



**SILESIA UNIVERSITY OF TECHNOLOGY**  
**FACULTY OF AUTOMATIC CONTROL, ELECTRONICS AND COMPUTER**  
**SCIENCE**

**Internet Technologies – project work**

**Strengthening**

**Authors: Remigiusz Abramik and Szymon Abramik**

**Year, semester: 2022 year, 5 semester,**

**group, section: 3 group, 4 section.**

**Project supervisor: dr inż. Jacek Loska**

**Gliwice, December 2022**

## Table of Content

1.	Introduction.....	3
2.	Aim and scope of the project .....	5
3.	Schedule .....	6
3.1.	Schedule approved at the beginning.....	6
3.2.	Schedule reflecting actual work .....	7
4.	Software implementations.....	8
4.1.	Creating basic website structure .....	8
4.2.	Creation of the login and registration structure .....	13
4.3.	Coding applications responsible for diets and trainings .....	18
4.4.	Programming fully interactive forum application .....	23
4.5.	Creating basic BMI calculator app.....	29
4.6.	Designing home page .....	32
5.	Summary.....	41
6.	Sources. ....	42
6.1.	Used libraries and frameworks. ....	42
6.2.	Sources from which we learned.....	<b>Error! Bookmark not defined.</b>

## 1. Introduction

Many people these days have problems with maintaining a healthy figure and a healthy lifestyle. When people are busy, they find it difficult to find the time and motivation to plan an active lifestyle and healthy meals.

Our project idea is to make it easier for such people to take the first steps in a healthy lifestyle and proper exercise without the need for special equipment or even leaving home. Currently people think that to perform exercises you need professional equipment, but this is not necessarily true, professional equipment is not necessarily useful for someone who does not perform advanced exercises on a daily basis, and many exercises performed on machines are not recommended for people without proper technique and developed muscles.

Often in order to take care of your figure or even to be healthier, you also need to rethink your daily eating habits. It is believed that you can lose weight or reduce weight simply by eating less and this is true, but it is much more effective to know the ingredients of the food we eat and to select our daily diet according to our needs for example to gain muscle, reduce weight or just simply being healthier. Unfortunately, few people have time to learn the number of calories in a meal or the content of protein, fats and carbohydrates.

That's why we plan our project to help people who do not have enough free time or do not want to learn all the rules and tips associated with them. On our website there will be sets of exercises tailored to each person with different experience, different physiques, different ages and exercise intensity. Each person will be able to find a specialized diet and tips on how to eat in order to achieve a specific goal, such as weight loss or weight gain, while maintaining a high percentage of muscle tissue. The site will help users to maintain their progress in the exercises they do and help them when they do them.

Anyone who does not know the proper technique for a given exercise need not worry, as the site will make sure that each exercise is accompanied by an explanation of how to perform it correctly and what to avoid. Exercises will be scheduled to perform all series on one day, and the plan will be divided into days with a scheduled day for breaks.

Exercises that require time to complete, the site will show a stopwatch that will help you control the time of exercise and repetition exercises will show the number of repetitions. Diets will also be tailored to our goal and our situation. In summary, our site is designed to help people who do not have the opportunity to take care of these things themselves, so that they can take care of themselves and develop in the field they want in their free time.

It's hard to maintain a job as well as an active and healthy lifestyle in today's busy society especially if you have a problem which is taking care of a loved one with problems, you have a family to take care of or you have a job that you have to take home with you, that's why we believe that our website is crucial in our world to use our free time in the most efficient way, so that exercising, cooking and working will not be the only things we do in our lives.

## **2. Aim and scope of the project**

First of all, on our site you will be able to create an account with a username and password, and each account will have its own profile with its own information. Our site will deal with the selection of exercises and diet depending on the needs of the user.

The site will provide training plans to choose from basic to advance for different parts of the body to choose from when selecting an exercise. This is to increase the intensity of the exercises for people who are already familiar with the exercises and have some basics. The site will help the user perform the exercises, displaying the exercise technique and number of repetitions, while the endurance exercises will be accompanied by a countdown timer to help keep track of the length of the exercise.

Exercises will be performed in batches and displayed in the correct order. Exercise plan will be set in days and for each day there will be set exercises or lack of them, the user's progress will be recorded and will follow the plan. Our plan is that the user will always have access to his/her progress in his/her own profile, where all the information he/she needs will be displayed, so that he/she can easily access it and plan his/her next activities and for the greatest comfort of the user in navigating the site.

Completed workouts will also be shown on user profile. The site will also offer a variety of diets that can be selected according to our requirements and dietary needs. Diets will be planned for long periods of 7 days a week, with at least 3 meals a day. The site will also be in lighter colours for the sake of the user's eyes and will take care of its aesthetic qualities which are as important in our competitive world as the functionality of the site.

### **3. Schedule**

#### **3.1. Schedule approved at the beginning**

1. Presentation and analysis of the project.

2. Planning the structure and content of the site and set up page and put it on the GitHub.

About how many pages we will need and how will we access them from the main page, if we will need moving parts of the site with CSS and what language we will need for what.

3. Creation of the login and registration structure.

This will allow users to make account and have access to their profiles after logging in.

4. Supplementing the site with a large amount of content (mainly graphically).

We will plan about design of the site and maybe pages to help with exercises (probably with html and CSS).

5. Setting up and operating the database with information about our users.

Setting data base with information about our users they information our progress with training and other profile information.

6. Coding of the backend X 2.

Developing not visible to us site functionality of web application (probably in python with Django library).

8. Programming comment system .

Making function that allows our users comment our content.

9. Refactoring and testing of our site.

10. Preparing for presentation and finishing the project.

### **3.2. Schedule reflecting actual work**

1. Presentation and analysis of the project.

2. Planning the structure and content of the site and set up page and put it on the GitHub.

About how many pages we will need and how will we access them from the main page, if we will need moving parts of the site with CSS and what language we will need for what.

3. Creation of the login and registration structure.

This will allow users to make account and have access to their profiles after logging in.

4. Implementing object with specified attributes for handling data stored in data base and then showed on our site.

5. Coding applications responsible for diets and trainings. Figuring out way to properly sending data via query set.

6. Programming fully interactive forum application, with the possibility of creating questions, answering them and agreeing with another answer.

7. Creating basic BMI calculator app.

8. Designing home page with CSS, JavaScript and Bootstrap.

9. Improving appearance for rest of website.

10. Preparing for presentation and finishing the project.

A lot of points have changed. Main reason for that is our lack of knowledge when we were starting the project. When our knowledge started to become wider, we changed some part of code or even rewrite some part completely. Some points needed more time and others could took only couple of hours.

In the end, we strayed far from the original concept, which was more focused on e-learning application and created a fully interactive forum website, which can be managed by administration but also users are allowed to see and widen content of website.

## **4. Software implementations**

### **4.1. Creating basic website structure**

#### **4.1.1. Defining the problem**

Since our goal is to create a website, we first conducted research on how a website is built and what is needed to create one. We came to realize that first we need to choose which backend and frontend technologies will we use. We will need framework which can easily create and operate on many applications. On our website we plan to use applications such as:

- Forum, (application for asking questions and answering them)
- Diets and Trainings, (applications for publishing diets and training created by users)
- BMI Calculator. (speak for itself)

It would be really helpful if our solution will contain completed user login and register system which we plan to implement. Lastly we plan to create sides specific to different users which of course requires login system.

#### **4.1.2. Analysis of possible solutions**

There are couple of technologies possible for our problem. First for backend we can use Django framework.

Django framework has many amenities which can make our work faster and easier but more importantly is based on python language which is pretty easy to understand and use. It has great documentation and lot of resources which we can look through.

Next possible backend technology is Laravel based on PHP language. Laravel has modular packaging system with a dedicated dependency manager, different ways for accessing relational databases, utilities that aid in application deployment and maintenance, and its orientation toward syntactic sugar.

And lastly we considered Express. Express, is a back end web application framework for building RESTful APIs with Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js.

For frontend technologies except basic languages like html, CSS, JavaScript there are a lot of possible technologies to use such as React.js, Angular.js, Vue.js, Flutter, Bootstrap, Ionic, HTML5 Boilerplate, NPM (Node Package Manager) and MeteorGrunt. We will have to select only a couple of them.



### 4.1.3. The proposed solution

Ultimately, to build the structure of the site (backend), we decided to use the python language, more specifically the framework offered by this language - Django, released under a BSD license. Django implements the model-template-view architectural pattern while other two model-view-controller which we find harder to understand. Main reasons for choosing Django than Laravel was easier to use and filter database communication. Express was rejected because of specifical connection with Node.js which is very complex then we initially assumed. Examples of Django-based sites include Pinterest, Instagram or Washington Times. The project is based on python version 3.10.8 and Django 4.1.2.

The Django framework offers many important and ready-made solutions that we can use, such as:

- Automatically generated and necessary administration panel, which should be used,
- Friendly document addresses with the possibility of freely shaping them,
- A simple but functional template system that is readable by both graphic designers and programmers,
- Separation of application logic (view), business logic (model), appearance (templates) and databases,
- Support for multilingual applications (internationalization),
- Very high scalability and performance under load,
- Efficient caching systems, Memcached support,
- It works with Apache through WSGI,
- DRY, or the principle of "don't repeat yourself" in relation to application development (e.g. Django generates the database structure from ordinary Python classes),
- It has a high-level ORM that allows you to easily and safely operate on databases without using SQL,
- Support for security against cross-site request forgery, cross-site scripting, SQL injection, password cracking and other common Internet attacks, most of them enabled by default.

It also has support for multiple databases such as PostgreSQL, SQLite, MySQL and Oracle. For the purposes of our project, a basic database, namely SQLite, will suffice. We can observe the progress of our work on a simple custom server offered by Django.

In addition to Django, we will need minor libraries such as Pillow and font awesome free. Pillow is Python Imaging Library which adds image processing capabilities to your python interpreter. Font awesome free is a library that allows you to use the free icons offered by the Font Awesome website.

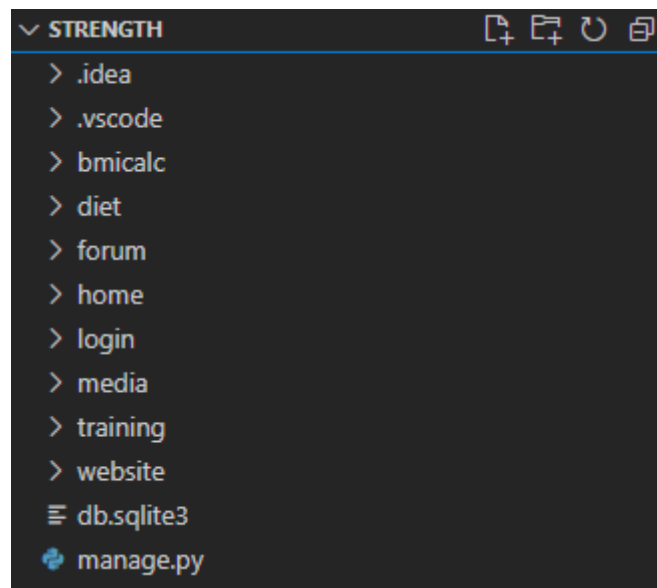
For the appearance of the site (frontend), in addition to basic markup languages such as html, CSS and scripting JavaScript, we used the Bootstrap and jQuery libraries.

Bootstrap is a framework released under the MIT license and developed by Twitter developers. It has a set of useful tools to facilitate the creation of graphical interface of pages. It allows to style forms, navigations bars, buttons etc... It also allows you to dynamically arrange elements on the page and change its layout depending on the width of the page displayed. The structure consists mainly of columns and rows.

The last library we used is jQuery, a JavaScript-based library that allows us to achieve interesting animation effects and dynamic changes on the page thanks to the execution of ajax requests.

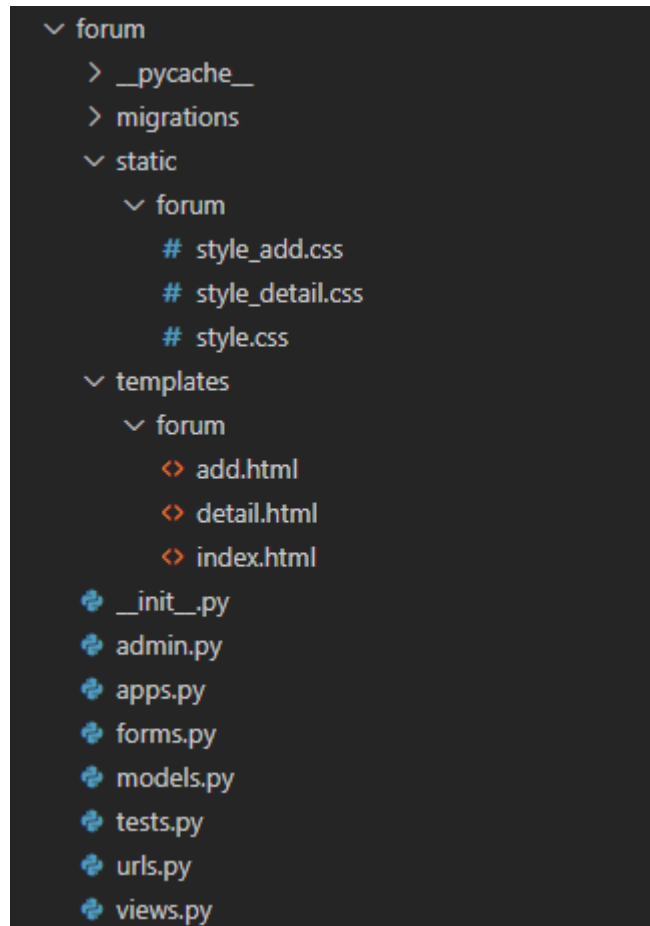
#### 4.1.4. Implementation

First we create project which will contain our page. Project will be done with Visual Studio Code text editor. Next we have to create all necessary applications, we can do it and start a project in command prompt. First we change direction to folder with our project and type **\$ django-admin startproject strengthening**. This will create all necessary files. Next we type **\$ python manage.py startapp appname** for all our application. We will get following file structure as below.



*Project structure.*

There are folders for each application which consist all required files, website is main project folder, db is of course database which is as you can see SQLite3 and lastly manage.py is configuration file which allows us to make various things. Media folder contains pictures.



*Application structure.*

Application have designated places for html, CSS and JavaScript files as you can see above. This type of structure help maintain clear project files structures. There is also visible model-template-view structure via model, views python files and template folder consist of html files. But for connecting our html files and URLs which are being written in browser we have to connect them via urls.py and views.py files. First if we have matching URL then urls.py file tell website what to do.

```
urlpatterns = [
    path('', views.indexView, name='index'),
    path('add/', views.CreatePost.as_view(), name='add'),
    path('like/', views.like_comment, name='like'),
    path('<slug:slug>/delete/', views.DeletePost.as_view(), name='delete'),
    path('<slug:slug>/', views.DetailView.as_view(), name='detail'),
]
```

*Urls.py file inside.*

For example if we have “website/add/” URL it tells to execute CreatePost class which is located in views.py file. In that class or function, based on if we use class or function view, we specify which html file is supposed to open.

If we want to add CSS or JavaScript files to our html we have to tell Django that we want some static files applied here via correct command.

```
{% load static %}
<link href="{% static 'forum/style_add.css' %}" rel="stylesheet" >
```

*Static files attachment command declaration.*

#### **4.1.5. Problems during applications development**

While working on basic structure of our project we encountered problems such as not finding html file property. We spelled wrongly not only some URLs declarations but also functions names and their declarations. It were easy to fix problems as soon as you understood where can problem lie.

Second problem we struggled was dynamic URLs implementations which we needed because our website contains dynamically changing content. Users can add, modify comment or review content at any time so we are forced to use them. We managed to use them in correct manner by using id of objects and adding slug attribute field in model and using Django documentation[\[1\]](#).

Last problem we encountered was that website didn't read our CSS files, we found solution also in Django documentation[\[2\]](#). Not specifying the value of STATIC\_ URL variable to “static/” caused website to searching for CSS files from wrong sources.

## **4.2. Creation of the login and registration structure**

### **4.2.1. Defining the problem**

Most of content on our website will be created by users themselves, easiest way in order to do that is to have register and login system. It will allow website to know who create which content on our website, show pages possible to see only to specific users, contain profile pictures, profile pages and activity of all users. We also need forms used for registering and logging in user.

### **4.2.2. Analysis of possible solutions**

We found two possible ways of dealing with this issue, first option is to create our own logging system with PHP usage. It isn't very complex to implement but it can be really problematic to correctly connect it with Django framework.

Second option is much simpler we can use build in Django user authentication and register system but there are things that are lacking like profile picture, slug field or description for profile picture we want to implement.

As for forms we can resolve it also in couple of ways, we can use build in Django forms, we can directly write all data fields that we need and last option is to create forms.py file and specify there which fields we want to fill.

### **4.2.3. The proposed solution**

Eventually we choose to use build in Django user authentication system because connecting PHP to Django framework proved to be very problematic. To solve problem with Django user model we need to create our own model which will be connected to user model. It will contain data like profile picture, profile page etc.

For forms we implemented the last method with using the forms.py file. Django forms are inflexible and it is hard to change their appearance which is very important factor. Directly writing fields would work but it for sure is not optimal.

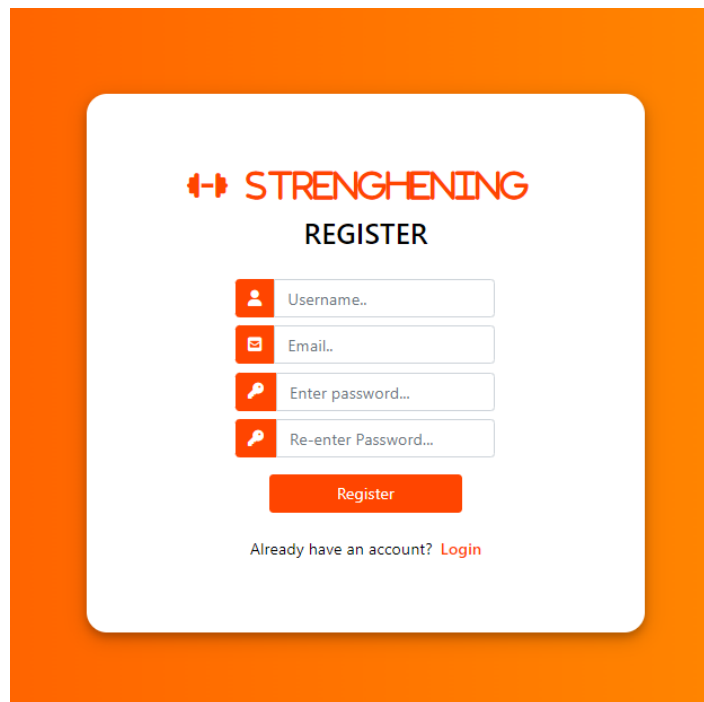
#### 4.2.4. Implementation

First we create model which consist all data that user can't with one to one relationship between them.

```
class Profile(models.Model):
    user = models.OneToOneField(User, null=True, on_delete=models.CASCADE)
    image = models.ImageField(default="profile_pics/default.png", upload_to='profile_pics')
    description = models.TextField(max_length=1000, default="You didn't change your description yet!")
    slug = models.SlugField(unique=True, blank=True, null=True)
```

*Profile model in models.py file.*

Next up is the form for us to register (with frontend already done with CSS and html):



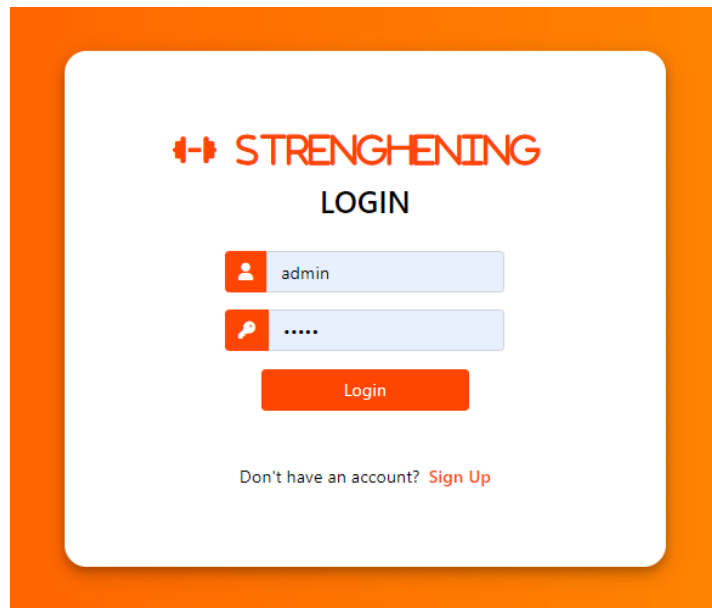
*Register form apperance.*

On the login page, you can see the site logo, which can lead us to the home page, as no login is required to enter the site. To register a user we are required to provide username, email and password. Button „Register” submit and save form. After saving form we are redirected to login page. If some data is incorrect you will get appropriate error. The fields in form are selected via forms.py class that we can specify.

```
class CreateUserForm(UserCreationForm):  
    class Meta:  
        model = User  
        fields = ['username', 'email', 'password1', 'password2']
```

*Register form in forms.py which chooses required fields.*

Page look pretty much the same with login form but with less data to write.



*Login form apperance.*

Our user can also logout by also accesing correct ulr. But immedietialy after accesing it he gets send back to home page without seeing actual page, because it doesn't exist and url is only used for loging out user.

```
# Url used for logging out  
path('logout/', views.logoutUser, name='logout'),
```

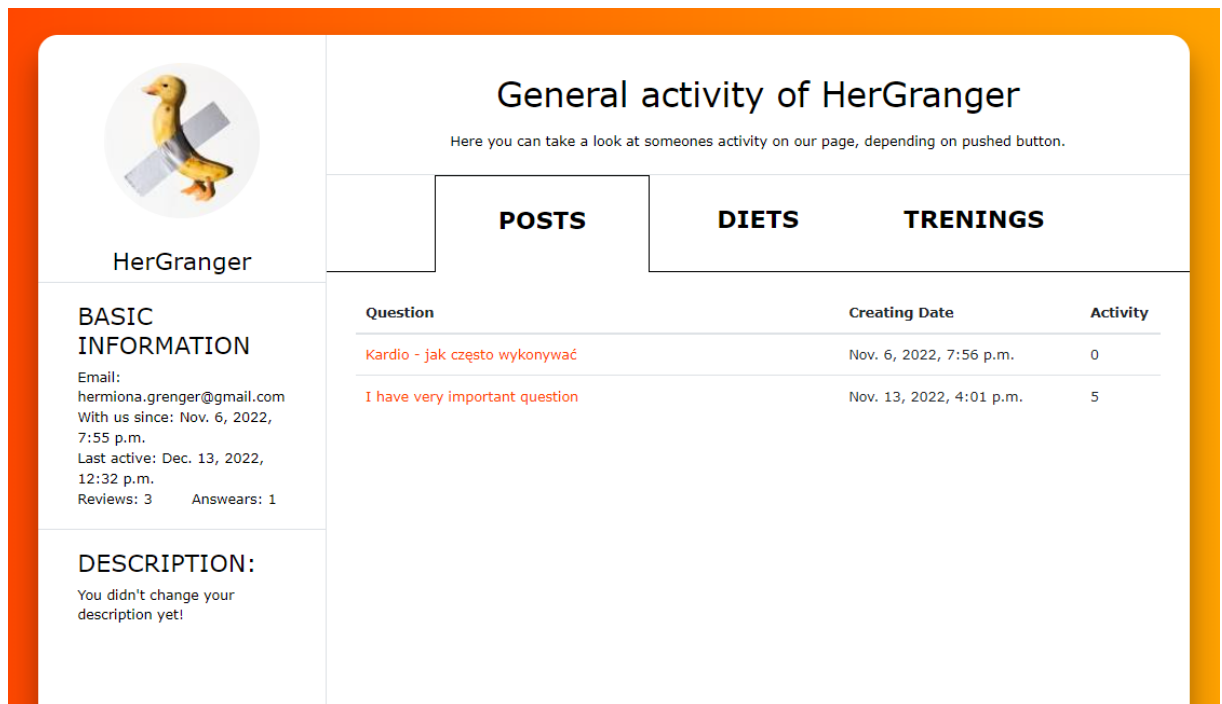
*URL pathing for logging out users.*

Happily django handles everything for us.

```
def logoutUser(request):
    logout(request) # Logout user
    # Send user to login page
    return redirect('login:login')
```

*Logout function provided by Django.*

Thanks to object profile we can create profile page which consist of user profile picture, activity, basic information and description. Profile page consist user specific objects, because from profile page you can delete content provided by you on website but only by yourself on your profile page.



**General activity of HerGranger**  
Here you can take a look at someones activity on our page, depending on pushed button.

	POSTS	DIETS	TRENINGS
<b>Question</b>	<b>Creating Date</b>	<b>Activity</b>	
Kardio - jak często wykonywać	Nov. 6, 2022, 7:56 p.m.	0	
I have very important question	Nov. 13, 2022, 4:01 p.m.	5	

**HerGranger**

**BASIC INFORMATION**  
Email: hermiona.grenger@gmail.com  
With us since: Nov. 6, 2022, 7:55 p.m.  
Last active: Dec. 13, 2022, 12:32 p.m.  
Reviews: 3    Answers: 1

**DESCRIPTION:**  
You didn't change your description yet!

*Profile page of user HerGrange.*

On profile page is everything our model „Profile” have and something more. Django also gives us opportunity to check when user was online last time. Buttons posts, diets and trenings buttons are created thanks to javascript and don't require to refresh the page or to access anything via url.



#### **4.2.5. Problems during applications development**

While creating login, register, profile page and logout system we encountered various problems. Firstly connection forms with views was very complicated and didn't work for various reasons like badly written url file, we didn't save form in proper way, conditions was badly arranged and did not handle all errors. We fixed problem with forms structure with knowledge provided by video from TechTim channel [\[3\]](#) and of course Django documentation [\[4\]](#).

As for user model problems we could fix them from Django documentation[\[5\]](#). Profile object model is based on video posted on YouTube platform[\[6\]](#), but we had to change some things because model from which we inspired was written on version 1.9 but we are operating on version 4.1 so a lot of changes was made for it to work properly.

### **4.3. Coding applications responsible for diets and trainings**

#### **4.3.1. Defining the problem**

We have to create two very similar applications which will allow users to read, review and create content about diets and trainings. Diets and trainings will consist of data like date of creation, author, title, description and reviews. Reviews will consist of description, author of review and date. We also want to add pagination and some sort of animation which allows user to see shortened description.

#### **4.3.2. Analysis of possible solutions**

Firstly we of course need to create models of objects, we can't change much here. But there are two possible ways for allowing user to create content and adding reviews. First one is to create forms by yourself like we did in authentication of user.

Another way of dealing with it is to use provided by Django class of generic View. It simplifies all process and do a lot of heavy lifting by itself but if one doesn't have many experience it can easily go wrong and waste you a lot of time.

As for pagination Django of course provides way to paginate which can easily be used. Other way to do it is pagination provided by jQuery. That method makes it more optimal but require developer to spend many more hours to get acquainted.

Last problem is how to get cut content of description because we cannot display in animation all of it if it is too long. We found only three ways to solve this, first use CSS, second html and last to get that value from python view function.

#### **4.3.3. The proposed solution**

For allowing user to create content on our side we will try using Django generic View because it can easily reduce amount of work we need to do but also we found completed model form which we can borrow some of structure idea and models.

Pagination provided by Django is pretty much the same as the one from jQuery but it requires a lot less effort to use which convinced us to use it.

Lastly we use python view function to get cut content thanks to `diet.context[0:200]` declaration. It cut data of type string to first 200 characters.

#### 4.3.4. Implementation

First we created models for trainings and diets I will show only one because they are very similar:

```
class Diet(models.Model):
    name = models.CharField(max_length=30)
    context = models.TextField(max_length=1000)
    image = models.ImageField(default="diet_pics/default.png", upload_to='diet_pics')
    author = models.ForeignKey(settings.AUTH_USER_MODEL, null=True, on_delete=models.CASCADE)
    date = models.DateTimeField(auto_now=True)
    cut_context = models.TextField(max_length=1000, null=True)
```

*Diet model practically identical to training model.*

As in post every diet have attributes of diet name, content, author and date of creating. But in diets and training models also stores image which will be displayed in tiles in diet/training application. Diet and training need reviews so we needed to create those as well.

```
class Review(models.Model):
    context = models.TextField(max_length=1000)
    author = models.ForeignKey(Profile, on_delete=models.CASCADE, null=True)
    date = models.DateTimeField(auto_now=True)
    object = models.ForeignKey(Diet, on_delete=models.CASCADE)
    rating = models.FloatField(default=0)
    number = models.IntegerField(null=True)
    def __str__(self):
        return str(self.object) + ' - ' + str(self.author)
```

*Review model.*

To make every review have star rating we needed to create a new field rating which stores the value of stars user gave with the review. For example if he gave diet a rating of 4.5, rating will store value 4.5.

Data is transferred to application main page and displayed as many tiles as there is diets. We made it so it displays title of diet and image, author of diet and his profile image, date of cration. To make application main look dynamic we used Bootstrap. When you use computer diets will line three for each row, if you are using phone, you will only get one diet in each row.

We also add some nice css effect which allows us to make beatiful animation. When we hover our mouse on top of picture we will get shotened version of content of diet.



*Diet before and after hovering with mouse.*

We add pagination, so on each page can be only 9 diets. There are three types of pagination navigation bar, when you are on first page, last and between them but paginator will always show on what page are you currently on.



*Three variant of pagination bar. When you are on first, last and any other page.*

One arrow sign will send you to next page, double arrow sign will send you to last page, and of course specific page number will send you to that exact page. Paginator work with get request.

For detailed view of page we send correct data to display, decorated it correctly with help of Bootstrap, css and html.



## Kopenhadzka


 **HerGranger**  
Nov. 6, 2022, 9:52 p.m.

### Description:

<p>Lorem ipsum dolor sit amet. Eum reiciendis consequatur est alias sint 33 ducimus alias cum consequatur magnam eos mollitia magni. Sed earum dolore a autem omnis eos galisum pariatur et ratione consequatur et sunt galisum At modi aliquam. </p><p>Ut quis iure sit fugit nesciunt


*Details of diet.*

Under details of diet are current reviews of diet sorted by date. Reviews consist of author of review, date of review, content and rating they gave.

 **admin** ★★★★★

Bardzo pyszna i przemyślana potrawa!!

Dec. 13, 2022, 12:24 a.m.

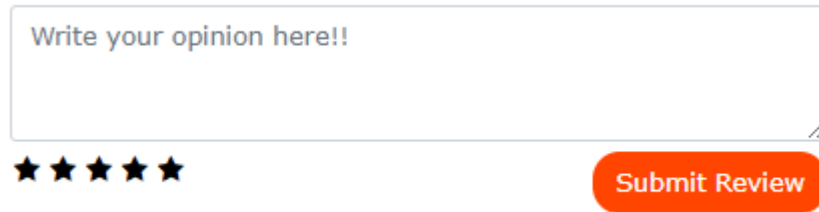
 **HerGranger** ★★★★★

Zawiera zdecydowanie za dużo kalorii.

Dec. 20, 2022, 12:49 a.m.

*Examples of possible reviews.*

Under list of reviews there is a form, it allows logged users to add their own review of diet, we did it so user only need to write anything and choose how many stars he wants to rate it. Form works in a way so that if users write review again, it will update previous review and not add another one.



*Form at the bottom to write your own review.*

#### **4.3.5. Problems during applications development**

The biggest problem in this applications were handling sending diet data, reviews data and handling review form at the same time because both of these action take other method. Entering the page and seeing its content is managed by get method, but submitting and saving form is managed by post method. It was very difficult to combine them at start if you have never did it before but we managed with help of Django documentation[\[4\]](#).

Paginations were also very problematic, we solved issue by cutting data send to applications into packages and handling pagination bar in three times of situations like it is showed above. Model of pagination is based on model showed in this video[\[7\]](#) and of course Django documentation[\[8\]](#).

The hardest part of this application to deal with was rating system which model is based on this video[\[9\]](#) but a lot of changes had to be made to make it work. We managed to work it by ourself. We had to also combine it with review system we borrowed from this source[\[10\]](#).

## **4.4. Programming fully interactive forum application**

### **4.4.1. Defining the problem**

We need to create fully user interactive users forum, where users can ask question, answer them and rate how helpful was answer. There will be displayed list of questions, user will be able to give answer to question after getting to details. Form for answering will be at bottom. Next to every answer we will put like button which will define if answer is helpful or not. Every user can like only once and can cancel that at any time.

### **4.4.2. Analysis of possible solutions**

For creating questions we can use the same methods of implementing form as in diet and trainings application. That means creating forms ourself or using forms provided by generic View library.

Pagination system can be chosen the same way as diets from jQuery or Django.

Comment system is more complex then review, because every comment is connected relation many to many to likes, likes are connected to users, post is connected one to many to comments and profile and author are connected one to one. As we can see there are a lot of relationships and therefore are many ways to implement this.

Most problematic is like system because we need to refresh number of likes and button appearance every time user clicks it. In order to do that we need to refresh only part of website and not all of it. In order to do that we need to use ajax API. There are many frameworks which are providing it and we need to decide on one.

### **4.4.3. The proposed solution**

This time we didn't use forms provided by Django to challenge ourselves and made our own step by step to understand the whole process. Pagination system will whole be taken from diet application because it will probably work correctly and will save time much time.

We decided one of possible structures of data base which will be showed in point 4.4.4.

After researching if there is easy to implement ajax framework all proved to be pretty much the same so we chose one from jQuery which we are already using.

#### 4.4.4. Implementation

So first for forum we created model with Post with attributes of:

```
class Post(models.Model):
    name = models.CharField(max_length=100)
    context = models.TextField(max_length=1000)
    author = models.ForeignKey(Profile, on_delete=models.CASCADE, null=True)
    start_date = models.DateTimeField(auto_now_add=True)
```

*Post model.*

Name of post and context are written by user while creating a post. Author is connected by one to many relationship with post, so author can have a lot of posts but post can only have one author. Start date is time type of data and stores data of creating a post. Author and start date are filled automatically and user can change it by himself. Post also has comment and likes but we needed to create them as another objects because comment also need to have attributes like start date, author, context and number of likes.

```
class Comment(models.Model):
    context = models.TextField(max_length=
    author = models.ForeignKey(Profile, o
    start_date = models.DateTimeField(auto_no
    post = models.ForeignKey(Post, on_d
    likes = models.ManyToManyField(User,
```

*Comment model.*



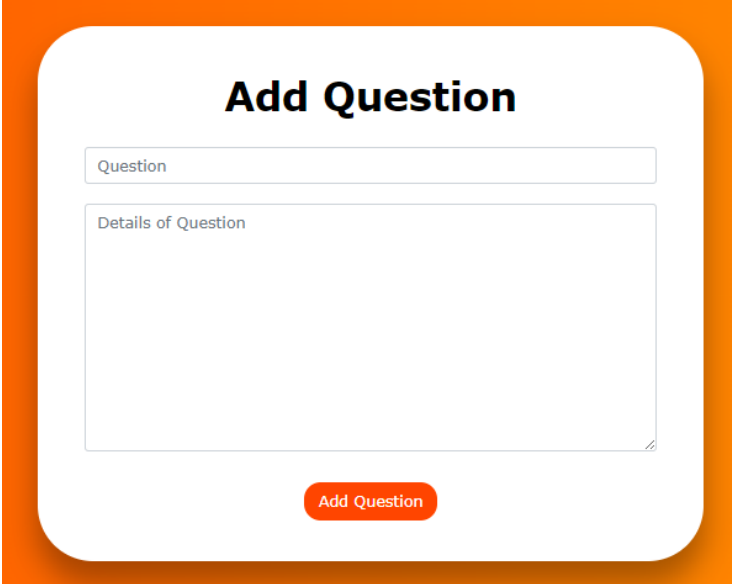
Likes are connected to comments by many to many relationship, comment can have many likes and people can like many comments. We created likes so that every user can like every comment only one time.

```
LIKE_CHOICES = {
    ('Like', 'Like'),
    ('Unlike', 'Unlike'),
}

class Like(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    comment = models.ForeignKey(Comment, on_delete=models.CASCADE)
    value = models.CharField(choices=LIKE_CHOICES, default='Like', max_length=10)
```

*Like model.*

Forum application allows logged in users to ask questions on forum. Question consist of title and detail explanation which are written directly by user, and author and date which are automatically assigned to question by backend commands. Below we show how create question form looks like:

The image shows a web form titled "Add Question" centered on a white background with rounded corners, set against an orange gradient backdrop. The form contains two input fields: a single-line text box labeled "Question" and a larger multi-line text area labeled "Details of Question". At the bottom center of the form is a red button with the text "Add Question" in white.

*Form for adding questions.*

It is very simple but to make it look better we added some css effects which change form while hovered or active. After submitting it you get redirected to forum application main page.

Question

Question

Question

Add Question

Add Question

CSS styling we add to make form look better.

FORUM		
This is forum section where you can ask questions and help other people by answering their question. If you wanna ask <a href="#">click here</a> .		
Question	Activity	Stats
 <b>Od ilu kg na ciężarkach zaczynać</b> <p>Lorem ipsum dolor sit amet. Et tenetur tenetur aut amet illo est itaque voluptatem eos corru...	1	Posted by <b>ronwes</b> Nov. 6, 2022, 7:57 p.m.
 <b>Jak radzić sobie z urazami</b> <p>Lorem ipsum dolor sit amet. Et tenetur tenetur aut amet illo est itaque voluptatem eos corru...	0	Posted by <b>harrypot</b> Nov. 6, 2022, 7:57 p.m.
 <b>Kardio - jak często wykonywać</b> <p>Lorem ipsum dolor sit amet. Et tenetur tenetur aut amet illo est itaque voluptatem eos corru...	0	Posted by <b>HerGranger</b> Nov. 6, 2022, 7:56 p.m.

List of questions listed by date of creation.

Forum application consist of list questions and every question consist of author and his profile picture, cut content, title, activity which is number of replay and date. In this application we also some hover and active css animations, and pagginations, this time we set maximum value of post per page at six.

<<

<

1

2

3

>

>>

Paggination bar – the same as in diet app.



When user open detail view of question we of course display values stored in post model.



**Kardio - jak często wykonywać**  
**HerGranger** - Nov. 6, 2022, 7:56 p.m.  
 <p>Lorem ipsum dolor sit amet. Et tenetur tenetur aut amet illo est itaque voluptat  
 accusamus minima et galisum placeat a inventore delectus a dolorem nihil et omnis  
 qui dolorem omnis in iure ipsa. Ab nobis delectus ut illum galisum est sint consectet  
 galisum hic dicta ipsum ut accusantium obcaecati eos eaque maxime et cumque mai  
 libere voluptatibus eos voluptatibus colute. </p>

Detail view of every question.

Then we get all reviews that are connected to this post via relationship one to many.

Answers		Stats
	<b>admin</b> - Dec. 20, 2022, 1:34 a.m. Raczej nie mam dobrej odpowiedzi na twoje pytanie.	<div>Unlike</div> <div>2 - likes</div>
	<b>ronwes</b> - Dec. 20, 2022, 1:35 a.m. Musisz dużo więcej jeść!!	<div>Like</div> <div>0 - likes</div>

*Example answer for questions.*

And last on our detail application view we have another form which allows us to add our own answers to question. It's decorated in exactly the same manner as create question form and review form without rating.

Write your answer here!!

Add Question

*Form for answering questions at the bottom.*

#### **4.4.5. Problems during applications development**

Comments are really simple to make, but we encountered problem which took us better part of the week. We encountered various problems with like buttons, first they needed to be connected to user in way that user can like every post but only one time and second likes need to be connected to specific post. It took us some time to come up with this model as solution.

Almost immediately after that we found another problem because every time we liked or disliked an answer page would refresh, because like button is built like a form and when form is submitted it redirects you somewhere else. In order to fix it we had to use ajax request. Ajax request allowed us to refresh only specific part of website in this case like button [\[11\]](#).

This application was probably the hardest one, because on top of sending data of question to display, all connected with this questions answers, two types of forms which need to work entirely separate, css decorations and even handling ajax request and some javascript. We encountered many problems with this part of project and spend many hours fixing every single one of them.

## **4.5. Creating basic BMI calculator app.**

### **4.5.1. Defining the problem**

The goal is to create very simple BMI calculator. It will ask user only about his weight in kilograms and height in meters. After writing necessary data and submitting it you will get appropriate result. It will say how much is your BMI and what mass compartment are you applying to.

### **4.5.2. Analysis of possible solutions**

Form we can use with any of previous solutions it won't be much problem.

Real problem is now getting that data and instead of putting it in database we grab it use it to calculate BMI and send appropriate message. There are couple ways of dealing with this. First we can use JavaScript and just react accordingly to data we are getting. Second solution is to get data via html id in views.py in python and send some data back.

### **4.5.3. The proposed solution**

We will use form made by ourselves but not in forms.py file like previously but entirely in html file. It will help us much because we chose to send written data to backend views.py, calculate it there and send back html response with proper data. That type of creating form is useful in that type of situation.

#### 4.5.4. Implementation

First we had made basic form and template to be able to get data like height and weight from the user.

## BMI Calculator

*BMI Calculator form before submitting.*

After we received data from user we calculate the BMI and display it with the message. Message is changing according to how much is his BMI: Severe Thinness, Moderate Thinness, Mild Thinness, Normal, Overweight, Overweight type I, Overweight type II and Overweight type III. All calculations, text confirmation, data sending and form validation is handled by the same function. After sending answer back by backend it gets displayed under form.

## BMI Calculator

**Your BMI = 26.23 kg/m<sup>2</sup> - Overweight**

*BMI Calculator form after submitting.*

In our backend we only need to make simple if function which handles calculation.

```
def MainView(request):
    context = {}
    if request.method == "POST":
        weight = float(request.POST.get("weight-metric"))
        height = float(request.POST.get("height-metric"))

        bmi = ((weight)/(height**2))

        if bmi < 16:
            state = "Severe Thinness"
        elif bmi > 16 and bmi < 17:
            state = "Moderate Thinness"
        elif bmi > 17 and bmi < 18.5:
            state = "Mild Thinness"
        elif bmi > 18.5 and bmi < 25:
            state = "Normal"
        elif bmi > 25 and bmi < 30:
            state = "Overweight"
        elif bmi > 30 and bmi < 35:
            state = "Overweight type I"
        elif bmi > 35 and bmi < 40:
            state = "Overweight type II"
        elif bmi > 40:
            state = "Overweight type III"

        context["bmi"] = round(bmi, 2)
        context["state"] = state

    return render(request, "bmicalc/main.html", context)
```

#### 4.5.5. Problems during applications development

We didn't encounter many major problems. Mainly thanks to forum, diet and trainings applications we were very skillful with handling data sending, connecting forms with views[\[12\]](#), handling requests and styling it. We also have a lot of completely ready frontend code which we often re-use.

## **4.6. Designing home page**

### **4.6.1. Defining the problem**

One if not the most important part of every website is home page. How does home page look, if is it nicely decorated and full of animated effects is really important, because it's first thing new user see on our website. So we had to put a lot of effort into making one.

### **4.6.2. Analysis of possible solutions**

There are many framework which can be used for styling our website. First one is of course Bootstrap, React, Vue.js, Angular, Svelte, jQuery, Ember, Backbone, Semantic UI. But there are way too many too master and it would take too much of our time so we have to decide on only some of them.

Except which framework and languages we will use we need to decide what we need to put there like widgets and if widgets what would they look like, some comments, reviews, profiles, what kind of navigation bar etc.

### **4.6.3. The proposed solution**

Firstly we decided we would like bar fixed always on top of our page if navbar is at top it would be half invisible but if you scroll down it would turn fully visible. Of course there must be buttons to each application and some kind of hover animation.

Next we decided to limit amount of used for us framework and use besides basic languages Bootstrap and jQuery. Bootstrap because not only is it really easy to use it also makes website easily responsible and helps to make mobile versions of pages really fast. And finally jQuery for handling any more advanced animations that CSS and JavaScript can't handle.



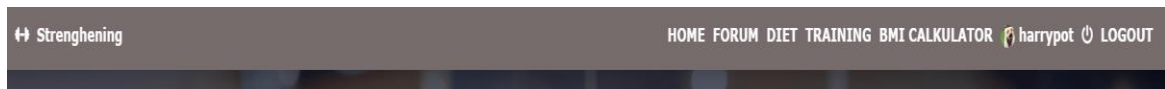
### 4.6.4. Implementation

First of all we made nice looking navigation bar transparent whenever it is at top with JavaScript.



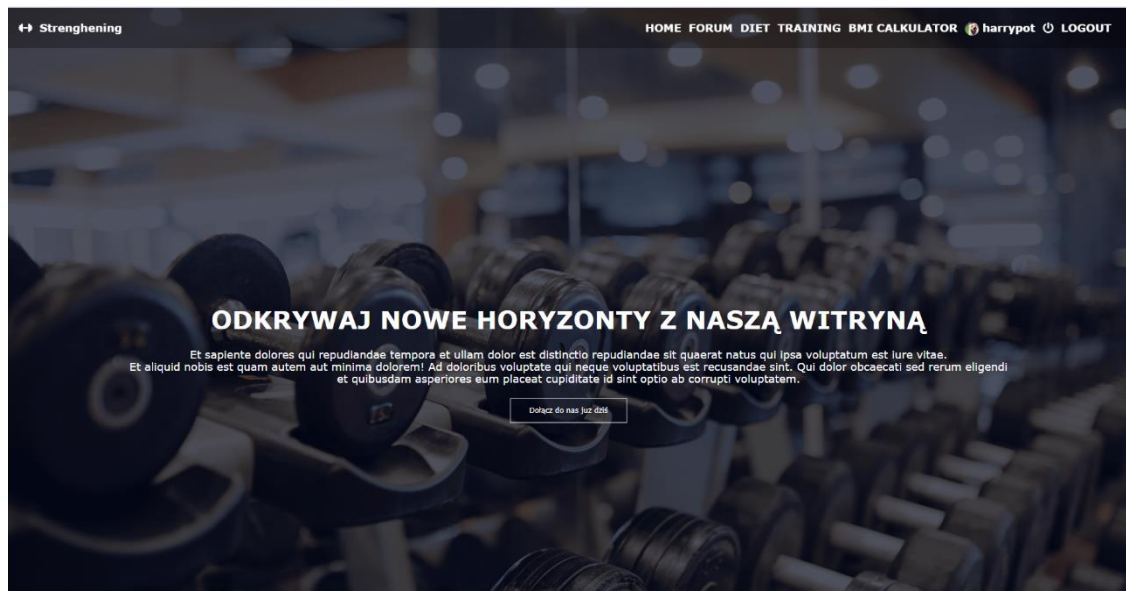
*Navigation bar on top.*

But when you scroll just a little to bottom it starts to change color to grey.



*Navigation bar on any other place.*

We added at top of home page big image to catch attention of people with some slogan on it and button redirecting to register page. We also added CSS position absolute to make picture always look the same. And of course CSS hover animation on every button.

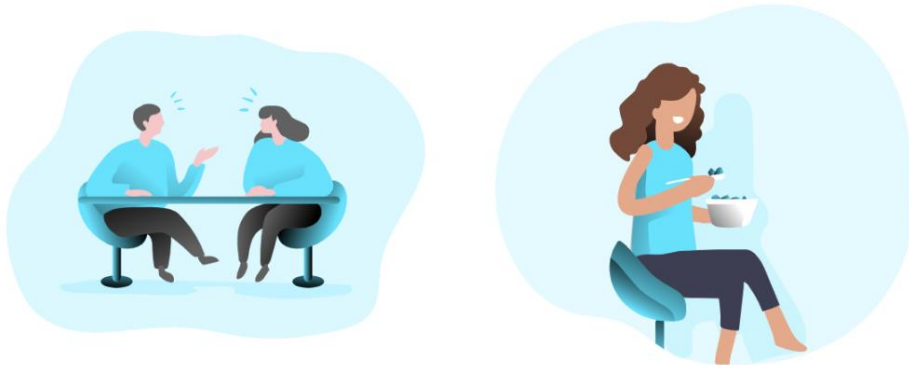


*At very first big picture is welcoming us.*

Right under picture we have four widgets. Every one of four of them is picture related to application which widgets leads to for example here we have forum and diets widgets.

## Nasza strona oferuje wiele możliwości

Et sapiente dolores qui repudiandae tempora et ullam dolor est distinctio repudiandae sit quaerat natus qui ipsa voluptatum est iure vitae.

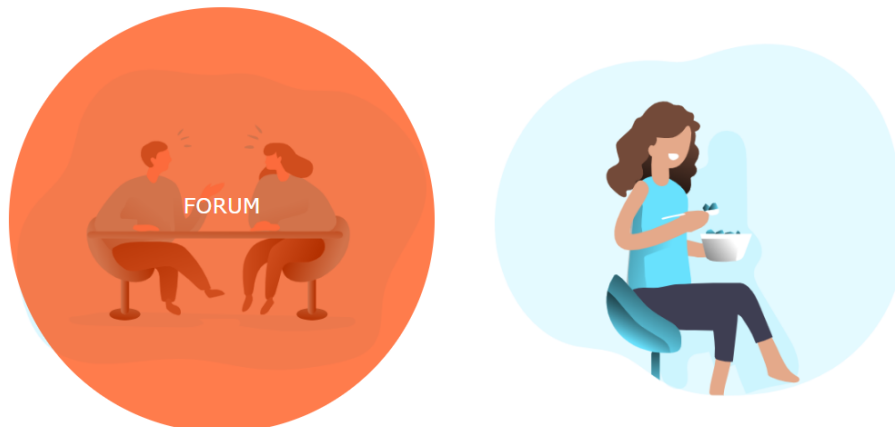


*Widgets when aren't hovered.*

We made new kind of animation using CSS which not only change background but display some text which appear slowly from the bottom.

## Nasza strona oferuje wiele możliwości

Et sapiente dolores qui repudiandae tempora et ullam dolor est distinctio repudiandae sit quaerat natus qui ipsa voluptatum est iure vitae.



*Widgets when hovered.*

Next we created two part section which contains one not changing article witch can be written with some interesting explanation, and second part with the most active topic on one of three applications. Getting object with the most activity was very hard and it required from us using two-staged filter, for loop and even counter. We got full data that objects and displayed it in nice manners.

## Troche do powiedzenia o naszej stronie!

Et sapiente dolores qui repudiandae tempora et ullam dolor est distinctio repudiandae sit quaerat natus qui ipsa voluptatum est iure vitae. Et aliquid nobis est quam autem aut minima dolorem! Ad doloribus voluptate qui neque voluptatibus est recusandae sint. Qui dolor obcaecati sed rerum eligendi et quibusdam asperiores eum placeat cupiditate id sint optio ab corrupti voluptatem.

### POST

#### I have very important question

Something very horriffaing happenend

Posted by **HerGranger** on Nov. 13, 2022, 4:01 p.m.

### DIET

#### Kopenhadzka

<p>Lorem ipsum dolor sit amet. Eum reiciendis conse

Posted by **HerGranger** on Nov. 6, 2022, 9:52 p.m.

### TRAIN

#### Kaszel

isdojfhiuwehfuiwe

Posted by **ronwes** on Nov. 7, 2022, 12:24 a.m.

*Widgets with some kind of article and three most popular topics on page.*

We also show the newest reviews at the moment, these are also contantly changing but are much easier to observe. First review comes from diet aplication and second comes from training applications.

## OPINIE

Pare kometarzy aby pokazać jak ludzie wyrażają sie o treściach na naszej stronie



**HerGranger** Dec. 20, 2022, 12:49 a.m.

★★★★☆

Zawiera zdecydowanie za dużo kalorii.



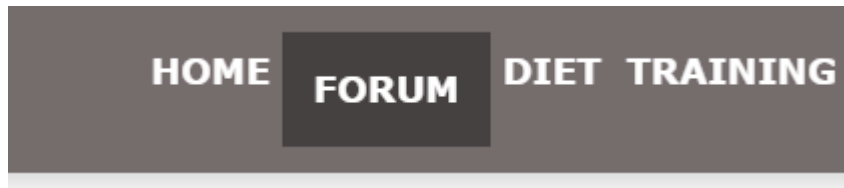
**admin** Dec. 13, 2022, 4:30 p.m.

★★★★★

sadsada

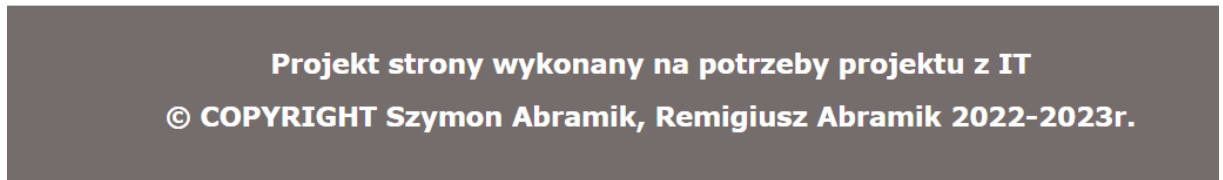
*Two newest reviews.*

We decorated our navigation bar buttons so if we hover over any link it changes background and overall size.



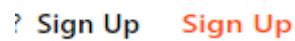
*Navbar buttons when hovered.*

Every normal website have some kind of footer, so we made one which can contain information about copyrights, rules, financials partners or inform about contact info.



*Footer with some kind of information.*

We added some styling int the rest of buttons,



*Example of styling buttons.*

and created mobile version of our home page:

## **Nasza strona oferuje wiele możliwości**

Et sapiente dolores qui repudiandae tempora et ullam dolor est distinctio repudiandae sit quaerat natus qui ipsa voluptatum est iure vitae.



*Mobile version of widget segment.*

## Troche do powiedzenia o naszej stronie!

Et sapiente dolores qui repudiandae tempora et ullam dolor est distinctio repudiandae sit quaerat natus qui ipsa voluptatum est iure vitae. Et aliquid nobis est quam autem aut minima dolorem! Ad doloribus voluptate qui neque voluptatibus est recusandae sint. Qui dolor obcaecati sed rerum eligendi et quibusdam asperiores eum placeat cupiditate id sint optio ab corrupti voluptatem.

### POST

#### I have very important question

Something very horriffaing happenend

Posted by **HerGranger** on Nov. 13, 2022, 4:01 p.m.

### DIET

#### Kopenhadzka

<p>Lorem ipsum dolor sit amet. Eum reiciendis consequat...

Posted by **HerGranger** on Nov. 6, 2022, 9:52 p.m.

### TRAIN

#### Kaszel

isdofjhiuwehfuiwe

Posted by **ronwes** on Nov. 7, 2022, 12:24 a.m.

## OPINIE

Pare komentarzy aby pokazać jak ludzie wyrażają się o treściach na naszej stronie



**HerGranger** Dec. 20, 2022, 12:49 a.m.

★★★★☆

Zawiera zdecydowanie za dużo kalorii.



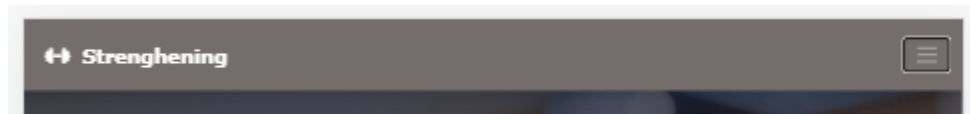
**admin** Dec. 13, 2022, 4:30 p.m.

★★★★★

sadsada

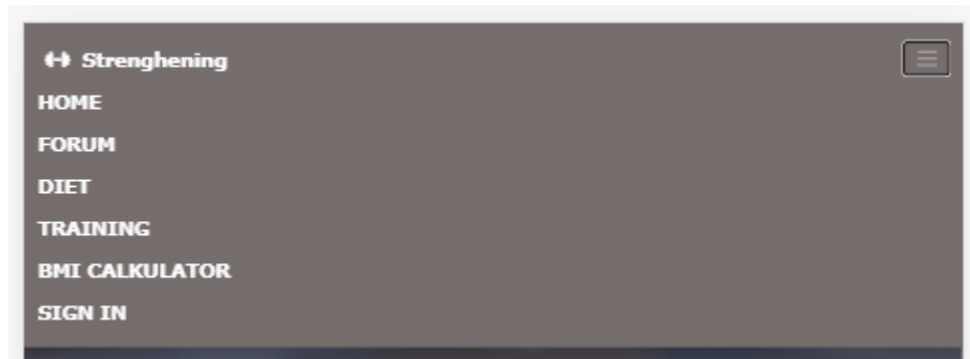
*Mobile version of other segments.*

At last we implemented collapsed navbar for mobile version. Before opening:



*Navigation bar normal.*

After opening:



*Navigation bar opened.*

#### **4.6.5. Problems during applications development**

We of course encountered some problems with too wide or short padding or margin, badly designed grid and flex system, couldn't understand the difference between background and background-colour, scaling of multiple objects, fixing navbar at top, animating navbar[\[13\]](#), sizing height of screen or sizing of photos. We fixed our problems mostly using bootstrap, but we fixed some with just CSS and @media command[\[14\]](#).

All this front end decorating took a lot of time because it is about constatly writing somethin and looking if it look good and symetric. If not try again. But bootstrap made making mobile versions of pages and navigation bar a lot easier. Bootstrap provides you with changing the size of column depending on screen size and have already implemented collapsing navbar. So without bootstrap i don't think it would be possible to make all this mobile versions within our time limit.



## **5. Summary.**

After completing the work on the project, it definitely differs from the initial intentions and goals we gave ourselves to achieve. This is mainly due to a definite increase in our knowledge of technologies and solutions used in web development. At the beginning of the project we did not foresee how much effort, testing, knowledge, planning and time it takes to create a seemingly simple website.

At many stages of our project we encountered many problems, such as transferring data, receiving and sending requests, designing more complex structures, handling many forms at once, proper styling of pages, creating mobile versions, and securing them from undesired users. Fortunately, we were able to solve all the problems we encountered.

The search for solutions to our problems was definitely helped by the Stack Overflow forum for developers, where many people encountered virtually identical problems to us. Stack Overflow, along with the documentation of the frameworks, helped us locate and then fix each problem. However, if we had not been well acquainted with all the elements we were using and had not carefully planned the structure of the entire site and divided it into appropriate sections, the project could have taken much longer.

The project helped us not only to acquire knowledge of web technologies such as the languages, libraries and frameworks we used, but also to develop the necessary skills such as working independently, searching for appropriate solutions, getting familiar with them and finally implementing them. Other skills such as patience and conscientiousness to solve errors and working in a team along with the very helpful web hosting service GitHub were also required.

## 6. Sources.

### 6.1. Used libraries and frameworks.

Bootstrap: <https://getbootstrap.com>

jQuery: <https://jquery.com>

Python: <https://www.python.org>

Django: <https://www.djangoproject.com>

Font Awesome: <https://fontawesome.com>

### 6.2. Other sources.

- [1]. Django documentation URL usage - Django Software Foundation - last visited: 8.01.2023  
<https://docs.djangoproject.com/en/4.1/topics/http/urls/>
- [2]. Static files management in Django documentation - Django Software Foundation - last visited: 13.10.2022  
<https://docs.djangoproject.com/en/4.1/howto/static-files/>
- [3]. Source to understand forms video - @TechWithTim - last visited: 15.10.2022  
<https://www.youtube.com/watch?v=vM9mcWr1RMg&t=1127s>
- [4]. Django documentation forms usage - Django Software Foundation - last visited: 2.01.2023  
<https://docs.djangoproject.com/en/4.1/topics/forms/>
- [5]. Basic user model provided by Django - Django Software Foundation - last visited: 15.12.2022  
<https://docs.djangoproject.com/en/4.1/ref/contrib/auth/>
- [6]. Advanced user model - @thenewboston - last visited: 29.10.2022  
<https://www.youtube.com/watch?v=eMGtdtNR4es&list=PL6gx4Cwl9DGBImzzFcLgDhKTTfNLFx1IK&index=37>
- [7]. Pagination example model we based our website - @Codemycom - last visited: 27.12.2022  
[https://www.youtube.com/watch?v=wY\\_BNsxCei4](https://www.youtube.com/watch?v=wY_BNsxCei4)
- [8]. Handling pagination in Django - Django Software Foundation - last visited: 27.10.2022  
<https://docs.djangoproject.com/en/4.1/topics/pagination/>

- [9]. Example of star rating system - @Pyplane - last visited: 2.1.2023  
[https://www.youtube.com/watch?v=iz1GB\\_q5txM&t=2055s](https://www.youtube.com/watch?v=iz1GB_q5txM&t=2055s)
- [10]. Example of review system - @rathankumar - last visited: 2.1.2023  
<https://www.youtube.com/watch?v=eIN1nZCt7Ww>
- [11]. Django and Ajax button model - @Pyplane - last visited: 10.12.2022  
<https://www.youtube.com/watch?v=kRrPtIjnxqs&t=1401s>
- [12]. Getting data from form by id – all answers authors - last visited: 15.12.2022  
<https://stackoverflow.com/questions/4706255/how-to-get-value-from-form-field-in-django-framework>
- [13]. Navigation bar animation - @WebZoneCode - last visited: 3.1.2023  
<https://www.youtube.com/watch?v=kmmxxfyufLE&t=595s>
- [14]. Home page styling - @EasyTutorialsVideo - last visited: 3.1.2023  
<https://www.youtube.com/watch?v=oYRda7UtuhA&t=948s>