;Choose LifeStyle				Message "Username can only contain letters, numbers, and underscores." is shown	"Register failed due to incorrect Username"	Pass
			Choose Subscription Type; Enter valid UserName; invalid Password; valid: FirstName; LastName; Gender; Age; Height; Weight; Choose LifeStyle				Message "The Password must be at least 6 characters long." is shown	"Register failed due to incorrect Password"	Pass
			Choose Subscription Type; Enter valid UserName; valid Password;invalid with special characters FirstName; valid: LastName; Gender; Age; Height; Weight; Choose LifeStyle				Message "First Name can only contain letters." is shown	"Register failed due to incorrect FirstName"	Pass
			Choose Subscription Type; Enter valid UserName; valid Password; valid FirstName; invalid with special characters LastName; valid: Gender; Age; Height; Weight; Choose LifeStyle				Message "Last Name can only contain letters." is shown	"Register failed due to incorrect LastName"	Pass
			Choose Subscription Type; Enter valid UserName; valid Password; valid FirstName; valid LastName; valid: Gender; invalid Age; valid: Height; Weight;Choose LifeStyle				Message "Age must be between 1 and 150" is shown	"Register failed due to incorrect Age"	Pass
			Choose Subscription Type; Enter valid UserName; valid Password; valid FirstName; valid LastName; valid: Gender; valid Age; valid: Height; invalid: Weight;Choose LifeStyle				Message "Weight must be between 10 and 300." is shown	"Register failed due to incorrect Weight"	Pass
			Choose Subscription Type; Enter valid UserName; valid Password; valid FirstName; valid LastName; valid Gender; valid Age; invalid: Height; valid: Weight;Choose LifeStyle				Message "Height must be between 50 and 300" is shown	"Register failed due to incorrect Height"	Pass
			Choose Subscription Type; Enter valid UserName; valid Password; valid FirstName; valid LastName; invalid Gender; valid Age; valid: Height; valid: Weight;Choose LifeStyle				Message "Gender must be 'F' or 'M.'" is shown	"Register failed due to incorrect Age"	Pass
			No choose Subscription Type; Enter valid UserName; valid Password; valid FirstName; valid LastName; valid Gender; valid Age; valid Height; valid: Weight;Choose LifeStyle				Message "Subscription Type is required."	"Register failed due Subscription Type"	Pass
			Choose Subscription Type; Enter valid UserName; valid Password; valid FirstName; valid LastName; valid Age; valid Height; valid: Weight; No choose LifeStyle				Message "The Lifestyle field is required."	"Register failed due Select LifeStyle"	Pass
			Press "Register " button		Click on "Register" button shown	The user is on the Login page	Message "LOG IN WITH USER ACCOUNT". + Register in DataBase	"Register succeeded"	Pass
			Press "Back" button		Click on "Back" button shown	The user is back on the main page	Message "LOGIN OR REGISTER TO USE THE APPLICATION!"	Button "back" succeeded	Pass

TS_Login	Verify Login Functionality	5	Enter <b>valid</b> User Name and <b>valid</b> Password	Need an account	1.Enter UserName 2.Enter Password 3.Click'Login'button	The user is on the Login page	LOGGED IN AS ADMIN Welcome to your admin account! Select the operation you want to perform!	Successful login!	Pass
			Enter <b>valid</b> User Name and <b>invalid</b> Password				Message "Incorrect username or password." is shown	"Incorrect username or password"	Pass
			Enter <b>invalid</b> User Name and <b>valid</b> Password				Message "Incorrect username or password." is shown	"Incorrect username or password"	Pass
			Enter <b>invalid</b> User Name and <b>invalid</b> Password				Message "Incorrect username or password." is shown	"Incorrect username or password"	Pass
			Enter <b>invalid with special characters</b> UserName and <b>valid</b> Password				Message "UserName can only contain letters, numbers, and underscores	"Incorrect username or password."	Fail
TS_HomePage	Verify Buttons on Index page Functionality	1	Click Home button functionality	Need an account	Click on Home button	Go back on the principal page	Message"LOGIN OR REGISTER TO USE THE APPLICATION!"	Go on Home page succesfuly	Fail
Ts_ManageUsers	Verify Functionality on Manage Users page	4	Create new user button functionality	Need an account	1.Subscription Type 2.UserName 3.Password 4.FirstName 5.LastName 6.Gender 7.Age 8.Height 9.Weight 10.KcalDaily	The Admin is on the Add User page	The users were created!	Successful created!	Pass
			Back button functionality		1.Click "Back" button	The Admin is on the principal page of Admin	Message"LOGGED IN AS ADMIN"	Successful login!	Pass
			Delete button functionality		1.Click "Delete" button	The Admin is on the Delete User Page	Message "ARE YOU SURE YOU WANT TO DELETE THIS USER?"	Successful deleted!	Pass
			Details button functionality		1.Click "Details" button	The Admin is on the Details User Page	See details about the selected user	Successful details!	Pass
Ts_ManageAdmins	Verify Functionality on Manage Admins page	3	Create new admin button functionality	Need an account	1.UserName 2.Password 3.Create	The Admin is on the principal page of Admin	Message"LOGGED IN AS ADMIN"	Successful login!	Pass
			Delete button functionality		1.Click "Delete" button	The Admin is on the Delete User Page	Message "ARE YOU SURE YOU WANT TO DELETE THIS USER?"	Successful deleted!	Pass
			Back button functionality		1.Click "Back" button	The Admin is on the principal page of Admin	Message"LOGGED IN AS ADMIN"	Successful login!	Pass
Ts_ManageFoods	Verify Functionality on Manage Foods page	2	Click create food button; Enter <b>Valid</b> : type food;Name;Calories;Proteins;Fats;Carbohydrates;Fibers;Vitamina A;Vitamina B; Vitamina C; Vitamina D; Vitamina E; Click create	Need an account	1.Click Create food button 2.Choose type food 3.Name 4.Calories 5.Proteins 6.Fats 7.Carbohydrates 8.Fibers 9.Vitamina A 10.Vitamina B 11.Vitamina C 12.Vitamina D 13.Vitamina E 14.Click Create	The Admin is on the create food pace	Successful created food! + Insert in DataBase	Successful created!	Fail
			Back button functionality		1.Click "Back" button	The Admin is on the principal page of Admin	Message"LOGGED IN AS ADMIN"	Successful login!	Pass
Ts_ManageFoodTypes	Verify Functionality on ManageFoodTypes page	5	Click Create button; Enter <b>valid</b> name	Need an account	1.Enter Name 2. Click Create	The Admin is on the Manage Food Types page	See the food types created	Successful created!	Pass
			Click Create button; Enter <b>invalid with spacial characters</b> name				Can't write names with spacial characters;	Successful created!	Fail
			Edit button functionality		1.Click Edit 2.Save	The Admin is on the Edit FoodType page	Save succesfully changes!	Edit sucesfuly!	Pass
			Delete button functionality		1.Click on Delete button	The Admin is on the Delete FoodType Page	Deleted succesful!	No execut	Fail
			Back button functionality		1.Click "Back" button	The Admin is on the principal page of Admin	Message"LOGGED IN AS ADMIN"	Successful login!	Pass
Ts_LogOut	Verify Functionality on LogOut Page	1	Log Out button functionality		1.Click on Log Out button	The Admin is on the principal page	Message"LOGIN OR REGISTER TO USE THE APPLICATION!"	Logged Out succesful	Pass
TS_Login	Verify Login Functionality Coach	4	Enter <b>valid</b> User Name and <b>valid</b> Password	Need a unique username	1.Enter User Name 2.Enter Password 3.Click'Login'button	The user is on the Coach Page	Successful login! Message "VIDEO FILES UPLOAD!" is shown	Successful login! Message "VIDEO FILES UPLOAD!" is shown	Pass
			Enter <b>valid</b> User Name and <b>invalid</b> Password				Message "The Password must be at least 6 Characters long." is shown	"Login failed due to incorrect password or password"	Pass
			Enter <b>invalid with special characters</b> User Name and <b>valid</b> Password				Message "Username can only contain letters, numbers, and underscores." is shown	"Login failed due to incorrect password or password"	Pass
			Enter <b>invalid</b> User Name and <b>invalid</b> Password				Message "Incorrect username or password" is shown	"Login failed due to incorrect password or password""	Pass
Videos_Upload	Verify Upload Video Functionality	4	Choose file	Need an account	1.Click Choose file button	The user is on the Coach Page	The coach can choose the videos he wants to upload	Successful choosen videos	Pass
			Choose video files		1.Click Choose file button		Can't choose other types of files others than mp3/mp4	Successful choosen files	Warning!
			No choose file		1.Don't press "Choose file" button 2.Press upload video files		Message "Nu ai ales niciun fisier"	Message "Nu ai ales niciun fisier"	Pass
			Click Upload files		1.Click Upload video files button		The coach upload videos on page	Successful uploaded videos	Pass

## CONSLUSION

In conclusion, UpFit is more than just a fitness application. It embodies a vision of empowerment, a commitment to supporting individuals on their journey towards a healthier lifestyle. With its user-friendly interface and comprehensive features, UpFit strives to inspire users to take charge of their well-being and make informed choices about their diet and fitness.

The application's ability to track meals, monitor calorie intake, and help users achieve their fitness goals goes beyond mere functionality. It resonates with the deep desire we all have to live healthier, more fulfilling lives. UpFit serves as a trusted companion, providing the tools and information needed to transform aspirations into tangible results.

Through its intuitive design and robust capabilities, UpFit aims to instill a sense of motivation, encouraging users to make positive changes and embrace a balanced approach to their health. It aspires to be the catalyst that empowers individuals to unlock their full potential and experience the transformative power of a healthy lifestyle.

UpFit is more than just an application; it's a partner on the journey towards personal well-being, offering support, guidance, and the belief that each individual has the capacity to achieve their goals and live their best life.