ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

на правах рукопису

Писклинець Володимир Романович

УДК 004.4

Розробка онлайн-платформи для спортзалу із можливістю створення особистого кабінету користувача

Спеціальність 121 – «Інженерія програмного забезпечення»

Кваліфікаційна робота на здобуття кваліфікації бакалавр

Нормоконтроль

_____Стисло О.В.

(підпис, дата, розшифрування підпису)

Студент

_____Писклинець В.Р.

(підпис, дата, розшифрування підпису)

Допускається до захисту

Завідувач кафедри

_____к.т.н., доц. Ващишак С.П.

(підпис, дата, розшифрування підпису)

Керівник роботи

к.т.н., проф. каф. ІТ

_____ Пашкевич О.П.

(підпис, дата, розшифрування підпису)

Івано-Франківськ – 2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

Освітній ступінь: «бакалавр»

Спеціальність: 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

«____» _____ 2024 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Писклинець Володимир Романович

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи:
Розробка онлайн-платформи для спортзалу із можливістю створення особистого кабінету користувача

керівник роботи:
Пашкевич Олег Петрович, к.т.н., проф. каф. ІТ

затверджена наказом вищого навчального закладу від « 12 » березня 2024 року № 19/1

2. Термін подання студентом роботи 05.06.2024

3. Вихідні дані роботи: Python, Django, HTML, CSS, Stripe

4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)
   1. Аналіз наявних аналогів
   2. Розробка прототипу сайту
   3. Реалізація функціоналу для особистого кабінету користувача

5. Дата видачі завдання 14.03.2024

КОНСУЛЬТАНТИ РОЗДІЛІВ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Розділ	Консультант (прізвище, ініціали та посада)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Огляд та аналіз існуючих аналогів	27.03.2024	Виконано
2.	Проектування прототипу сайту	01.04.2024	Виконано
3.	Розробка сайту	23.04.2024	Виконано
4.	Оформлення пояснювальної записки	15.05.2024	Виконано
5.	Оформлення графічного матеріалу та підготовка до захисту роботи	20.05.2024	Виконано

Студент _____ Писклинець В.Р.

(підпис) (прізвище та ініціали)

Керівник роботи _____ Пашкевич О.П.

(підпис) (прізвище та ініціали)

Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Сторінка	Опис графічного матеріалу	Сторінка	Опис графічного матеріалу
14	Приклад блочної подачі інформації	43	Інформаційні блоки про послуги спортзалу
15	Карта міста на сайті/Вигляд головного банеру сайту при зменшеному розмірі екрану	44	Розклад занять на сайті
16	Широкий вибір активностей та погана якість фото/Інформація про тренера на сайту SportLife	45	Інформаційний блок
17	Структура сайту Альянс/Приклад статті на сайті	46	Інформаційний блок про тренерів/Відгуки на сайті
18	Таблиця розцінок спортклубу Альянс/Некоректне відображення таблиці	48	Футер сайту/Сторінка про тренажерний зал
24	UML діаграма сайту спортзалу	50	Сторінка групових тренувань/Перелік статей на сайті
25	Навігація по сайту(хедер)	51	Вигляд статті у відкритому вигляді
26	Wireframe головного банера та блоки інформації/Інформаційних блок про тренерів	52	Вигляд статей у адмін панелі

27	Відгуки та футер сайту	53	Вигляд блоку тренера/Вигляд детальної сторінки тренера
28	Перелік статей/Вигляд статті	54	Доступні абонементи для покупки
29	Блоки з тренерами/Детальна інформація про тренера	58	Форма вводу реквізитів для оплати/Дані про оплату із сайту Stripe
30	Вигляд абонементів	59	Форма для авторизації на сайті/Форма для реєстрації на сайті
31	UML діаграма особистого кабінету користувача/Сторінка реєстрації	60	Форма введення пошти для зміни паролю
32	Особистий кабінет користувача/Створення плану тренувань	61	Лист з посиланням на зміну паролю/Форма зміни паролю
33	Сторінка створеного плану тренувань/Структура проєкту	62	Сторінка особистого кабінету користувача/Вигляд сторінки кабінету, де немає жодного плану
34	Gymsite urls/Templates та Static	63	Сторінка з переліком вправ
35	Модель Article/Модель Abonement	66	Відсортований список вправ
36	Модель Purchase/Зв’язки між моделями аплікейшина gymsite	67	Сповіщення про успішне додавання вправи у план
38	Users templates/Users urls/Users Forms.py	68	Відображення доданих вправ у плані тренувань
39	Модель Exercise	75	Відображення створеного плану
40	Модель UserExercisePlan/ Зв’язки між моделями аплікейшина users		
42	Головний банер сайту та хедер		

АНОТАЦІЯ

В результаті роботи був реалізований сайт спортзалу з можливістю створення особистого кабінету користувача, що збільшило зручність надання послуг відвідувачам спортзалу, а також підвищило ефективність ведення бізнесу спортивним залом та збільшило кількість клієнтів.

У першому розділі нашого дослідження був проведений детальний аналіз існуючих аналогів та конкурентів, де було виділено їх переваги та недоліки з погляду функціонального наповнення та дизайну. Це дозволило зробити відповідні висновки при розробці сайту.

У другому розділі була подана детальна інформація про будову сайту. Спроектовано вигляд сайту з урахуванням всіх запланованих функцій з урахуванням результатів дослідження, проведеного в першому розділі.

Третій розділ присвячений реалізації сайту спортзалу за допомогою таких технологій розробки, як: мов програмування – Python, мови розмітки – HTML, CSS та фреймворка Django.

КЛЮЧОВІ СЛОВА: PYTHON, HTML, CSS, DJANGO.

SUMMARY

As a result of the work, a gym website was implemented with the ability to create a personal user account, which increased the convenience of providing services to gym visitors, as well as increased the efficiency of doing business by the gym and increased the number of customers.

In the first section of our research, we conducted a detailed analysis of existing analogues and competitors, where we highlighted their advantages and disadvantages in terms of functionality and design. This allowed us to draw relevant conclusions when developing the website.

The second section provided detailed information about the structure of the site. We designed the site's appearance with all the planned functions based on the results of the research conducted in the first section.

The third section is devoted to the implementation of the gym website using such development technologies as: programming languages - Python, HTML, CSS and the Django framework.

KEYWORDS: PYTHON, HTML, CSS, DJANGO.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

HTML	–	Hyper Text Markup Language
CSS	–	Cascading Style Sheets
MVC	–	Model-View-Controller
ORM	–	Object relational mapping
SQL	–	Structured Query Language
URL	–	Uniform Resource Locator
PCI DSS	–	Payment Card Industry Data Security Standard
WYSIWYG	–	What You See Is What You Get
API	–	Application Programming Interface
HTTP	–	HyperText Transfer Protocol request

ВСТУП

Актуальність теми. Актуальність даної теми полягає в зростаючій потребі у зручних та ефективних способах управління фітнес-процесом для клієнтів спортзалів у сучасному цифровому світі. Завдяки стрімкому розвитку технологій та збільшенню популярності фітнесу серед населення, попит на інноваційні рішення для оптимізації тренувань та контролю за ними постійно зростає.

Онлайн-платформи для спортзалів з особистими кабінетами користувачів відкривають широкі можливості для забезпечення зручності та персоналізації тренувань. Вони дозволяють клієнтам спортзалу легко вести облік своїх тренувань, а також отримувати доступ до розкладу занять та інших сервісів спортзалу безпосередньо через веб-платформу або мобільний додаток.

Ці технологічні рішення сприяють залученню нових клієнтів, збереженню існуючих та підвищенню рівня задоволення від фітнес-процесу, що в свою чергу сприяє підвищенню конкурентоспроможності спортзалу на ринку. Таким чином, розробка та впровадження онлайн-платформ для спортзалів з особистими кабінетами користувачів є важливим напрямком розвитку фітнес-індустрії, що відповідає сучасним потребам та очікуванням клієнтів.

Мета роботи. Розробка онлайн-платформи для спортзалу з можливістю створення особистого кабінету користувача.

Об’єкт роботи. Онлайн-платформа для спортзалу.

Предмет роботи. Розробка онлайн-платформи для спортзалу із можливістю створення особистого кабінету користувача.

Завдання роботи. Відповідно до вибраної теми в роботі
покладені такі задачі як:

1) пошук існуючих на даний момент аналогів для їхнього аналізу;
2) вибір мови програмування та технологій розробки;
3) розроблення зручного дизайну;
4) проведення тестування продукту.

Методи роботи. Для вирішення поставленого завдання були використані мова програмування Python, база даних MySQL, фреймворк Django.

Результати роботи. Результатом роботи буде створена та успішно функціонуюча онлайн-платформа для спортзалу з можливістю створення особистого кабінету користувача. Ця платформа буде надавати клієнтам спортзалу зручний та ефективний спосіб керувати своїми тренуваннями.

Апробація результатів дослідження. Матеріали кваліфікаційної роботи були представлені на І Всеукраїнській науково-практичній інтернет-конференції «ІТ ЕКОСИСТЕМА: Цифровізація бізнес-процесів в умовах війни», яка відбулась 23-24 листопада 2023 року в Університеті Короля Данила.

Структура роботи. Розділи – 3. Загальний обсяг основної частини –64 сторінок. Список використаних джерел – 20.

РОЗДІЛ 1. АНАЛІЗ КОНКУРЕНТІВ

1.1 Індустрія спортивних клубів

Перед тим, як розпочати роботу над реалізацією веб-ресурсу, потрібно дослідити таке явище, як спортзали, що вони пропонують своїм відвідувачам та що в собі містять.

Перш за все, спортклуби – це місця, куди люди приходять за покращенням своєї фізичної форми. Для цього їм потрібні тренажери, за допомогою яких можна було розраховувати на повноцінне тренування всіх частин тіла. Спортклуби пропонують широкий спектр обладнання, яке дозволяє клієнтам займатися спортом за їхніми власними уподобаннями та потребами. Вони діляться на безліч видів, але всіх їх об’єднує одна спільна риса – бути інструментом для різного роду вправ.

Переважна більшість людей, приходячи в спортклуб, стикаються з невідомими до цього тренажерами і як результат – здійснюють ряд грубих помилок при їх експлуатації. Часто буває і загальне нерозуміння мети тренувань та цілковитої неясності у тому як виконується окремо взята та чи інша вправа. Поради тренерів в цьому середовищі є невід'ємною частиною досвіду тренувань. Вони допомагають клієнтам правильно виконувати вправи, розробляють індивідуальні програми тренувань та надають цінні поради щодо харчування та загального здоров'я.

Дедалі частіше, коли здоров'я та фізична активність стають дедалі важливішими, спортзали почали змінювати свою головну мету діяльності. Тепер ці заклади не лише пропонують доступ до тренажерного залу, але й створюють цілісні середовища, де спорт стає не лише засобом тренування, але й засобом саморозвитку та релаксації.

Відкриваються нові студії присвячені різного роду активностям. Наприклад, по сусідству із залами для тренувань, можуть розташовуватися ринги для заняття

боксом. Створюються зали для самого заняття фітнесом, стретчингом, гімнастикою. Відвідувачам пропонуються послуги з оздоровлення та відновлення після тренувань: різного роду масажі, спа-процедури, тощо. Це дозволяє клієнтам забезпечити своє тіло комплексним доглядом, який допоможе швидше відновитися після навантажень.

Подібне розширення структури спортивних залів призводить до кратного збільшення кількості відвідувачів, що заставляє обдумувати можливість оптимізації процесу тренувань. Як вихід, з'являються групові тренування.

Для тих, хто прагне займатися фізичними вправами регулярно та ефективно, вигідним рішенням буде скористатися абонементами. Абонементи, як зручний і ефективний спосіб надання послуг користується надзвичайно високою популярністю і є невід'ємною частиною будь-якого спортзалу. Прийнято ділити їх на часові абонементи(місяць, рік, квартал) та разові(дозволяють відвідувати спортзал протягом обмеженого періоду, наприклад, на одне чи кілька тренувань).

Не можна обійти стороною і вигляд спортзалу. Естетика – важлива складова сприйняття, адже вона формує перше враження про спортклуб та впливає на емоційний стан відвідувачів. Це твердження також справедливе і по відношенню до зовнішнього вигляду веб-сайту. Естетика не лише робить спортзал більш привабливим, але й:

1) збільшує лояльність клієнтів;
2) підвищує цінність бренду;
3) допомагає виокремитися від конкурентів;
4) сприяє позитивним відгукам та рекомендаціям.

Отже, в процесі дослідження спортзалів, ми прийшли до висновку, що основними складовими цього бізнесу є: наявність абонементів, різного роду додаткових активностей, наявність спортивного інвентарю, досвідчених тренерів та привабливий зовнішній вигляд спортзалу та його залів.

1.2 Огляд та аналіз існуючих аналогів і конкурентів

Задля збільшення клієнтів та зручнішого надання послуг, спортзали почали надавати перевагу створенню своїх сторінок в мережі інтернет або всередині різних месенджерів, наприклад в Instagram. Ознайомившись із основними аспектами спортзалів, пропоную дослідити, як їх реалізували на своїх веб-ресурсах конкуренти.

Спершу, хочу звернути увагу на сайт мережі харківських спортзалів - PULSE GYM [1]. Перевагами даного сайту є блочна структурованість контенту, коли кожна незалежна від інших інформація подається в окремому блоці, відокремлено від іншої (рис. 1.1).

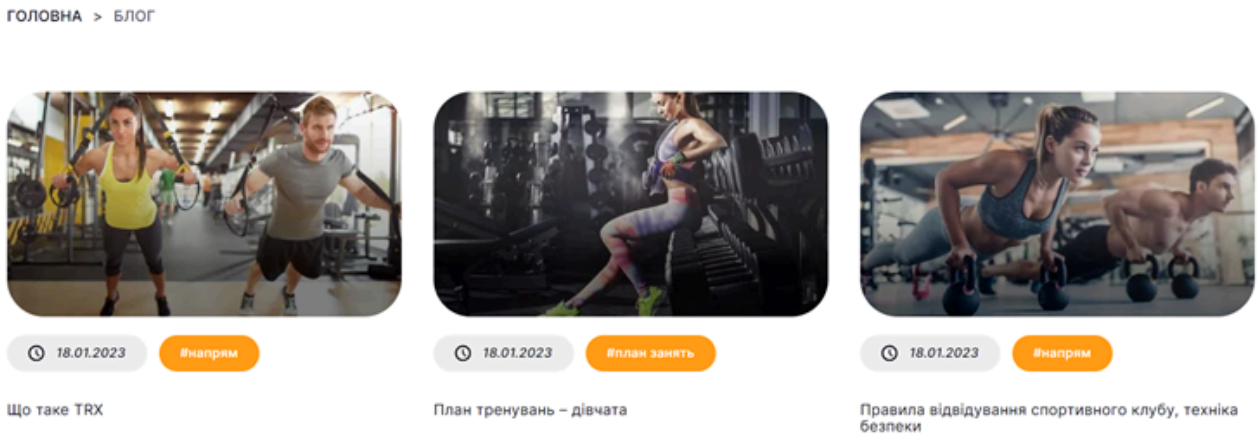


Рисунок 1.1 – Приклад блочної подачі інформації

Також, сайт побудований за всіма принципами колористики. Дотримана гармонія тонів. Підтримується адаптивний дизайн. Розмір елементів сайту легко змінюється при зменшенні розміру екрану (рис. 1.2). Якість наповнення при такому процесі не страждає.

Сайт гарно структурований. Інформація подається логічно. Також виділю доступність сайту іншою мовою.

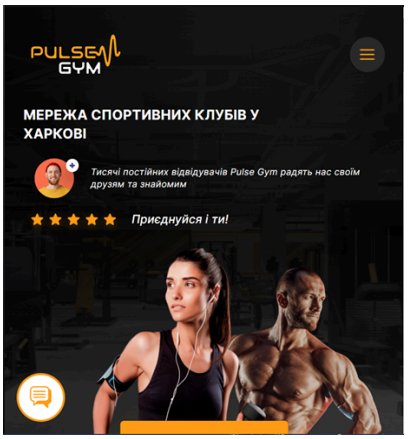


Рисунок 1.2 – Вигляд головного банеру сайту при зменшеному розмірі екрану

Але сайт має й свої недосконалості. Основним є рішення зробити карту Харкова по всій довжині екрану, незалежно від його розмірів (рис. 1.3). Цю карту можна збільшувати і зменшувати скільки завгодно. На ній показані всі спортзали даної мережі в місті. Але при скролінгу веб-ресурсу через цю карту, скролитися починає сама карта, а не сайт.

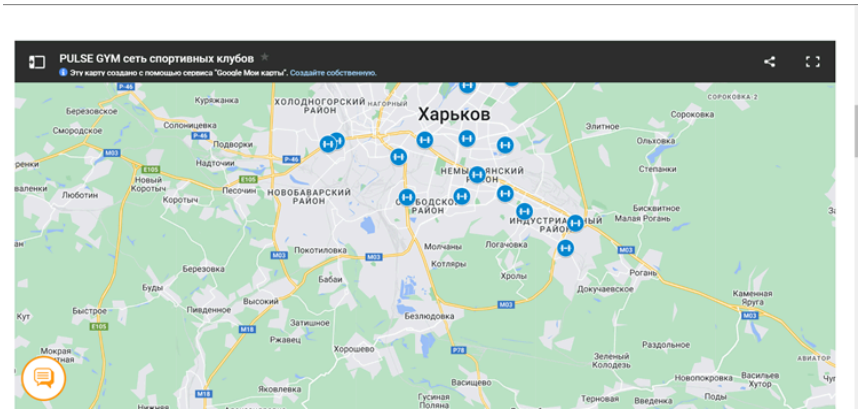


Рисунок 1.3 – Карта міста на сайті

Також, відгуки можна чомусь гортати тільки вперед, а, наприклад, від останнього до першого не можна. Кнопка пролистування назад на першому відгуку просто не активна.

Одним із найвідоміших конкурентів в Україні є мережа спортзалів SportLife [2]. Їхні спортзали розташовані у більшості великих міст країни. Перевагами

сайту є точна і актуальна інформація про кожен спортзал, незалежно від його місця розташування. Коректні адресні дані, кольористика та структурованість веб-ресурсу, широкий вибір активностей (рис. 1.4). Але одразу кидаються в очі картинки поганої якості при перегляді активностей на відповідній сторінці, що сильно портить перше враження від сайту і потенційно може змусити користувача залишити його.

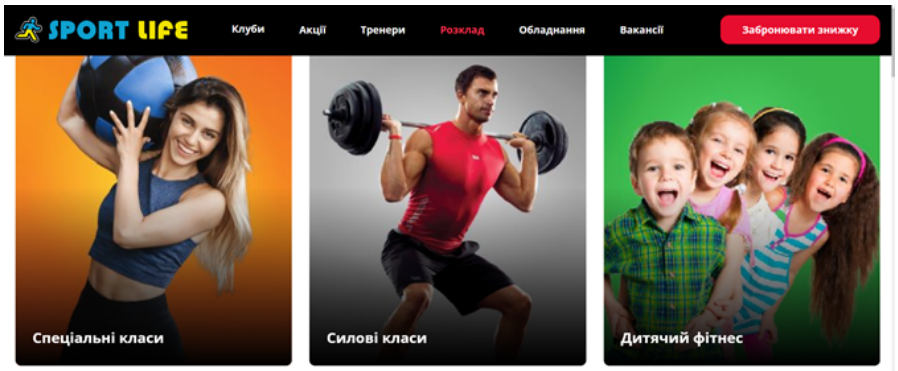


Рисунок 1.4 – Широкий вибір активностей та погана якість фото

Вивчаючи сайт, можу виділити зручну навігацію по тренерам і коротку, недостатню інформацію про них та метод подання цієї інформації (рис. 1.5).

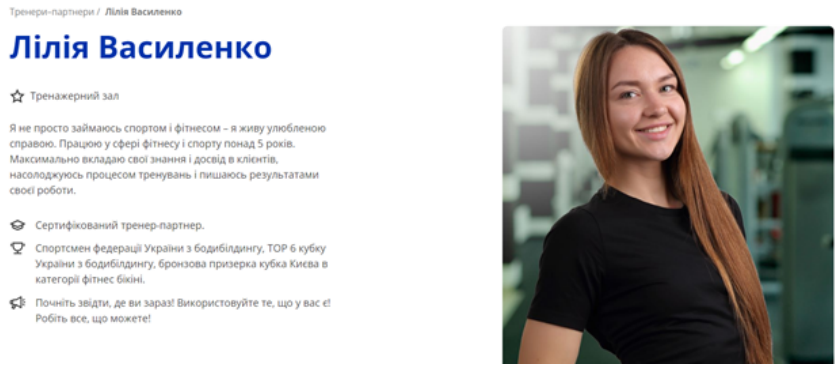


Рисунок 1.5 – Інформація про тренера на сайту SportLife

Під кінець, хотів би звернути увагу на сайт одного з популярних залів міста Івано-Франківськ, а саме веб-ресурсу спортивного клубу «Альянс» [3]. Одразу варто виділити жахливий дизайн та структуру веб-сайту (рис. 1.6). При перегляді в першу чергу кидаються в очі банери партнерів та оголошення самого залу, а вже

потім можна помітити новини клубу, які несуть в собі мінімум інформації (рис. 1.7). Також, у статтях відсутнє фокусування на основному вмісті. Текст малий, відсутня робота над оформленням сторінки. Всі елементи розташовані, як небудь. Зображення мале і розташоване не на своєму місці.

Окремо хочу звернути увагу на відсутність активних посилань у тексті, хоча вказані посилання на інстаграм сторінки.

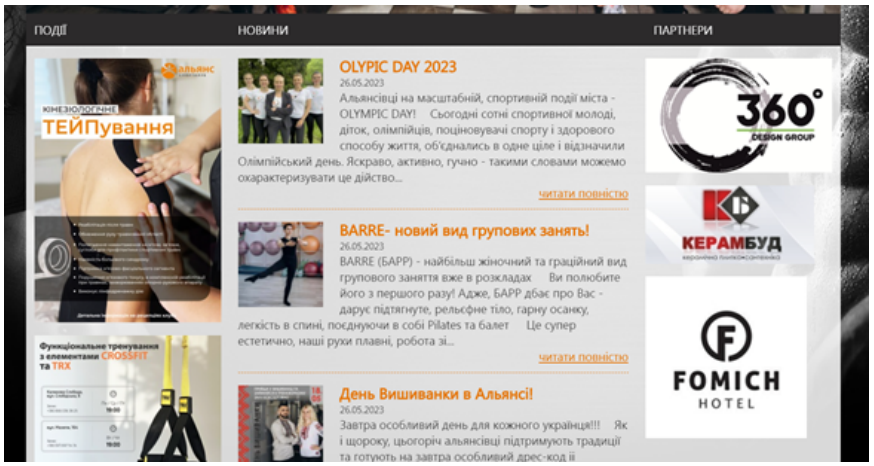


Рисунок 1.6 – Структура сайту Альянс

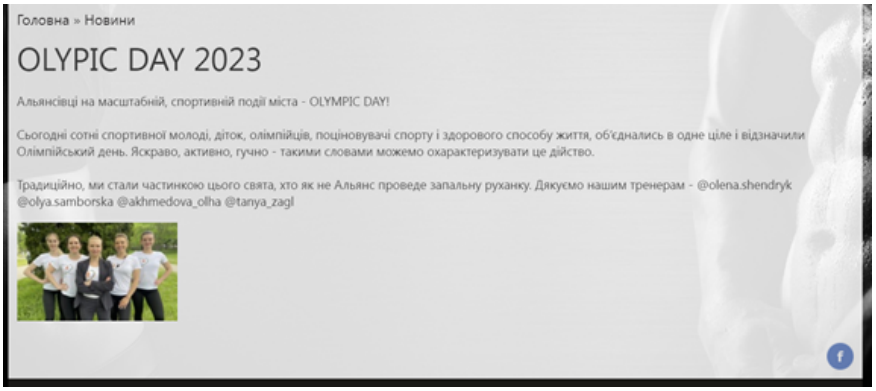
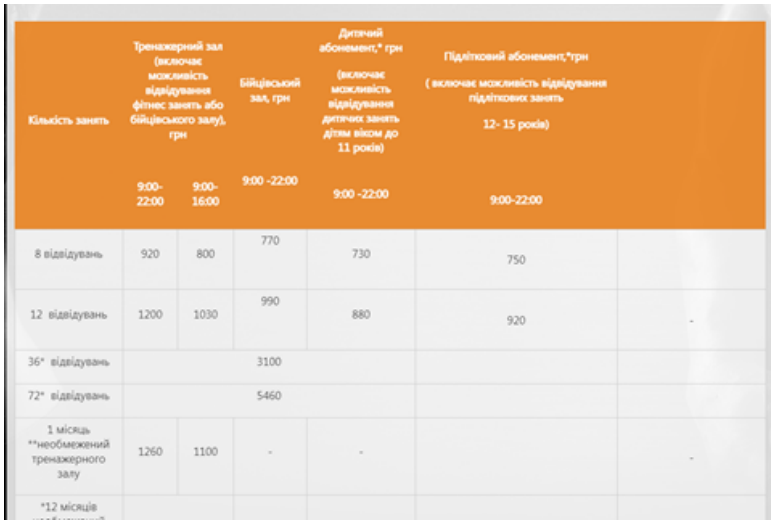


Рисунок 1.7 – Приклад статті на сайті

Сайт має інформативний хедер, де виділені всі основні категорії по яким можна перейти, але якщо перейти до «прайс-лист», то можна помітити, що таблиця, яка представляє перелік цін містить у собі багато лишньої інформації, а також жахливу дизайнерську реалізацію (рис. 1.8).



Кількість занять	Тренажерний зал (включає можливість відвідування фітнес занять або бійцівського залу), грн		Бійцівський зал, грн	Дитячий абонемент,* грн (включає можливість відвідування дитячих занять дітям віком до 11 років)	Підлітковий абонемент,*грн (включає можливість відвідування підліткових занять 12- 15 років)	
	9:00-22:00	9:00-16:00	9:00 -22:00	9:00 -22:00	9:00-22:00	
8 відвідувань	920	800	770	730	750	
12 відвідувань	1200	1030	990	880	920	-
36* відвідувань			3100			
72* відвідувань			5460			
1 місяць **необмежений тренажерного залу	1260	1100	-	-		-
*12 місяців						

Рисунок 1.8 – Таблиця розцінок спортклубу Альянс

Також, якщо перейти і переглянути розклад занять, то можна помітити, що одна таблиця виходить за рамки сайту (рис. 1.9).

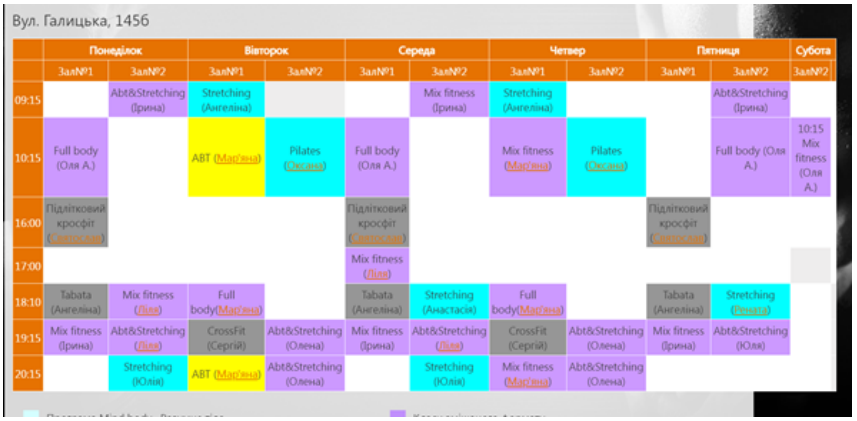


Рисунок 1.9 – Некоректне відображення таблиці

1.3 Постановка задачі

Розробка веб-сайту для спортивного клубу вимагає комплексного підходу, що враховує як помилки, так і успіхи конкурентів, а також результати дослідження, зробленого у підрозділі 1.1.

Завданням розробки веб-сайту є створення платформи, що максимально відповідає вимогам клієнтів та забезпечує їм зручність взаємодії з послугами та

ресурсами спортивного клубу. При розробці варто звернути увагу на наповнення сайту, зробити його максимально інформативним і легким для читання.

Також, вважаю доцільним реалізувати можливість додавання функції створення особистого кабінету в якому б користувач міг створити свій індивідуальний план тренувань.

Висновки до розділу 1

У розділі було проведено дослідження індустрії спортивних клубів та наявних конкурентів.

Основними складовими бізнесу спортивних клубів, як показало дослідження, є:

1. Наявність абонементів.
2. Різноманітні додаткові активності.
3. Наявність спортивного інвентарю.
4. Досвідчені тренери.
5. Привабливий зовнішній вигляд клубу.

Досліджуючи сайти конкурентів, було виявлено ряд спільних рис, серед яких були:

1. Структурована та логічна подача інформації.
2. Привабливий та інтуїтивний дизайн.
3. Адаптивність.
4. Строгість в стилі.

Їх варто взяти до уваги при розробці власного сайту, як стандарт розробки подібного роду сайтів, а також внести свої корективи:

– дати користувачам більше інформації про зал, включно з фото, замість короткої і місцями обмеженої адресою та контактами, інформацією у конкурентів;

– розробляти сторінки без лишньої інформації, яка б відволікала користувача від основного тексту;

– у подачі інформації про абонементи та розклад, користуватися локанічністю у висвітленні інформації та візуальною привабливістю для легшого сприйняття тексту.

Перейняти досвід можна у поданні інформації про тренерів, як це було зроблено на сайті SportLife: максимум вичерпної інформації лише за тим вийнятком, що інформацію, яка подається від лиця тренера наповнити інформативністю з метою дати зрозуміти користувачу, що даний тренер може зробити персонально для тебе. В чому він розбирається і який у нього досвід.

Чого не вистачало і що варто було б реалізувати – можливість створення персонального плану тренувань в середовищі сайту, що дало б більшу зацікавленість даним сайтом та стало б в пригоді відвідувачам.

РОЗДІЛ 2. ПРОЄКТУВАННЯ САЙТУ

2.1 Технології розробки

Реалізація завдання буде здійснюватися за допомогою таких основних мов програмування для конструювання сайтів, як:

1. CSS [17] – для стилізації сайту;
2. HTML [18] – для побудови сторінок та загальної структури сайту;
3. JavaScript [19] – для анімування ключових елементів.

Також, у розробці буде використовуватися мова програмування Python [4] – відома своєю простотою та читабельністю коду. Вона має широкий спектр бібліотек та фреймворків, що значно спрощує процес розробки. Python дозволяє писати менше коду для досягнення бажаного результату порівняно з іншими мовами програмування, такими як Java або C# [20]. Це зменшує час розробки та ризик помилок.

Сам же процес написання здійснюватиметься за допомогою фреймворку Django [5], який є одним з найпопулярніших веб-фреймворків для розробки веб-додатків. Архітектура Django базується на моделі MVC, де моделі відповідають за роботу з даними та логікою бази даних, представлення (Views) обробляють запити та відображають відповідний вміст. Шаблони (Templates) відповідають за відображення даних для користувача.

Він призначений для швидкої розробки веб-додатків з використанням принципів "замовник-програміст". Принцип полягає в тому, що фреймворк, як Django, намагається забезпечити рішення для типових завдань розробки веб-додатків, щоб програміст міг швидко створювати функціональність без необхідності писати код з нуля. Це означає використання стандартних практик, вбудованих модулів і готових рішень для типових завдань, що дозволяє розробнику зосередитися на специфічних функціях додатку, замість концентування уваги на рутинних задачах.

Django використовує об'єктно-реляційне відображення (ORM) для взаємодії з базою даних. Моделі Django визначають структуру даних, з якими працює додаток, і автоматично створюють SQL-запити для взаємодії з цими даними. Представлення Django відповідають за обробку запитів веб-клієнтів та відповідний вихід. Вони отримують дані від моделей та передають їх в шаблони для подальшого відображення. Він дозволяє розробникам легко розширювати функціональність додатку за допомогою сторонніх бібліотек та розширень.

Шаблони Django використовуються для відображення веб-сторінок. Вони містять HTML-код разом з вставками Django-коду, що дозволяє динамічно генерувати вміст на сторінці. Django використовує файл маршрутизації для визначення, які представлення будуть викликані для кожного URL-запиту. Це виконується за допомогою шаблонів маршрутизації, які відповідають за відповідність URL-адреси до певного представлення.

Крім того, Django надає зручний інтерфейс адміністратора, що спрощує керування контентом та даними. Django автоматично створює міграції, щоб зберегти структуру бази даних синхронізованою з визначенням моделей. Це дозволяє легко маневрувати зі структурою бази даних під час розвитку додатку. Django має вбудовану систему для аутентифікації користувачів, а також для надання їм доступу до певних частин додатку за допомогою системи прав.

Python та Django мають велику та активну спільноту розробників, що забезпечує швидке розв'язання проблем та підтримку. Документація для обох технологій є досить повною та зрозумілою, що дозволяє швидко засвоїти їх для нових розробок.

Базою даних для реалізації сайту була вибрана база даних SQLite [6] – вбудоване реляційне сховище даних, яке надає можливість зберігати, організовувати та отримувати доступ до даних. SQLite працює у вбудованому режимі, що означає, що весь движок бази даних інтегрований безпосередньо в програму, що використовує його. Це спрощує розповсюдження та використання. SQLite підтримується на багатьох операційних системах, включаючи

найпопулярніші: Windows, macOS, Linux, що дозволяє розробникам створювати програми, які працюють на різних пристроях.

Ця база даних має невеликий розмір, що дозволяє легко використовувати його в обмежених середовищах, таких як мобільні пристрої чи вбудовані системи. SQLite підтримує багато стандартних операцій SQL, що дозволяє виконувати запити, оновлення, вставки та видалення даних. SQLite підтримує транзакційну безпеку, що дозволяє виконувати групу операцій як атомарну одиницю, що забезпечує цілісність даних. SQLite має мало опцій конфігурації, що спрощує процес використання та управління базою даних. SQLite підтримує велику кількість одночасних підключень та операцій з даними, хоча для великих обсягів даних та високих навантажень можуть бути кращі альтернативи.

Середовищем розробки я вибрав VS Code який є простим у використанні та швидким у навчанні, що робить його більш привабливим для новачків. VS Code пропонує потужний, легкий та розширюваний редактор коду, який підтримує багато мов програмування. Однією з головних переваг VS Code є його екосистема розширень. Користувачі можуть легко налаштовувати редактор, додавши розширення для своїх потреб. Екосистема включає в себе тисячі розширень для підтримки різних мов програмування, інструментів розробки, тем оформлення.

VS Code [7] має вбудовану підтримку Git [8], що робить роботу з версіями та спільною роботою над проектами зручнішою. У порівнянні з IntelliJ IDEA [15] або PyCharm [16], VS Code має менші вимоги до ресурсів комп'ютера, що означає, що він може працювати добре навіть на менш потужних комп'ютерах. Встановлення та налаштування VS Code швидше, оскільки ви можете вибирати лише ті розширення, які вам потрібні, тоді як IDE від JetBrains [9] можуть потребувати додаткового часу на налаштування та завантаження. Робочий інтерфейс VS Code легший у вивченні, ніж аналогічний від продукту JetBrains.

VS Code також має велику спільноту користувачів, що активно розробляє розширення та надає підтримку через форуми та соціальні мережі. Крім того, цей редактор постійно оновлюється, отримуючи нові функції та поліпшення, що забезпечує актуальність і відповідність сучасним вимогам розробки.

2.2 Структура сайту

2.2.1 Загальна структура сайту спортзалу

Першим етапом в аналізі структури веб-сайту є ідентифікація головних сторінок, що складають його та його відображення у вигляді діаграми [10] (рис. 2.1). Головні розділи зазвичай відображають основні тематичні аспекти або функціональні зони сайту. У моєму випадку головними сторінками є:

1. Головна сторінка;
2. Сторінка з інформацією про зал;
3. Сторінка з інформацією про групові тренування;
4. Сторінка з інформацією про персональні тренування;
5. Сторінка з інформацією про інші активності у спортзалі;
6. Сторінка переліку статей сайту;
7. Сторінка з вмістом статті;
8. Сторінка переліку тренерів;
9. Сторінка з вмістом інформації про окремо взятого тренера;
10. Сторінка переліку абонементів;
11. Сторінка оформлення покупки абонемента.

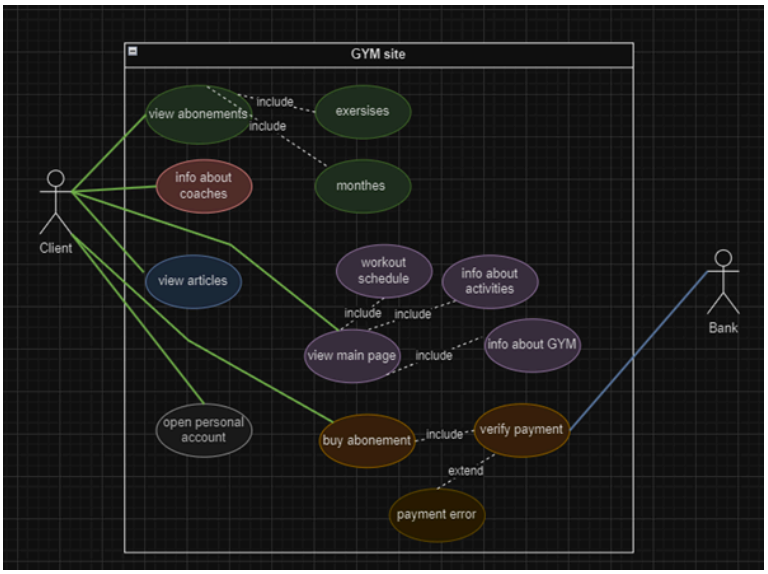


Рисунок 2.1 – UML діаграма сайту спортзалу

Навігація по сайту здійснюється через хедера(верхню частину сайту) та містить такі посилання на сторінки (рис. 2.2):

1. Головна сторінка(у вигляді логотипа спортзалу);
2. Тренери;
3. Розклад;
4. Статті;
5. Абонементи;
6. Особистий кабінет користувача.

Рисунок 2.2 – Навігація по сайту(хедер)

2.2.2 Головна сторінка

Зайшовши на сайт, відвідувача зустріне головний банер сторінки, який буде ілюструвати фото одного із частин спортзалу. Нижче знаходяться блоки, які пропонують користувачам дізнатися більше про різні аспекти спортзалу, такі як:

1. Групові тренування;
2. Персональні тренування;
3. Тренажерний зал;
4. Інші активності.

Ці блоки містять посилання на відповідні сторінки з ілюстрованими фото. На кожній з них, користувач може ознайомитися з детальною інформацією по обраній ним тематиці і сподіватися дізнатися те, що він очікує (рис. 2.3).

Візуальне оформлення цієї та подальших сторінок виконано у середовищі Figma [11], яке є найпопулярнішою платформою для створення та редагування дизайну різного роду програмних продуктів.

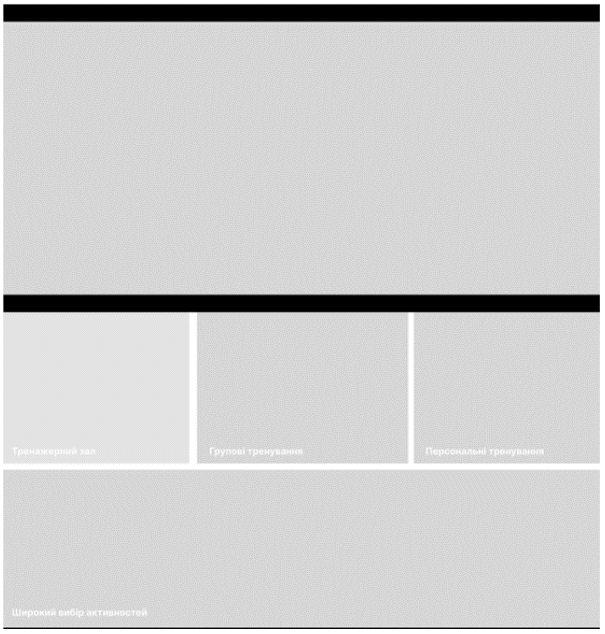


Рисунок 2.3 – Wireframe головного банера та блоки інформації

На головній сторінці також розміщений блок з розкладом тренувань, який вказує на час та типи доступних тренувань. Це допомагає відвідувачам швидко зорієнтуватися у графіку та вибрати зручний час для відвідування.

Для привертання уваги до професійного підходу до тренувань може бути розміщений інформаційний блок про тренерів. Цей блок дає зрозуміти відвідувачу, що спортзал має кваліфікованих тренерів, які готові завжди допомогти у будь-якому питанні та посилання на сторінку з докладною інформацією про кожного з них (рис. 2.4).

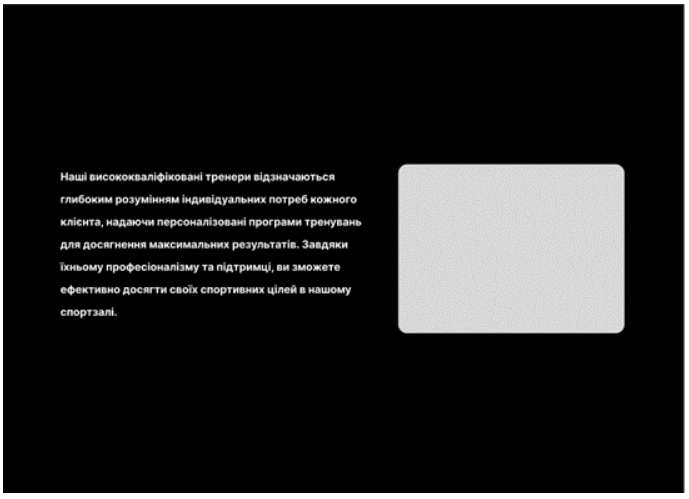


Рисунок 2.4 – Інформаційних блок про тренерів

Один з важливих елементів головної сторінки - блок з відгуками. Відгуки клієнтів допомагають підтвердити якість послуг спортзалу та створюють додатковий мотив для нових відвідувачів (рис. 2.5).

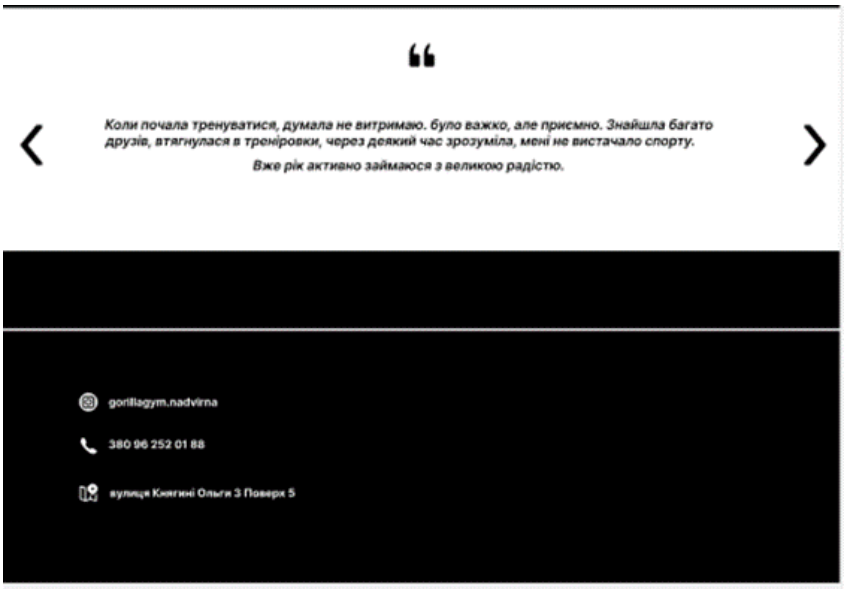


Рисунок 2.5 – Відгуки та футер сайту

На кожній сторінці сайту в футері розміщується контактна інформація, яка дозволяє відвідувачам зв'язатися з адміністрацією спортзалу для отримання додаткової інформації або консультації, а також вказує, де розміщується спортивний клуб (рис. 2.5).

2.2.3 Статті

На сторінці відвідувач може знайти перелік цікавих та інформативних статей на різні теми, пов'язані з покращенням своєї фізичної форми, тренуваннями, підготовкою до цих тренувань та іншим корисним контентом, який допоможе покращити заняття спортом (рис. 2.6).

Кожна окремо взята стаття супроводжується відповідним до контексту статті фото, що допомагає читачеві краще зрозуміти та візуалізувати інформацію, а також підсилює загальне враження від матеріалу.

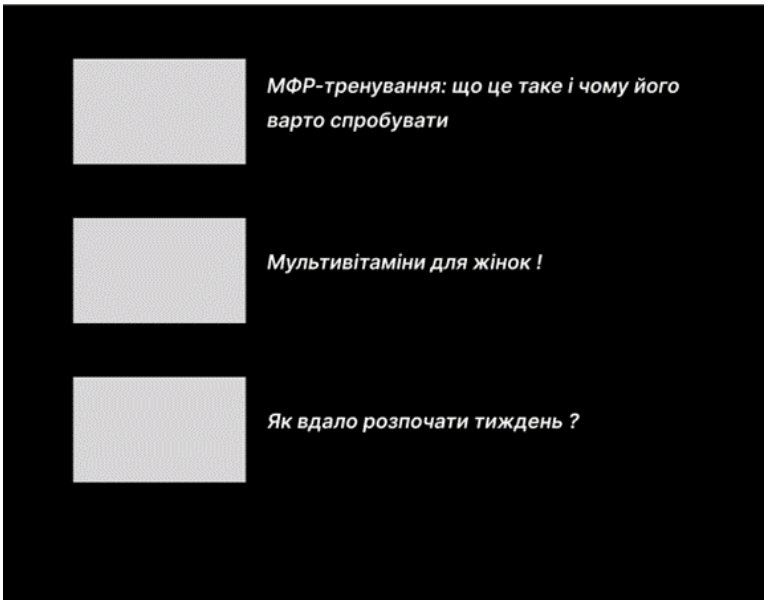


Рисунок 2.6 – Перелік статей

Натиснувши на будь-яку статтю, користувач попаде на сторінку, де міститься самий опис обраної теми (рис. 2.7).

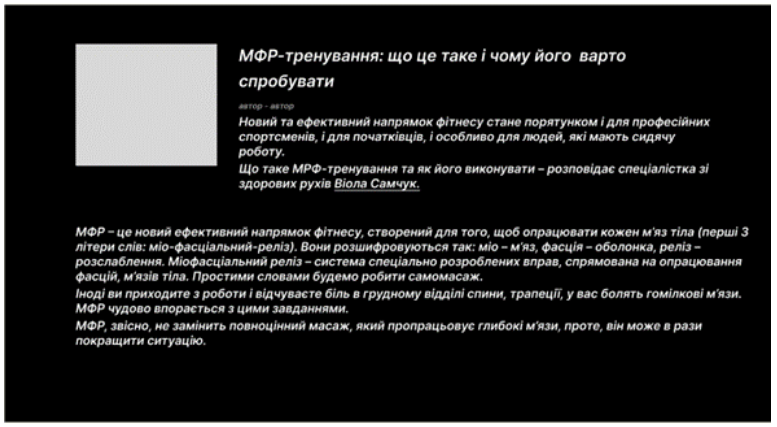


Рисунок 2.7 – Вигляд статті

2.2.4 Тренери

Перелік тренерів містить в собі інформацію про тренерське звання, досвід роботи та кількість проведених занять (рис. 2.8). Ця інформація покликана допомогти користувачу вибрати потрібного йому тренера, спираючись на найважливіше – досвід.

Рисунок 2.8 – Блоки з тренерами

Натиснувши кнопку “дізнатися більше”, можна отримати конкретнішу інформацію (рис. 2.9). Наприклад, про компетентність тренера у вигляді його спеціалізації, чим займається у спортзалі(веде групові заняття, відновлює після травм, учить боксу).

Рисунок 2.9 – Детальна інформація про тренера

2.2.5 Абонементи

Можна переглянути актуальні абонементи з описом того, що вони в собі містять і що буде доступне покупцю по ним у разі придбання. Розділяються вони на дві категорії: по заняттях та щомісячні (рис. 2.10).

Рисунок 2.10 – Вигляд абонементів

Вибравши один з них, користувача переадресовує на сторінку з оформленням покупки, де він повинен ввести дані карти і дочекатися підтвердження операції. Після чого, статус купленого абонемента буде відображатися у нього в особистому кабінеті, що допоможе користувачу завжди знати, що він оформив.

2.2.6 Особистого кабінету користувача

Структура особистого кабінету не є такою розлогою, як основного сайту, але несе в собі важливий функціонал, а саме – можливість створення персонального плану тренувань на основі вже наявного переліку вправ (рис. 2.11).

Для доступу в особистий кабінет від користувача вимагається створити свій обліковий запис на сторінці реєстрації. Після успішного проходження, користувач переадресовується на сторінку кабінету.

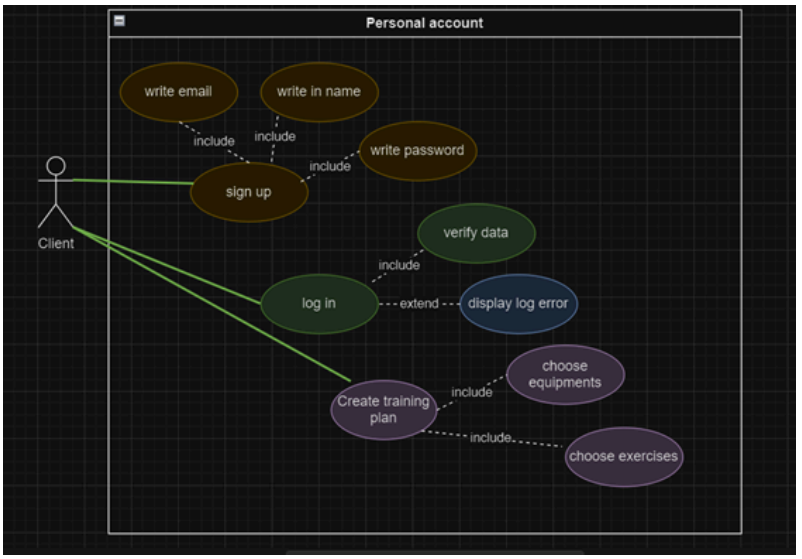


Рисунок 2.11 - UML діаграма особистого кабінету користувача

У блоці навігації, користувач може натиснути на кнопку під назвою “особистий кабінет” і попасти на сторінку реєстрації, де може перейти і на сторінку входу, якщо уже зареєстрований.

Для реєстрації потрібно буде ввести всього лиш такі дані, як: електронну пошту, ім’я та створити пароль (рис. 2.12).

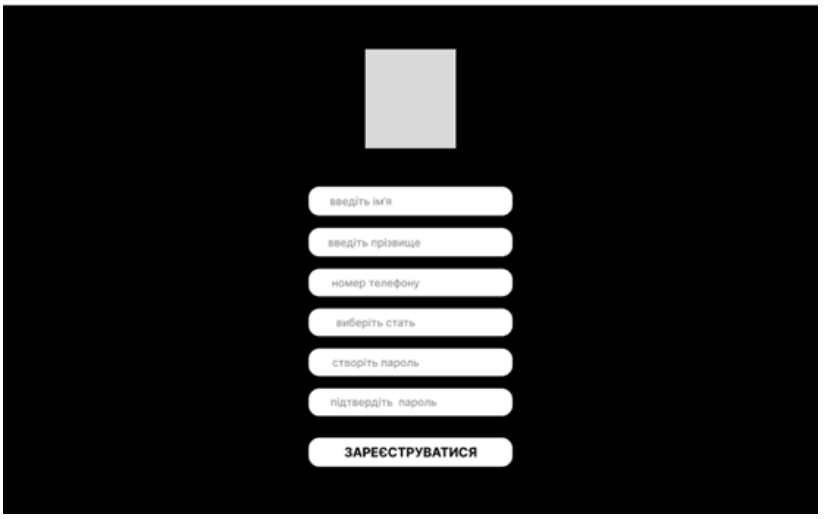


Рисунок 2.12 – Сторінка реєстрації

Завершивши процедуру входу/реєстрації, користувач попадає на саму сторінку кабінету, де можна створити свій план тренувань. Сам кабінет ділиться

на дві секції: верхня і нижня. У верхній дублюється ім’я та прізвище, вказані під час реєстрації, а також можна вибрати свій аватар, нижня ж присвячена створенню та відображенню плану тренувань (рис. 2.13).

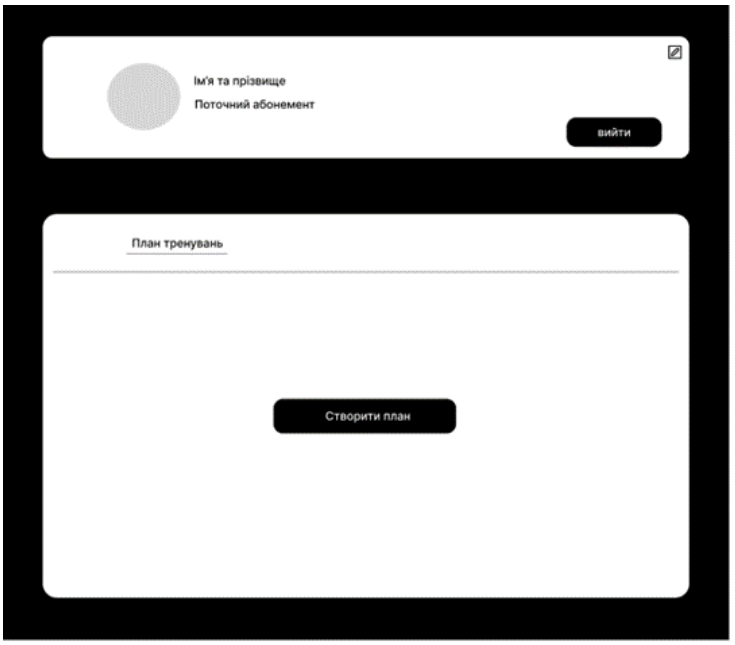


Рисунок 2.13 – Особистий кабінет користувача

У відповідній сторінці йому буде представлено різноманіття різних вправ та тренажерів, що допоможе сформувати дійсно ефективну програму занять, яка буде завжди доступна, зайшовши у кабінет (рис. 2.14).

Рисунок 2.14 – Створення плану тренувань

Користувач також може змінити порядок вправ, за бажанням, на відповідній сторінці, де розташований весь перелік вибраних попередньо вправ (рис. 2.15).

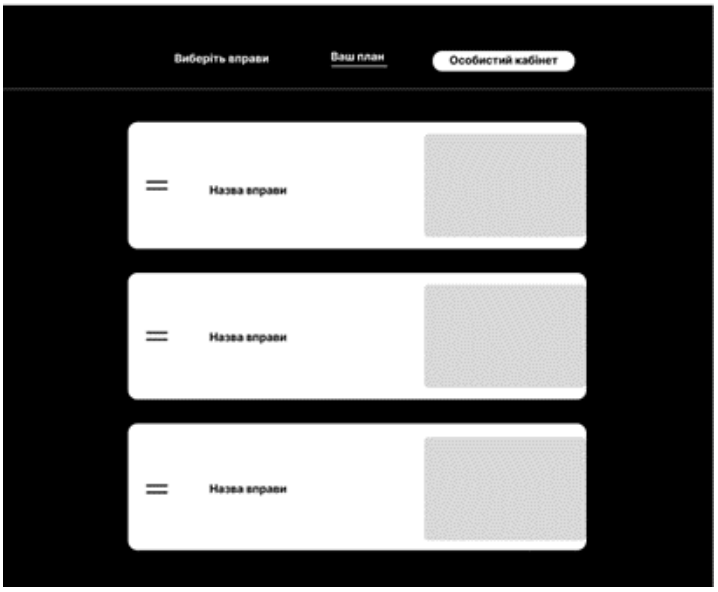


Рисунок 2.15 – Сторінка створеного плану тренувань

2.3 Технічна структура проєкту

Загальна ієрархія проєкту типова для проєктів на django. Включає в собі папки з аплікейшинами, які в собі уже місять html, css файли та інші. Мій проєкт має 2 аплікейшина – gymsite та users (рис. 2.16).

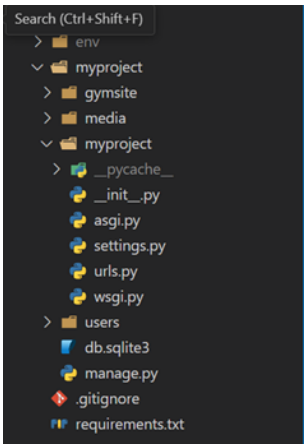


Рисунок 2.16 – Структура проєкту

Аплікейшин gymsite відповідає за розробку самого сайту(статичні сторінки та їх дизайн) та містить у собі мінімум роботи з бекендом. Звідси і назва папки. В той час users відповідає за розробку особистого кабінету користувача, містить у собі динамічні сторінки і являє собою таку собі бекенд частину проекту, бо саме в ній зосереджене функціональне навантаження сайту – реєстрація, авторизація, функціонал розробки та редагування планів тренувань.

2.3.1 Gymsite

Urls цього аплікейшина містять в собі посилання на всі сторінки сайту (рис. 2.17), в тому числі на сторінки пов'язані з реалізацією системи оплати.

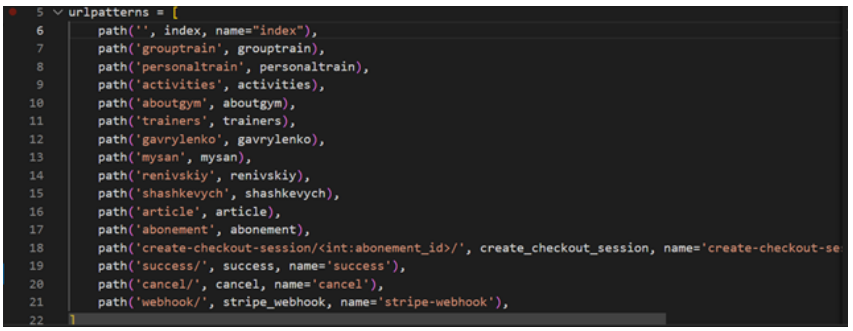


Рисунок 3.17 – Gymsite urls

Перелік templates містить всі html сторінки, реалізовані в даному аплікейшині, а папка static містить файли, відповідальні за дизайн цих сторінок (рис. 2.18).

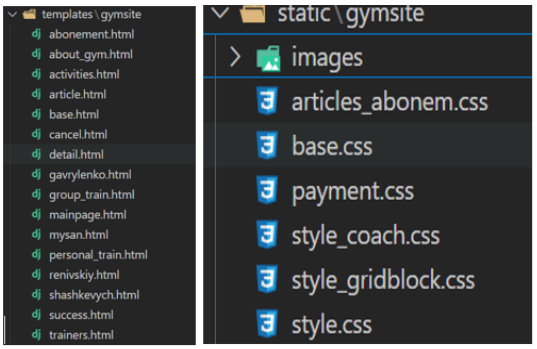


Рисунок 2.18 – Templates та Static

Папка models містить інформацію про 3 моделі, що використовуються у даному аплікейшині для зберігання та взаємодії з інформацією, а саме – статті(Article), абонементи(Abonement) та покупки(Purchase).

Модель Article містить у собі поле для заголовка статті, поле для фото та поле для самого вмісту статті (рис. 2.19).

Рисунок 2.19 – Модель Article

Також був використаний RichTextField, імпортований з бібліотеки ckeditor, що покращило роботу над текстом. RichTextField - це поле для введення тексту в Django, яке інтегрується з редактором CKEditor [12] для створення багатофункціональних текстових полів у вашій веб-додатку. CKEditor є одним з найпопулярніших редакторів WYSIWYG для веб-розробки. Поле RichTextField забезпечує можливість вводу і форматування тексту, вставка зображень, таблиць, посилань та інших медіа-елементів, що робить його ідеальним для створення контенту на сайті.

Модель Abonement містить поля для назви абонементу, ціни за абонемент, а також для опису абонемента (рис. 2.20). Важливою його складовою є поле type, що відповідає за тип абонемента і визначається вона списком TYPE_CHOICES.

Рисунок 2.20 – Модель Abonement

Модель Purchase містить поля користувачем до якого покупка буде прив’язана, абонементом, який користувач купив, електронною поштою, яка буде задіяна у процесі купівлі та поле про час здійснення покупки (рис. 2.21).

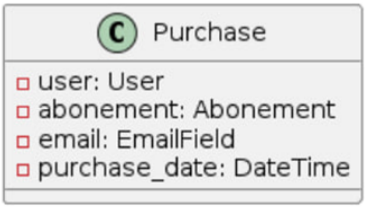


Рисунок 2.21 – Модель Purchase

Параметри on_delete=models.CASCADE у полі про користувача означають, що при видаленні користувача з бази даних, пов'язані з ним покупки будуть видалені також. null=True і blank=True означають, що це поле може бути порожнім (null) або не заповненим. У полі з абонементом міститься рядок, який означає, що якщо цей абонемент буде видалений з бази даних, то всі покупки, пов’язані з ним також будуть видалені(on_delete=models.CASCADE).

Такі моделі, як Abonement та Purchase буде тісно пов’язані з користувачем та між собою. Зв’язки між ними виглядають наступним чином: кожен користувач може мати безліч абонементів та безліч покупок цих абонементів, а кожному абонементу відповідає певна покупка користувача (рис. 2.22).

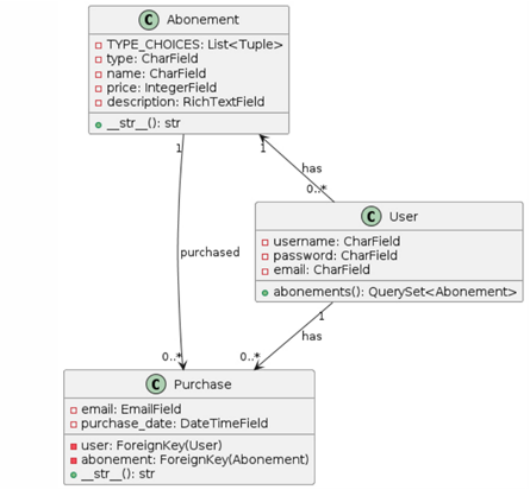


Рисунок 2.22 – Зв’язки між моделями аплікейшина gymsite

У views подане відображення всіх статичних сторінок сайту та логіка функціонування платіжної системи.

Для реалізації можливості оплачувати на сайті, був інтегрований платіжний сервіс Stripe [13]. Stripe забезпечує високий рівень захисту фінансових даних і особистої інформації клієнтів. Для цього використовуються різні методи, такі як шифрування даних за стандартом PCI DSS, двофакторна аутентифікація, виявлення шахрайства та інші технології безпеки.

Stripe пропонує зручний і легкий у використанні API для інтеграції платіжних можливостей в веб-сайти, мобільні додатки та інші платформи. Він складається всього із 2 ключів, які потрібно вписати у проєкт і потім використовувати у написанні логіки роботи.

Робота сервісу дуже проста. Коли клієнт здійснює платіж через Stripe, дані операції надходять на сервер Stripe через захищене з'єднання. Після цього Stripe обробляє платіж, перевіряючи карткові дані, виконуючи транзакцію і надсилаючи підтвердження про успішне здійснення оплати. У випадку успішної транзакції кошти перераховуються на рахунок магазину або бізнесу. Весь процес відбувається автоматично і максимально ефективно. Платежі через Stripe відбуваються миттєво, що дозволяє клієнтам одразу отримувати товари або послуги після здійснення оплати.

Stripe надає розширену аналітичну звітність щодо фінансових транзакцій, включаючи звіти про продажі, відшкодування, податки та іншу важливу інформацію, що дозволяє бізнесам ефективно керувати своїми фінансами та розвивати стратегії продажів.

2.3.2 Users

Аплікейшин Users має менше сторінок за аплікейшин Gymsite, тому обсяг templates для нього менший. Зате включає в собі такі важливі сторінки, як сторінка реєстрації та авторизації, сторінка особистого кабінету користувача та сторінка складання планів тренувань (рис. 2.23).

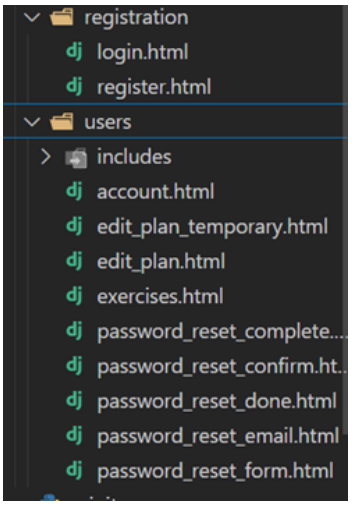


Рисунок 2.23 – Users templates

Urls для даного аплікейшина виглядають наступним чином (рис. 2.24).

```
urlpatterns = [
    path('account/', account, name='account'),
    path('exercises/', exercises, name='exercises'),
    path('edit_plan/', edit_plan, name='edit_plan'),
    path('delete_plan/', delete_plan, name='delete_plan'),
    path('add_exercise_to_plan/', add_exercise_to_plan, name='add_exercise_to_plan'),
    path('remove_exercise_from_plan/', remove_exercise_from_plan, name='remove_exercise_from_plan'),
    path('save_temporary_plan/', save_temporary_plan, name='save_temporary_plan'),
    path('remove_temporary_exercise/', remove_temporary_exercise, name='remove_temporary_exercise'),
    path('update_temporary_quantity/', update_temporary_quantity, name='update_temporary_quantity'),
    path('update_quantity/', update_quantity, name='update_quantity'),
    path('update_exercise_order/', update_exercise_order, name='update_exercise_order'),
    path('register/', RegisterView.as_view(), name='register'),
    path('password_reset/', CustomPasswordResetView.as_view(), name="password_reset"),
    path('password_reset/done/', CustomPasswordResetDoneView.as_view(), name="password_reset_done"),
    path('reset/<uidb64>/<token>/', CustomPasswordResetConfirmView.as_view(), name="password_reset_confir
    path('reset/done/', CustomPasswordResetCompleteView.as_view(), name="password_reset_complete"),
```

Рисунок 2.24 – Users urls

Forms.py відповідає за форму для реєстрації користувача, яка просто розширює функціонал заздалегідь реалізованої форми реєстрації – полем для вводу електронної пошти (рис. 2.25).

```
django-gym > myproject > users > forms.py > RegisterForm > Meta
1  from django.contrib.auth.forms import UserCreationForm
2  from django.contrib.auth.models import User
3
4  class RegisterForm(UserCreationForm):
5      class Meta(UserCreationForm.Meta):
6          model = User
7          fields = UserCreationForm.Meta.fields + ('email', )
```

Рисунок 2.25 – Users Forms.py

Views включає код для роботи з функціоналом реєстрації, авторизації, зміни пароля, створення та редагування планів, а моделі аплікейшина містять моделі для вправ та плану користувача.

Моделями даного аплікейшина є модель вправ(Exercise) та модель плану тренувань(UserExercisePlan).

Модель Exercise містить поля назвою вправи, фото, яке б супроводжувало вправу, поле для іконки, яка б допомогла користувачу зрозуміти, що можна зробити з вправою та поля type для розділення вправ на типи, в залежності від частини тіла, яку користувач хоче натренувати (рис. 2.26).

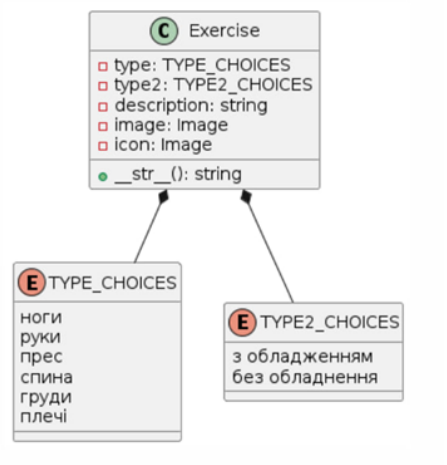


Рисунок 2.26 – Модель Exercise

Модель UserExercisePlan має поле, що зв’язує план тренувань з конкретним користувачем. Коли користувач видаляється (on_delete=models.CASCADE), всі пов'язані з ним плани тренувань також видаляються. Поле exercise зв'язує цей план тренувань з конкретною вправою. Так само, як і з користувачем, коли вправа видаляється, вона також видаляється з усіх планів тренувань. Також модель має поле для введення кількості повторень для кожної вправи, поле відповідальне за порядковий номер вправи у персональному плані тренувань користувача, поле з назвою плану тренувань.

Поле added_on типу DateTimeField, яке автоматично встановлює час додавання запису до бази даних. Plan_id типу UUIDField, яке створює унікальний

ідентифікатор для кожного плану тренувань. При створенні нового запису, це поле автоматично заповнюється випадковим унікальним значенням UUID (рис. 2.27).

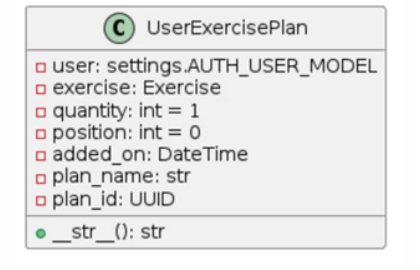


Рисунок 2.27 – Модель UserExercisePlan

Зв’язок між цими моделями виглядає таким чином, що кожна вправа належить будь-якому плану тренувань, а сам план тренувань належить користувачу, який його формував (рис. 2.28).

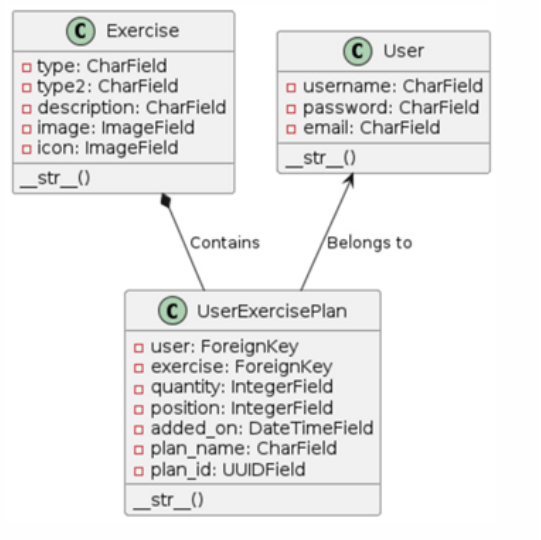


Рисунок 2.28 – Зв’язки між моделями аплікейшина users

Висновки до розділу 2

У цьому розділі було докладно описано прототип сайту та структуру проекту, що включає основні компоненти: views, urls, models і templates. Було проведено детальний аналіз кожної моделі програми, визначено їхні атрибути та

методи, а також продемонстровано взаємозв'язки між моделями за допомогою діаграми зв'язків.

Структура проекту була організована таким чином, щоб забезпечити максимальну модульність і масштабованість додатку. Визначення URL-адрес забезпечує зручну навігацію по сайту, а views реалізують логіку відображення даних, взаємодіючи з моделями та шаблонами. Моделі представляють дані та бізнес-логіку, а шаблони відповідають за відображення інтерфейсу користувача, забезпечуючи привабливий та функціональний дизайн. Діаграма зв'язків між моделями надала наочне уявлення про структуру даних і їх взаємодію, що допомагає краще розуміти та управляти складністю проекту.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ФУНКЦІОНАЛУ

3.1 Розробка сторінок сайту

Код сайту міститься на сторінці GitHub [14], де можна відслідкувати прогрес розробки по комітах та в додатках.

3.1.1 Головна сторінка та похідні від неї сторінки

Зайшовши на сайт, відвідувача зустрічає головний банер із зображенням спортзалу, а також хедер по якому користувач може взаємодіяти з вмістом сайтом (рис. 3.1). Наприклад, перейти до розкладу чи переглянути статтю, придбати абонемент чи ознайомитися з тренерським складом.

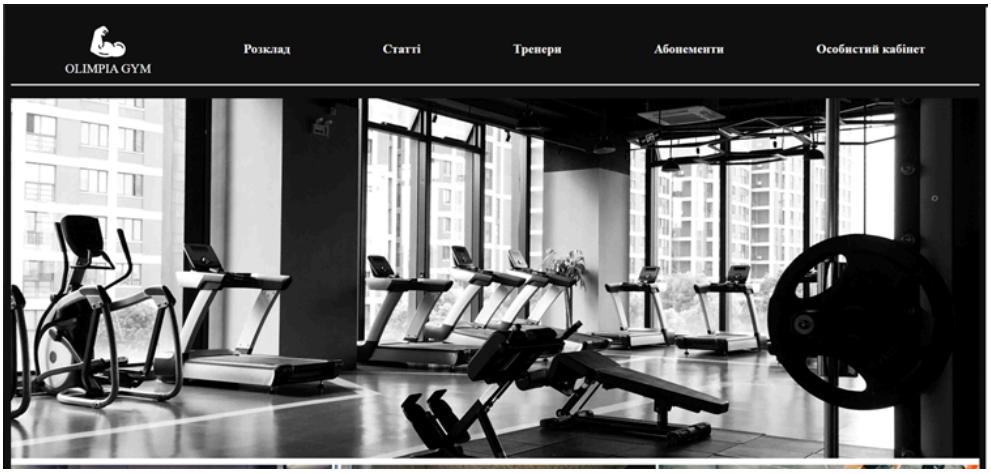


Рисунок 3.1 – Головний банер сайту та хедер

Хедер складається із логотипу, який веде на головну сторінку, розкладу, який переміщає користувача до тої частини сайту, де розміщується розклад занять у спортзалі, сторінки зі статтями, тренерами, актуальними абонементами та сторінки, що веде у особистий кабінет користувача за допомогою властивості тега

<a>, а саме атрибута href. Всередині цього атрибута поміщається посилання на відповідну сторінку:

```
     <header>
<nav>
<div>
<a href="/mainpage">
<img width="50px" height="auto" src="{% static 'gymsite/images/logo.png'
%}" alt="logo" />
<span>OLIMPIA GYM</span>
</a>
<a href="/mainpage/#table"><h4>Розклад</h4></a>
<a href="/mainpage/article"><h4>Статті</h4></a>
<a href="/mainpage/trainers"><h4>Тренери</h4></a>
<a href="/mainpage/abonement"><h4>Абонементи</h4></a>
<a href="{% url 'login' %}"><h4>Особистий кабінет</h4></a>
</div>
</nav>
</header>
```

Нижче інформативні блоки з тим, що пропонує спортзал своїм відвідувачам. Натиснувши на один із них, користувача переадресовують на відповідну сторінку, де він отримує детальнішу інформацію про зал, види активностей, доступних у залі, про персональні та групові тренування (рис. 3.2).

Рисунок 3.2 – Інформаційні блоки про послуги спортзалу

Працюють ці блоки так само, як і елементи хедера сайту – за допомогою атрибута href. Також ці блоки містять зображення, які відображаються на сайті за допомогою атрибута src, HTML тега img:

```
      <div class="box">
<div class="g_container">
<div class="gym_zal">
<a href="/mainpage/aboutgym">
<img src="{% static "gymsite/images/gym_zal.jpg" %}" alt="gym_zal">
<span><strong>Тренарежний зал</strong></span>
</a>
</div>
```

Під цими блоками розташований статичний розклад занять до якого можна перейти з хедеру сайту (рис. 3.3).

Рисунок 3.3 – Розклад занять на сайті

Розклад виконаний за допомогою табличних тегів HTML, де міститься вся інформація, потрібна користувачу. Наприклад, в цьому фрагменті коду написана верхня частина таблиці з днями тижня:

```
      <div id="table" class="table_component">
<table>
<caption><h1>РОЗКЛАД</h1></caption>
```

```
<thead>
<tr>
<th><h3>ПН</h3></th>
<th><h3>ВТ</h3></th>
<th><h3>СР</h3></th>
<th><h3>ЧТ</h3></th>
<th><h3>ПТ</h3></th>
</tr>
</thead>
```

В цьому фрагменті коду наведено, як записувалося основне наповнення таблиці розкладу з видами активностей на певний день та часом:

```
<tbody>
<tr>
<td><h3>Бокс</h3><br><h4>13:00</h4></td>
<td><h3>Фітнес</h3><br><h4>12:30</h4></td>
<td><h3>Дитяча Аеробіка</h3><br><h4>14:00</h4></td>
<td><h3>Фітнес</h3><br><h4>13:00</h4></td>
<td><h3>Бокс</h3><br><h4>14:50</h4></td>
</tr>
<tr>
```

Кожен сайт незалежно від тематики містить інформацію про розіграші, знижки, подарунки. Нижче розкладу знаходиться саме такий інформаційний блок, який дає змогу користувачу отримати щось. В даному випадку, знижку на абонемент при заданих умовах (рис. 3.4). Вся інформація даного блоку міститься всередині html коду.

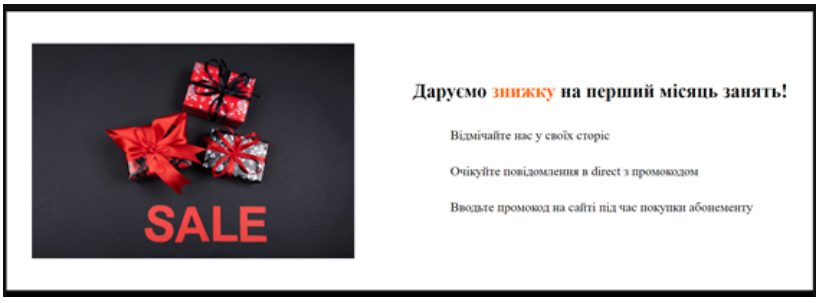


Рисунок 3.4 – Інформаційний блок

Одразу після нього користувач може прочитати про тренерів спортзалу, а потім одразу перейти на відповідну сторінку, щоб дізнатися більше (рис. 3.5).

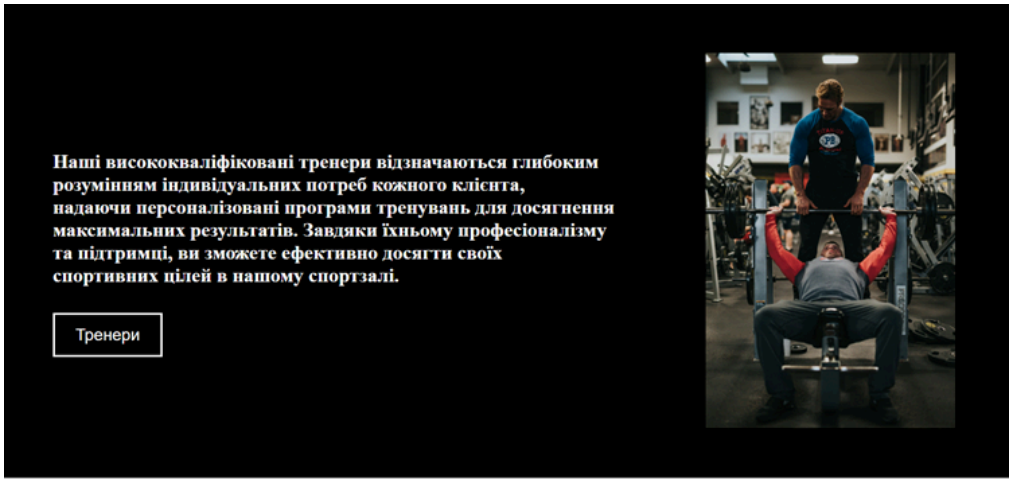


Рисунок 3.5 – Інформаційний блок про тренерів

Вся інформація блоку міститься там же, де й інформація попереднього блоку про знижки та акції.

Важливою складовою сайтів є відгуки користувачів чи клієнтів про зал. Такі відгуки доступні і на цьому сайті. Вони динамічні, представлені у вигляді слайдера, який можна гортати вліво і вправо. Індикатор у вигляді крапок нижче самого слайдера, показує на якому відгуку користувач перебуває у поточний момент і скільки всього доступно відгуків для перегляду (рис. 3.6). Динамічність реалізована лише за допомогою HTML та CSS, без допомоги JavaScript.

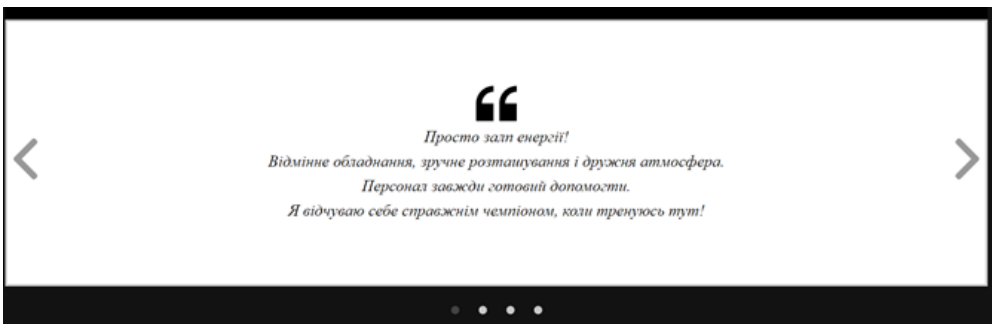


Рисунок 3.6 – Відгуки на сайті

Гортання відбувається за допомогою стрілок по обидва боки контейнера відгуків і здійснюється, коли програма реєструє натискання на одну із цих кнопок. Натискання фіксується за допомогою :checked:

```
#slide1:checked ~ #slides .inner {
margin-left: 0;
}
#slide2:checked ~ #slides .inner {
margin-left: -100%;
}
#slide3:checked ~ #slides .inner {
margin-left: -200%;
}
#slide4:checked ~ #slides .inner {
margin-left: -300%;
}
```

Коли користувач змінює слайди, індикатор, виконаний у вигляді чотирьох крапок нижче блоку відгуків – змінює свій вигляд у залежності від положення користувача на слайді за допомогою :checked:

```
#slide1:checked ~ #bullets label:nth-child(1),
#slide2:checked ~ #bullets label:nth-child(2),
#slide3:checked ~ #bullets label:nth-child(3),
#slide4:checked ~ #bullets label:nth-child(4) {
background: #444;
}
```

Закриває головну сторінку простий футер із основною інформацією про спортзал: сторінкою в месенджері Instagram, номером телефону менеджера спортзалу та актуальною адресою, по якій спортзал можна з лешкістю знайти (рис. 3.7).

Рисунок 3.7 – Футер сайту

Натиснувши на інформаційний блок під назвою “Тренажерний зал”, користувача перенаправляє на детальну сторінку про спортзал з усією інформацією про обладнання, асортимент активностей та фото (рис. 3.8). Перегляд фото здійснюється по верхніх кнопкам, які при наведені змінюють колір із сірого на білий.

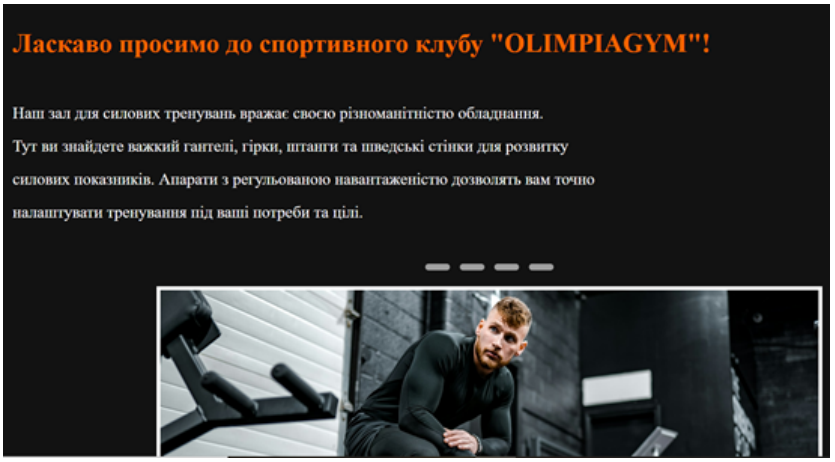


Рисунок 3.8 – Сторінка про тренажерний зал

 - блок з класом info, що містить інформацію про спортивний клуб. Параграф з описом тренажерного залу (<p>) та його обладнання включає елементи
 для переносу рядків:

```
<div class="info">
<h1>Ласкаво просимо до спортивного клубу "OLIMPIAGYM"!</h1>
<p>Наш зал для силових тренувань вражає своєю різноманітністю
обладнання.<br>
```

```
Тут ви знайдете важкий гантелі, гірки, штанги та шведські
стінки для розвитку<br>
силових показників. Апарати з регульованою навантаженістю
дозволять вам точно<br>
налаштувати тренування під ваші потреби та цілі.</p>
</div>
```

: Блок з класом slider, що містить слайдер зображень та об'єднує всі слайди та радіо-кнопки. <input type="radio" name="r" id="r1" checked> - радіо-кнопки для перемикання між слайдами. Перша кнопка (з id="r1") встановлена як активна (checked):

```
<div class="slider">
<div class="slides">
<input type="radio" name="r" id="r1" checked>
<input type="radio" name="r" id="r2">
<input type="radio" name="r" id="r3">
<input type="radio" name="r" id="r4">
<div       class="slide       s1"><img       src="{%       static
'gymsite/images/slider-gym1.jpg' %}" alt="slider-gym"></div>
<div          class="slide"><img          src="{%          static
'gymsite/images/slider-gym2.jpg' %}" alt="slider-gym"></div>
<div          class="slide"><img          src="{%          static
'gymsite/images/slider-gym3.jpg' %}" alt="slider-gym"></div>
<div          class="slide"><img          src="{%          static
'gymsite/images/slider-gym4.jpg' %}" alt="slider-gym"></div>
```

Перейшовши на сторінку групових тренувань, користувач отримує вичерпну інформацію по даній активності (рис. 3.9). Для зручності все основне було винесено у спеціальний блок з відповідними, до кожної окремо взятої інформації, іконками на білому фоні, що виділяє дану інформацію з-поміж іншого тексту сторінки.

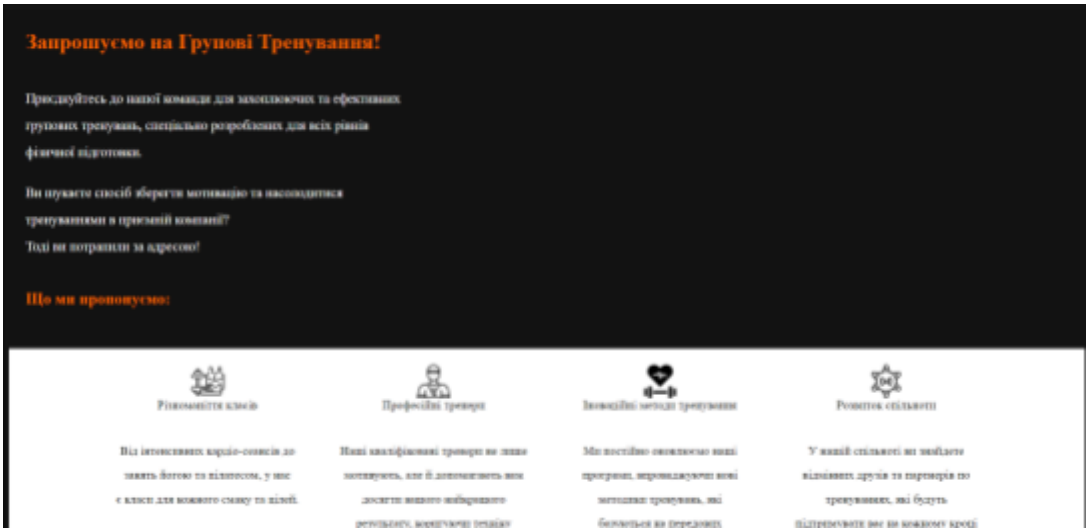


Рисунок 3.9 – Сторінка групових тренувань

Сторінка персональних тренувань та сторінка про доступні активності у спортзалі – виконані по тому ж принципу, де вся важлива інформація винесена у спеціальний блок на білому фоні.

3.1.2 Сторінки зі статтями

Якщо перейти на сторінку зі статтями сайту, то можна буде побачити всі доступні для прочитання статті, а натиснувши на них – відкрити вміст та перейти до прочитання (рис. 3.11).

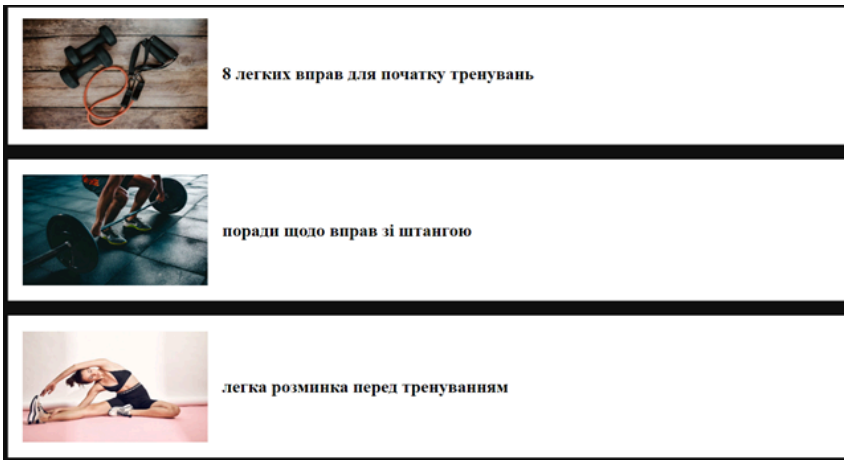


Рисунок 3.11 – Перелік статей на сайті

Вся інформація про статті, включно із назвою, картинкою та вмістом – містяться у базі даних у моделі Article. Щоб відтворити доступні статті, веб-браузер відправляє HTTP-запит у базу даних з метою отримати потрібні ресурси для відображення. Змінна items, використовуючи ORM Django, отримує всі об'єкти з моделі Articles, а objects.all() повертає. За допомогою функції render виконується обробка шаблону:

```
def article(request):
 items = Articles.objects.all()
 context = {
 'items':items
 }
 return render(request, "gymsite/article.html", context)
```

Розгорнувши статтю, можна помітити те саме фото, що й у блоці статті, а також детальну інформацію, де виділена основне на що варто звернути увагу користувачу при прочитанні (рис. 3.12).

Рисунок 3.12 – Вигляд статті у відкритому вигляді

Функція articleId(request, id) отримує HTTP-запит як вхідний параметр request і додатковий параметр id, який визначає ID конкретної статті. Метод get використовується для отримання одного об'єкту з моделі Articles, де значення поля

id відповідає значенню змінної id. Якщо об'єкт з таким id не буде знайдений, буде викинуто виключення DoesNotExist.

Створюється словник context, який містить ключ 'item', прив'язаний до конкретної статті, знайденої в базі даних, а за допомогою render обробляється шлях до шаблону HTML, щоб вивести сторінку для користувача:

```
def articleId(request, id):
item = Articles.objects.get(id=id)
context = {
'item':item
}
return render(request, "gymsite/detail.html", context)
```

Додавання та редагування інформації для статей здійснюється через адмін панель django сайту (рис. 3.13). Як тільки зберегти зміни, вони миттєво відобразяться на сайті і стаття чи її вміст обробляються відповідно до внесених змін.

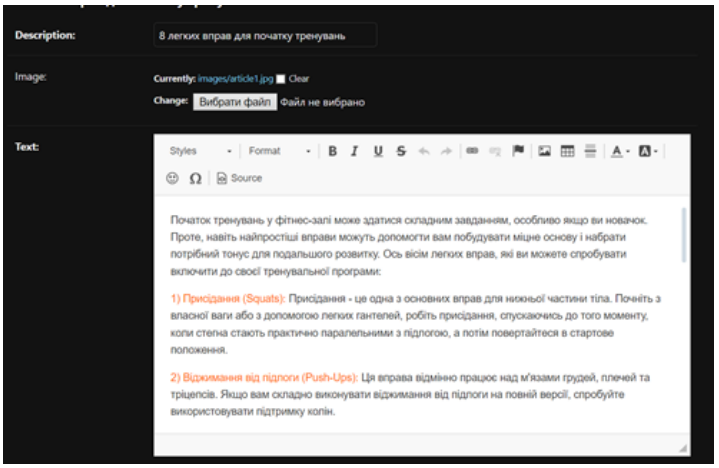


Рисунок 3.13 – Вигляд статей у адмін панелі

3.1.3 Сторінки з тренерами

Перше, що бачитиме користувач при відкритті сторінки з тренерами – це фото тренера, категорія тренера, кількість проведених занять та досвід роботи

(рис. 3.14). Ця інформація уже дасть розуміння про компетенцію тренера та допоможе зрозуміти чи варто довіряти йому, чи ні.

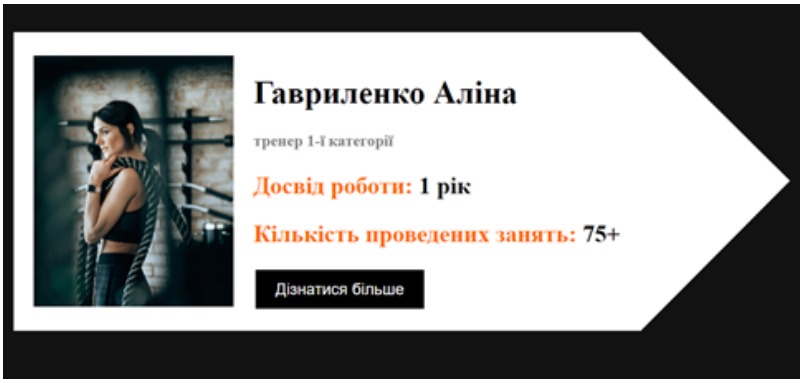


Рисунок 3.14 – Вигляд блоку тренера

Детальну інформацію про сертифікати, активності якими займається тренер та коротку інформацію про саму людину, користувач зможе отримати натиснувши на кнопку “Дізнатися більше” (рис. 3.15). Ключові слова було виділено спеціальним кольором для концентрації уваги користувача на найважливішому з написаного.

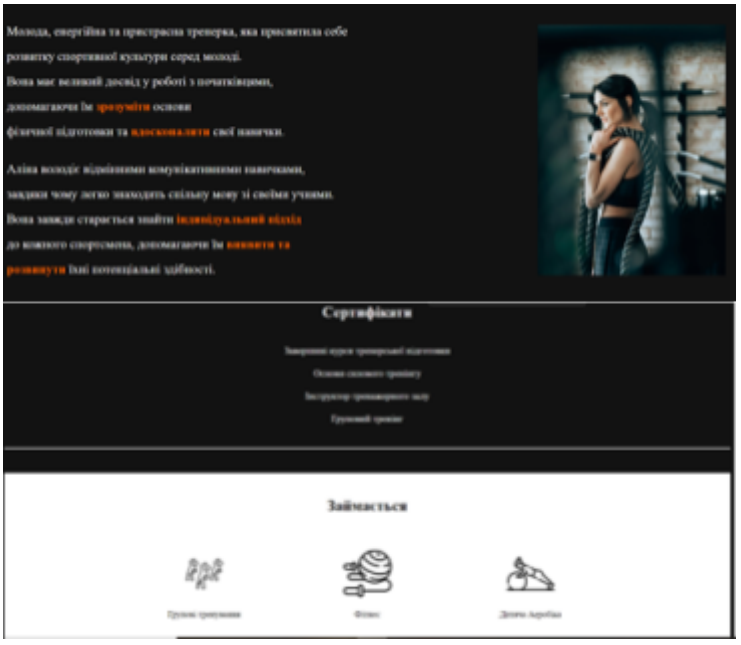


Рисунок 3.15 – Вигляд детальної сторінки тренера

3.1.4 Абонементи

На сторінці абонементів представлені всі актуальні, на даний час для цього спортзалу, абонементи, які пропонують користувачам різні конфігурації послуг, в залежності від виду. Абонемент ділиться на щомісячні та на певну кількість тренувань (рис. 3.16).

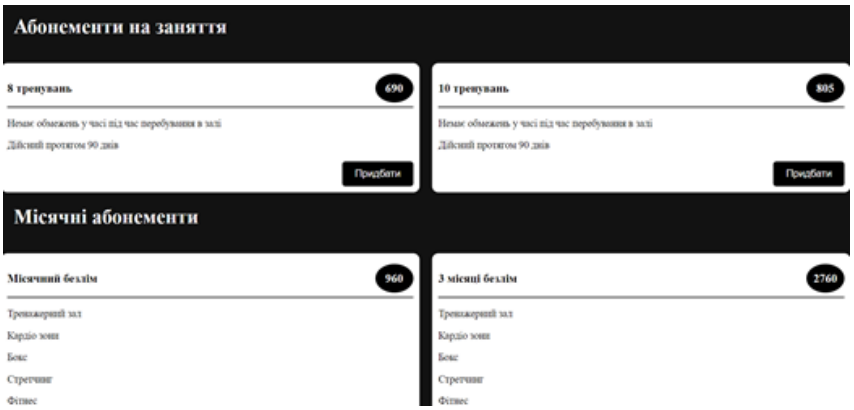


Рисунок 3.16 – Доступні абонементи для покупки

За допомогою функції Abonement.objects.all(), змінна items отримує всю доступну інформацію із моделі Abonement. Метод filter змінних exercise_abonements та monthly_abonements сортує абонементи по двом типам, щоб потім коректно відобразити на сторінці - абонементи занять розмістити зверху, місячні знизу (рис. 3.16).

Змінна context містить у собі всі дані для передачі шаблон, включно з інформацією про кожен абонемент. Редагування всієї інформації здійснюється через адмін панель. Там же можна додати новий абонемент, щоб потім розмістити його на сайті:

```
def abonement(request):
items = Abonement.objects.all()
exercise_abonements = Abonement.objects.filter(type='заняття')
monthly_abonements = Abonement.objects.filter(type='місяці')
context = {
```

```
'items':items,
'exercise_abonements': exercise_abonements,
'monthly_abonements': monthly_abonements
}
return render(request, "gymsite/abonement.html", context)
```

Як тільки користувач хоче купити якийсь із абонементів, він зможе це зробити, натиснувши на кнопку «Придбати». Купівля здійснюватиметься за допомогою сервісу Stripe, який був успішно інтегрований у даний сайт.

Використовуючи ORM, змінна abonement отримує об'єкт абонемента за переданим ідентифікатором abonement_id:

```
def create_checkout_session(request, abonement_id):
abonement = Abonement.objects.get(id=abonement_id)
try:
print("Creating Stripe session with metadata:", {'abonement_id': str(abonement_id)})
```

Метод stripe.checkout.Session.create створює сесію Stripe з наступними параметрами:

1) payment_method_types: типи методів оплати (в даному випадку тільки картка);

2) line_items: список предметів для оплати, що містить дані про ціну, валюту, дані про продукт, назву абонемента та суму. Кількість товару(в даному випадку тільки один абонемент, бо бізнес-логікою передбачена покупка лише одного абонементу за раз), електронна пошта:

```
line_items=[
{
'price_data': {
'currency': 'uah',
'product_data': {
'name': abonement.name,
},
```

```
'unit_amount': abonement.price * 100,
},
'quantity': 1,
},
],
mode='payment',
customer_email=request.POST.get('email'),
```

У разі успішної оплати, користувача перенаправляє на відповідну сторінку, а функція success обробляє успішне завершення платежу через систему Stripe. Вона перевіряє наявність параметра session_id в GET-запиті, якщо він присутній:

```
session = stripe.checkout.Session.retrieve(session_id)
abonement_id = session.metadata.get('abonement_id')
if abonement_id:
abonement = Abonement.objects.get(id=abonement_id)
return render(request, "gymsite/success.html", {"abonement":
abonement})
else:
return HttpResponse('abonement_id не знайдено в метаданих',
status=400)
except stripe.error.StripeError as e:
return HttpResponse(str(e), status=400)
except Abonement.DoesNotExist:
return HttpResponse('Абонемент не знайдений', status=404)
else:
return HttpResponse('session_id не передано', status=400)
```

Окремий код приймає події від Stripe, перевіряє їх автентичність і обробляє тип події checkout.session.completed, щоб створити запис про покупку у базі даних. Цей код містить змінну payload, яка містить тіло запиту, яке надходить від Stripe, а sig_header - підпис, який використовується для перевірки автентичності запиту. try-except перевіряється автентичність події, використовуючи підпис і секретний ключ вебхуків.

```
@csrf_exempt
 def stripe_webhook(request):
 payload = request.body
 sig_header = request.META['HTTP_STRIPE_SIGNATURE']
 try:
 event = stripe.Webhook.construct_event(
 payload, sig_header, settings.STRIPE_WEBHOOK_SECRET
 )
```

Якщо перевірка не проходить, повертається статус 400 (Bad Request). Якщо тип події checkout.session.completed, витягується інформація про сесію: abonement_id з метаданих і customer_email з даних сесії. Якщо обидва значення (abonement_id і customer_email) присутні, функція намагається знайти відповідний Abonement. Якщо він існує, створюємо новий запис про покупку (Purchase). Якщо Abonement не знайдений, повертається статус 404 (Not Found). У разі успішної обробки запиту повертається статус 200 (OK):

```
if event['type'] == 'checkout.session.completed':
session = event['data']['object']
abonement_id = session.get('metadata', {}).get('abonement_id')
customer_email = session.get('customer_email')
if abonement_id and customer_email:
try:
abonement = Abonement.objects.get(id=abonement_id)
Purchase.objects.create(email=customer_email,
abonement=abonement)
except Abonement.DoesNotExist:
return HttpResponse(status=404)
return HttpResponse(status=200)
return HttpResponse(status=200)
```

Візуально весь цей процес виглядає так: користувач натискає на кнопку «Придбати» і переадресовується на сторінку оплати, де повинен ввести свою електронну пошту, дані карти та ім’я. Після чого підтвердити операцію (рис.

3.17). У лівій частині форми оплати містяться дані про абонемент, який цей конкретний користувач купує та суму оплати.

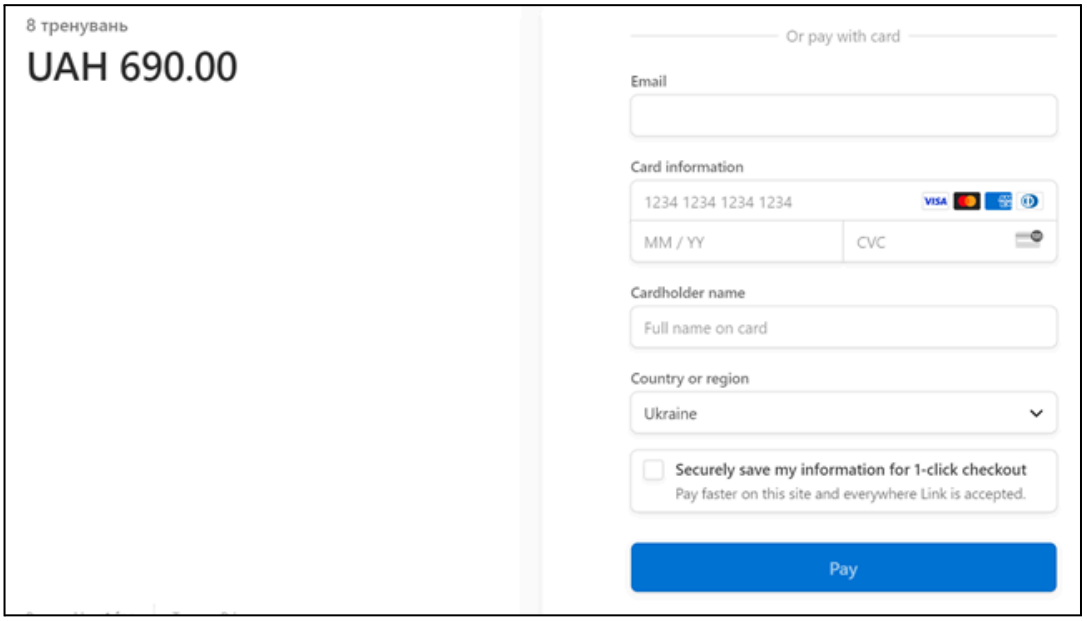


Рисунок 3.17 – Форма вводу реквізитів для оплати

У разі успішної оплати, на сайті Stripe, у відповідній сторінці про відслідковування платежів, з’явиться запис про успішну оплату того чи іншого користувача з інформацією, хто здійснив покупку(ім’я та електронна пошта), а також сума операції (рис. 3.18).

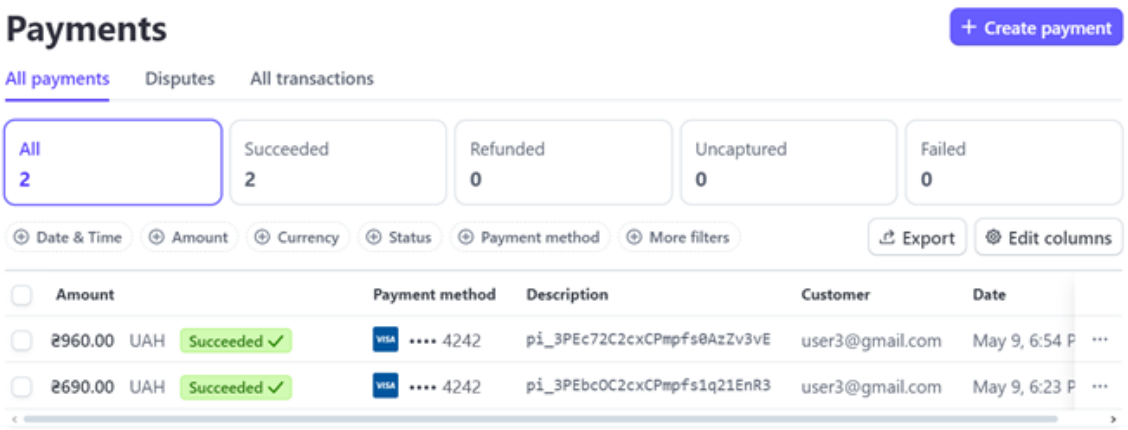


Рисунок 3.18 – Дані про оплату із сайту Stripe

3.2 Особистий кабінет користувача

3.2.1 Реєстрація, авторизація, зміна паролю

Коли користувач захоче зайти у свій персональний кабінет – його спочатку перекине на сторінку авторизації, де він зможе ввести свій нікнейм та пароль(якщо зареєстрований) або перейти до реєстрації по посиланню під формою. По натисканню на логотип сайту, користувач буде переадресований на головну сторінку сайту (рис. 3.19).

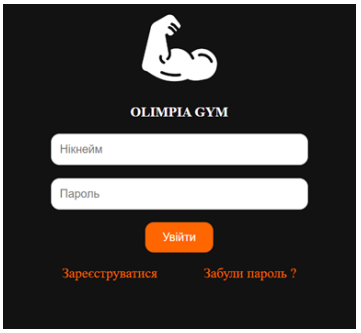


Рисунок 3.19 – Форма для авторизації на сайті

Сторінка реєстрації має уже більше обов’язкових полів для заповнення, а саме – нікнейм, пошта, пароль, повторне введення паролю. Після реєстрації користувач переадресовується на сторінку авторизації для повторного вводу даних і заходу на сторінку персонального кабінету (рис. 3.20).

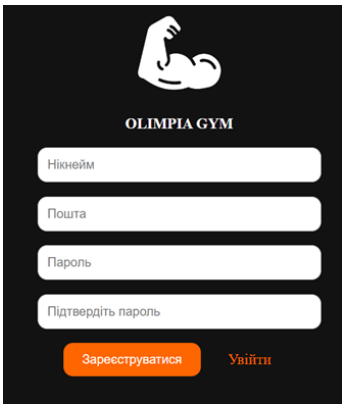


Рисунок 3.20 – Форма для реєстрації на сайті

Клас RegisterView наслідується від FormView, що надає функціональність для обробки форм. Змінна form_class = RegisterForm вказує, що в цьому представленні буде використовуватися форма RegisterForm. Змінна template_name вказує шлях до HTML-шаблону, який буде використано для рендерингу сторінки реєстрації, success_url = reverse_lazy("users:account") – визначає URL, на який буде перенаправлено користувача після успішної реєстрації:

```
    class RegisterView(FormView):
form_class = RegisterForm
template_name = 'registration/register.html'
success_url = reverse_lazy("users:account")
```

Метод form_valid перевизначає стандартний метод form_valid, form.save() зберігає дані форми. У випадку реєстрації, зберігає новий обліковий запис користувача:

```
    def form_valid(self, form):
form.save()
return super().form_valid(form)
```

Якщо користувач авторизований, хоче ввійти у свій особистий кабінет, але не пам’ятає паролю – він може його змінити, перейшовши за посиланням на формі авторизації. Йому пропонують ввести електронну пошту на яку відправиться посилання для заміни паролю (рис. 3.21).

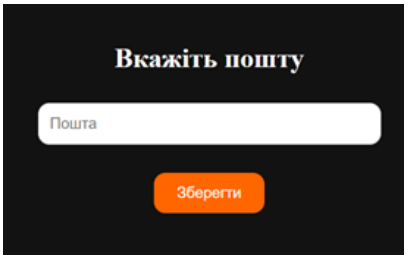


Рисунок 3.21 – Форма введення пошти для зміни паролю

Повідомлення на зміну паролю надсилається миттєво, тому при натисканні кнопки «Зберегти», воно уже буде лежати у вхідних повідомленнях. Відкривши лист, користувач може прочитати через що його турбують, скопіювати адресу посилання і перейти до зміни паролю (рис. 3.22).

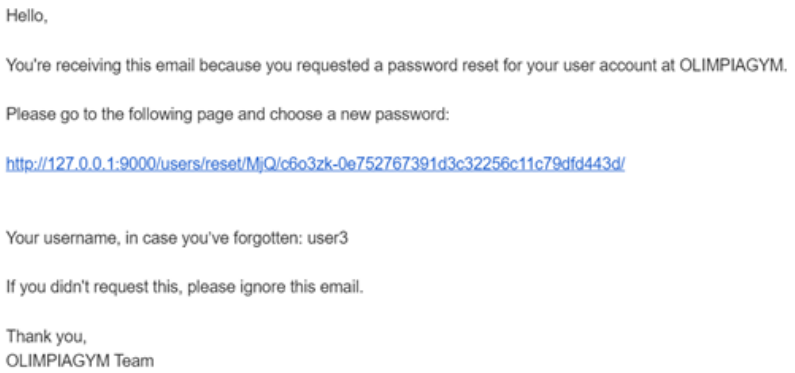


Рисунок 3.22 – Лист з посиланням на зміну паролю

Форма зміни паролю містить всього два поля – ввід нового паролю та повторення нового паролю, після чого користувача переадресовує на сторінку, де вказано, що все пройшло успішно і пропонують перейти на сторінку авторизації, де користувач зможе ввести новостворений пароль і увійти до свого персонального кабінету (рис. 3.23).

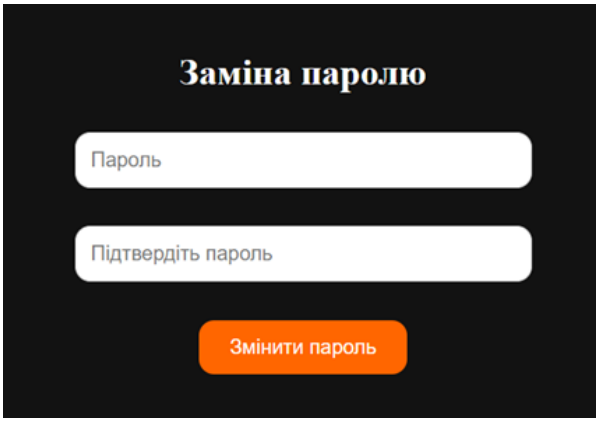


Рисунок 3.23 – Форма зміни паролю

Після введення нових даних – користувач може зайти на сторінку свого особистого кабінету, щоб створити чи переглянути свій план тренувань (рис. 3.24).

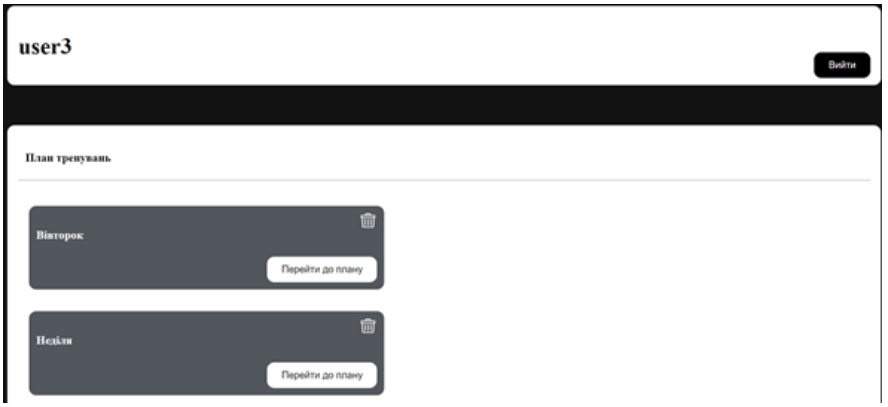


Рисунок 3.24 – Сторінка особистого кабінету користувача

3.2.2 Персональний план тренувань: список вправ

Якщо користувач ще не створив ніякий план тренувань, то при заході на його особистий кабінет – він побачить пусте поле знизу з кнопкою про створення плану тренувань (рис. 3.25). Якщо ж він уже мав створений, то спочатку відображатимуться створені плани, а потім кнопка (рис. 3.24).

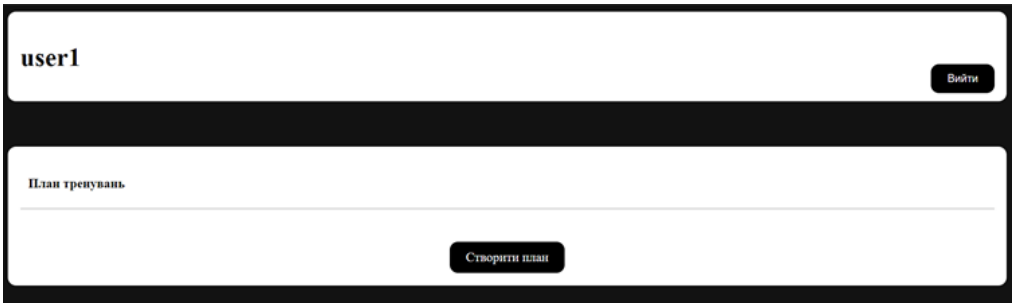


Рисунок 3.25 – Вигляд сторінки кабінету, де немає жодного плану

Натиснувши на кнопку, користувач попадає на сторінку з переліком доступних вправ з фільтром, по якому він зможе сортувати їх для швидкого і зручного формування плану (рис. 3.26).

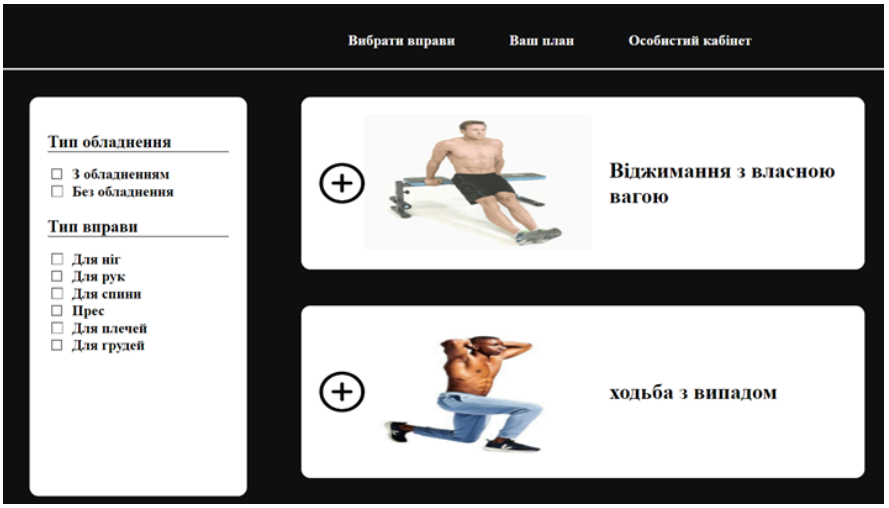


Рисунок 3.26 – Сторінка з переліком вправ

Додавання вправ на сайт здійснюється через адмін панель, де можна ввести назву вправи, додати супроводжувальне фото. Після чого зберегти. Збережені нововведення одразу відобразяться на сторінці з вправами, так як програма отримує всі дані з моделі Exercise і виводить вміст на сторінку:

```
def exercises(request):
 equipment_types = request.GET.getlist('equipment_type[]')
 exercise_types = request.GET.getlist('exercise_type[]')
```

Фільтрація здійснюється за допомогою filter. Якщо equipment_types не порожній (тобто, якщо користувач вибрав типи обладнання), то виконується фільтрація items за полем type2, яке повинно відповідати хоча б одному значенню зі списку equipment_types. Аналогічно, якщо exercise_types не порожній, то виконується додаткова фільтрація items за полем type, яке повинно відповідати хоча б одному значенню зі списку exercise_types. filter(type2__in=equipment_types) повертає новий QuerySet, який містить тільки ті об'єкти, у яких поле type2 співпадає з одним із значень у списку equipment_types. Filter(type__in=exercise_types) знову повертає новий QuerySet, що містить лише ті об'єкти, які відповідають умовам:

```
items = Exercise.objects.all()
 if equipment_types:
 items = items.filter(type2__in=equipment_types)
 if exercise_types:
 items = items.filter(type__in=exercise_types)
```

Також перевіряється, чи є запит асинхронним (AJAX-запитом). Це здійснюється шляхом перевірки заголовка X-Requested-With. Якщо запит є AJAX-запитом, то генерується HTML за допомогою функції render_to_string, яка рендерить шаблон users/includes/exercise_list.html з контекстом, що містить відфільтрований список вправ (items). Результат повертається у форматі JSON (JsonResponse), що містить HTML-код:

```
if request.headers.get('X-Requested-With') == 'XMLHttpRequest':
html = render_to_string(
'users/includes/exercise_list.html',
{'items': items}
)
return JsonResponse({'html': html})
```

При виборі відповідних значень у фільтрі, сторінка відображає зміни у режимі реального часу за допомогою скрипту, підключеного до фільтру, який реагує на зміни стану у checkbox. Кожного разу, коли змінюється стан будь-якого чекбокса (встановлений або знятий), виконується $('input[type="checkbox"]').change(function(){ ... }):

```
<script>
$(document).ready(function(){
$('input[type="checkbox"]').change(function(){
var                        equipmentTypes                        =
$('input[name="equipment_type"]:checked').map(function(){
return this.value;
}).get();
```

var equipmentTypes = $('input[name="equipment_type"]:checked').map(function(){ return this.value; }).get(); — ця частина коду знаходить всі чекбокси з іменем equipment_type, які є вибраними (:checked). Використовує метод map, щоб створити новий масив, в якому кожен елемент — значення відповідного чекбоксу. Метод get() перетворює результат на звичайний масив JavaScript. Подібним чином обробляються чекбокси з іменем exercise_type:

```
var exerciseTypes =
$('input[name="exercise_type"]:checked').map(function(){
return this.value;
}).get();
```

$.ajax({...}); — ця функція надсилає асинхронний запит до сервера. $('.exercise').html(response.html); — замінює вміст HTML-елементу з класом exercise на HTML-код, отриманий у відповіді від сервера (response.html):

```
$.ajax({
 url: '{% url "users:exercises" %}',
 type: 'GET',
 data: {
 'equipment_type': equipmentTypes,
 'exercise_type': exerciseTypes
 },
 success: function(response) {
 $('.exercise').html(response.html);
 }
 });
 });
 });
```

Якщо попробувати відсортувати список вправ по категорії «вправи для ніг», користувач отримає 2 результати, так як на сайті поки що присутні лише 2 вправи для ніг (рис 3.27).

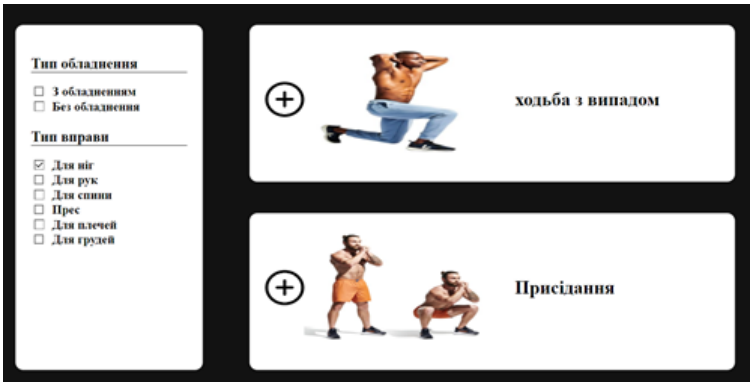


Рисунок 3.26 – Відсортований список вправ

Коли користувач натискає на плюсик у тілі вправи – вправа автоматично додається до його плану тренувань, який він може переглянути, натиснувши на відповідне значення у хедері сторінки (рис. 3.26). Додавання відбувається у режимі реального часу. Обробка такої події реалізована за допомогою скрипта, який реагує на зміну стану зображення всередині вправ з класом .icon:

```
$(document).on('click', '.icon img', function() {
 var exerciseId = $(this).data('exercise-id');
```

Після кліку на зображення виконується AJAX-запит, який відправляє ID вправи на сервер для додавання до плану користувача. Якщо додавання пройшло успішно – користувач побачить цю вправу на сторінці свого плану:

```
$.ajax({
url: '{% url 'users:add_exercise_to_plan' %}',
type: 'POST',
data: {
'exercise_id': exerciseId,
'csrfmiddlewaretoken': '{{ csrf_token }}'
},
success: function(response) {
alert(response.message);
},
error: function() {
```

```
alert('Error adding exercise.');
}
});
});
```

Саме ж додавання реалізоване по принципу отримання даних про id вправи. Метод first() повертає перший знайдений об'єкт або None, якщо об'єкт не знайдено:

```
@login_required
@require_POST
def add_exercise_to_plan(request):
exercise_id = request.POST.get('exercise_id')
if not exercise_id:
return JsonResponse({'status': 'error', 'message': 'Invalid exercise ID'}, status=400)
exercise = Exercise.objects.filter(id=exercise_id).first()
if not exercise:
return JsonResponse({'status': 'error', 'message': 'Exercise not found'}, status=404)
```

Якщо вправа не знайдена, повертається JsonResponse з помилкою та статусом 404 (Not Found). Якщо додавання пройшло успішно, то з’явиться відповідне повідомлення (рис. 3.28).

Рисунок 3.28 – Сповіщення про успішне додавання вправи у план

Також, в контексті додавання вправ у план, був реалізований метод додавання вправ спочатку у тимчасовий масив даних, щоб зміни у плані

одразу не вносилися у базу даних, а давали користувачу можливість спочатку все налаштувати. Аж після того, як користувач зберіг зміни, дані вказані у цьому тимчасовому плані передаються постійному, а в базі даних вносяться відповідні зміни:

```
temp_plan = request.session.get('temp_plan', [])
temp_plan.append({
'exercise_id': exercise.id,
'description': exercise.description,
'image_url': request.build_absolute_uri(exercise.image.url)
if exercise.image else None,
'icon_url':   request.build_absolute_uri(exercise.icon.url)   if
exercise.icon else None
})
request.session['temp_plan'] = temp_plan
return JsonResponse({'status': 'success', 'message': 'Exercise
added to your plan.'})
```

3.2.3 Персональний план тренувань: сторінка редагування плану

Додані вправи відобразяться на сторінці з планом тренувань користувача, де він зможе змінювати їхній порядок, встановлювати кількість підходів та дати назву плану, після чого зберегти його (рис. 3.29).

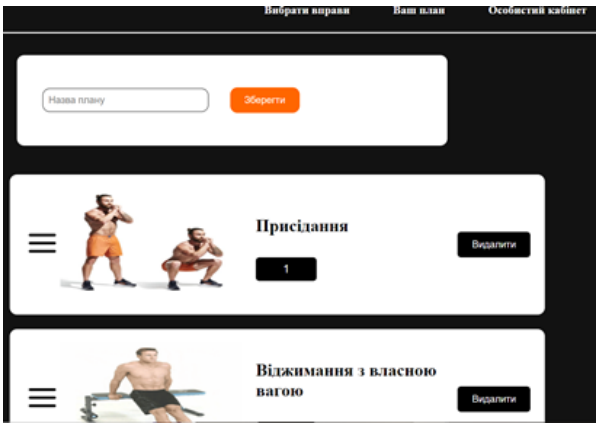


Рисунок 3.29 – Відображення доданих вправ у плані тренувань

Якщо користувач забажав видалити вправу з плану, то виконується спроба отримати об'єкт UserExercisePlan з бази даних за plan_item_id, plan_id та користувачем (request.user):

```
def remove_exercise_from_plan(request):
 plan_item_id = request.POST.get('plan_item_id')
 plan_id = request.POST.get('plan_id', None)
```

Якщо об'єкт знайдено, він видаляється за допомогою методу delete(). Повертається успішна відповідь у форматі JSON:

```
if plan_id:
 try:
 plan_item = UserExercisePlan.objects.get(id=plan_item_id,
 plan_id=plan_id, user=request.user)
 plan_item.delete()
 return JsonResponse({'status': 'success', 'message': 'Exercise
 removed from your plan.'})
```

Якщо об'єкт не знайдено (викидається виключення UserExercisePlan.DoesNotExist), повертається відповідь з помилкою та статусом 404. Також іде оновлення тимчасового масиву даних, де зберігалися вправи. По його закінченню, видалена вправа прибирається з тимчасового масиву:

```
except UserExercisePlan.DoesNotExist:
 return JsonResponse({'status': 'error', 'message': 'Exercise
 not found in the plan.'}, status=404)
 else:
 temp_plan = request.session.get('temp_plan', [])
 temp_plan = [item for item in temp_plan if
 str(item['exercise_id']) != plan_item_id]
 request.session['temp_plan'] = temp_plan
 return JsonResponse({'status': 'success', 'message': 'Exercise
 removed from temporary plan.'})
```

Після видалення на сторінці перестане відображатися видалена вправа. Решта – залишаться. Цей процес буде відбуватися в режимі реального часу за допомогою відповідного скрипта:

```
    $.ajax({
url: '{% url 'users:remove_exercise_from_plan' %}',
type: 'POST',
data: {
'plan_item_id': planItemId,
'plan_id': planId,
'csrfmiddlewaretoken': '{{ csrf_token }}'
},
success: function(response) {
alert(response.message);
location.reload();
},
error: function() {
alert('Error removing exercise.');
}
```

Коли користувач вводить відповідне числове значення, яке відповідає за кількість повторів для вправи, то програма одразу отримує ідентифікатор елемента плану вправ з POST-даних: plan_item_id = request.POST.get('plan_item_id'). Quantity = int(request.POST.get('quantity', 1)). Отримує кількість, яку користувач хоче встановити для елемента плану вправ. За замовчуванням, якщо дані не надані, кількість встановлюється на 1:

```
@login_required
 @require_POST
 def update_quantity(request):
 if request.headers.get('X-Requested-With') == 'XMLHttpRequest':
 plan_item_id = request.POST.get('plan_item_id')
 quantity = int(request.POST.get('quantity', 1))
 plan_id = request.POST.get('plan_id', None)
```

Якщо процес пройшов успішно, користувачу виведеться відповідне повідомлення на екран:

```
if plan_id:
 try:
 plan_item    =    UserExercisePlan.objects.get(id=plan_item_id,
 user=request.user)
 plan_item.quantity = quantity
 plan_item.save()
 return JsonResponse({'status': 'success', 'message': 'Quantity
 updated successfully.'})
```

Якщо ж по якійсь з причин не вдалося оновити числове значення, користувач отримає про це повідомлення на тому ж екрані. Також іде оновлення тимчасового масиву даних, де зберігалися вправи. По його закінченню, числове значення оновиться:

```
 except UserExercisePlan.DoesNotExist:
 return JsonResponse({'status': 'error', 'message': 'Plan item
 not found.'}, status=404)
 else:
 temp_plan = request.session.get('temp_plan', [])
 for item in temp_plan:
 if str(item['exercise_id']) == plan_item_id:
 item['quantity'] = quantity
 break
 request.session['temp_plan'] = temp_plan
 return JsonResponse({'status': 'success', 'message': 'Temporary
 plan quantity updated successfully.'})
 return  JsonResponse({'status':  'error',  'message':  'Invalid
 request'}, status=400)
```

Обробка даної дії здійснюється за скриптом, який реєструє обробник події для всіх елементів з класом .quantity-input, які знаходяться в документі. Обробник запускається при зміні значення в одному з цих елементів. Var

planItemId = $(this).data('plan-item-id'). З цього рядка отримується значення атрибута data-plan-item-id з елементу, який спричинив подію зміни. Це ідентифікує конкретний елемент плану. Тут отримується значення введеного користувачем числа в полі введення .quantity-input: var quantity = $(this).val():

```
    $(document).on('change', '.quantity-input', function() {
var planItemId = $(this).data('plan-item-id');
var quantity = $(this).val();
var planId = $(this).data('plan-id');
```

Тіло Ajax-запиту, відповідального за зміну числового значення, виглядає наступним чином:

```
    $.ajax({
url: '{% url 'users:update_quantity' %}',
type: 'POST',
data: {
'plan_item_id': planItemId,
'quantity': quantity,
'plan_id': planId,
'csrfmiddlewaretoken': '{{ csrf_token }}'
},
success: function(response) {
alert('Quantity updated successfully.');
},
error: function() {
alert('Error updating quantity.');
}
```

Зміна порядку відбувається, коли функція def update_exercise_order(request) обробляє запит на оновлення порядку вправ. Приймає об'єкт запиту request. Item_ids = [int(id.strip()) for id in order.split(',')] розбиває рядок 'order' на список ідентифікаторів елементів, перетворює кожен

ідентифікатор у ціле число та видаляє пробіли. Ітерація по вправам відбувається за допомогою функції for index, item_id in enumerate(item_ids):

```
@login_required
@require_POST
def update_exercise_order(request):
order = request.POST.get('order')
if not order:
return JsonResponse({'status': 'error', 'message': 'No order provided'}, status=400)
try:
item_ids = [int(id.strip()) for id in order.split(',')]
```

plan_item = UserExercisePlan.objects.get(id=item_id) – отримує об'єкт UserExercisePlan за заданим ідентифікатором. Plan_item.position = index: Присвоює полю position значення індексу у списку. Plan_item.save() зберігає зміни в базі даних. Після оновлення всіх позицій повертає відповідь у форматі JSON зі статусом успіху та повідомленням:

```
plan_item = UserExercisePlan.objects.get(id=item_id)
plan_item.position = index
plan_item.save()
return JsonResponse({'status': 'success', 'message': 'Order updated successfully.'})
except Exception as e:
return JsonResponse({'status': 'error', 'message': f'Error updating order: {str(e)}'}, status=500)
```

Процес зміни положення реалізований з використанням метода .sortable() з бібліотеки jQuery UI. Додається обробник події update, який виконується після завершення перетягування. За допомогою методу .sortable('toArray', {attribute: 'id'}) отримуємо масив ID елементів у потрібному порядку. Метод .map() використовується для видалення префіксу 'item-' з

кожного ID, щоб отримати порядковий номер. Після чого результат перетворюється в рядок за допомогою .toString():

```
$(document).ready(function() {
$(".sortable").sortable({
update: function(event, ui) {
var order = $(this).sortable('toArray', {attribute:
'id'}).map(function(item) {
return item.replace('item-', '');
}).toString();
```

Var csrfToken = $('#csrfToken').val() відповідає за відправлення даних на сервер за допомогою AJAX-запиту. Використовується метод $.ajax(). В тілі запиту передається порядок елементів:

```
var csrfToken = $('#csrfToken').val();
console.log("Order sent:", order);
console.log("CSRF Token:", csrfToken);
$.ajax({
url: '{% url 'users:update_exercise_order' %}',
type: 'POST',
headers: {
'X-CSRFToken': csrfToken
},
data: {
'order': order
},
success: function(response) {
console.log('Order updated');
},
error: function(xhr, status, error) {
console.log('Error updating order:', xhr.responseText);
}
```

Після всіх налаштувань користувач може ввести назву плану тренувань і натиснути кнопку «Зберегти» після чого план з’явиться на сторінці його особистого кабінету (рис. 3.30).

Рисунок 3.29 – Відображення створеного плану

Якщо користувач забажає видалити план, то його варто всього лише натиснути на іконку мусорника на плані і план видалиться. Під час кліку на зображення, визначається planId через пошук найближчого батьківського елемента з класом .plan і отримання значення атрибута data-plan-id:

```
$(document).ready(function() {
$('.delete-p img').click(function() {
var planId = $(this).closest('.plan').data('plan-id');
```

Рядок If (confirm('Are you sure you want to delete this plan?')) { ... } відповідає за виклик діалогового вікна для підтвердження видалення плану. Якщо користувач натискає "OK", код продовжує виконання. Після видалення з бази даних зникне запис, що користувач має такий план:

```
var planId = $(this).closest('.plan').data('plan-id');
 if (confirm('Are you sure you want to delete this plan?')) {
 $.ajax({
 url: '{% url 'users:delete_plan' %}',
 type: 'POST',
 data: {
```

```
'plan_id': planId,
'csrfmiddlewaretoken': '{{ csrf_token }}'
},
success: function(response) {
if (response.status == 'success') {
location.reload();
} else {
alert(response.message);
}
},
error: function() {
alert('Error deleting plan.');
}
});
}
});
});
```

Висновки до розділу 3

У цьому розділі ми детально розглянули процес створення та розробки веб-сайту для спортзалу з інтеграцією платіжної системи Stripe та особистим кабінетом для користувачів. За допомогою інтеграції Stripe ми забезпечили зручну та безпечну оплату послуг прямо на сайті, що сприяє підвищенню зручності та доступності для клієнтів спортзалу. Особистий кабінет користувача дозволяє зберігати особисті дані, переглядати історію оплат та тренувань, а також складати індивідуальні плани тренувань згідно їхніх потреб та цілей.

Крім того, варто зазначити, що розроблений веб-сайт має мобільну версію, що робить його доступним для користувачів на будь-яких пристроях, що підвищує його зручність та доступність.

ВИСНОВОК

Розроблена онлайн-платформа для спортзалу з особистим кабінетом користувача є актуальним та значимим внеском у розвиток фітнес-індустрії в сучасному цифровому світі. Популярність фітнесу серед населення постійно зростає, що створює потребу у зручних та ефективних інструментах управління тренуваннями. Розроблена платформа дозволяє клієнтам спортзалу легко створювати та керувати своїми тренувальними планами через веб-платформу або мобільний додаток.

Використання мови програмування Python, бази даних MySQL та фреймворку Django дозволило успішно реалізувати функціонал платформи. Проведення аналізу існуючих аналогів, вибір оптимальних технологій розробки, розробка зручного дизайну та тестування продукту сприяли створенню продукту, який відповідає потребам користувачів.

Отже, результатом роботи є функціональна та ефективна онлайн-платформа, яка забезпечує зручність, персоналізацію та контроль над тренувальним процесом для клієнтів спортзалу, сприяючи підвищенню їхнього задоволення від фітнесу та конкурентоспроможності спортзалу на ринку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. PULSE GYM: веб-сайт. URL: https://pulse-gym.com.ua (дата звернення: 27.03.2024)
2. SportLife: веб-сайт. URL: https://sportlife.ua/uk/ (дата звернення 27.03.2024)
3. Альянс: веб-сайт. URL: https://alliance.if.ua (дата звернення 27.03.2024)
4. Python: веб-сайт. URL: https://www.python.org (дата звернення 27.03.2024)
5. Django: веб-сайт. URL: https://www.djangoproject.com (дата звернення 27.03.2024)
6. SQLite: веб-сайт. URL: https://www.djangoproject.com (дата звернення 27.03.2024)
7. VS Code: веб-сайт. URL: https://code.visualstudio.com (дата звернення 27.03.2024)
8. Git: веб-сайт. URL: https://git-scm.com (дата звернення 04.03.2024)
9. JetBrains: веб-сайт URL: https://www.jetbrains.com (дата звернення 27.03.2024)
10. Diagrams.net: веб-сайт URL: https://app.diagrams.net (дата звернення 23.04.2024)
11. Figma: веб-сайт URL: https://www.figma.com (дата звернення 23.04.2024)
12. Ckeditor: веб-сайт URL: https://ckeditor.com (дата звернення 23.04.2024)
13. Stripe: веб-сайт URL: https://stripe.com (дата звернення 23.04.2024)
14. Мій GitHub: веб-сайт URL: https://github.com/VolodymyrPysklynets/gym_site.git (дата звернення 20.05.2024)
15. Intelig Idea: веб-сайт URL: https://www.jetbrains.com/idea/ (дата звернення 23.04.2024)

16. PyCharm: веб-сайт URL: https://www.jetbrains.com/pycharm/ (дата звернення 23.04.2024)

17. HTML: веб-сайт URL: https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML (дата звернення 23.04.2024)

18. CSS: веб-сайт URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення 23.04.2024)

19. JavaScript веб-сайт URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення 23.04.2024)

20. C# веб-сайт URL: https://learn.microsoft.com/uk-ua/dotnet/csharp/ (дата звернення 23.04.2024)

ДОДАТКИ

Додаток А

Код аплікейшина gymsite

```
{% extends "gymsite/base.html" %}
{% load static %}
{% block content %}
<html>
    <head>
        <title>OLIMPIAGYM</title>
          <link rel="stylesheet" href="{% static 'gymsite/style.css'
%}">
           <link rel="stylesheet" href="{% static 'gymsite/base.css'
%}">
    </head>
    <body>
        <div id="banner">
                  <img width="100%" height="450rem" src="{% static
"gymsite/images/banner.png" %}" alt="Banner">
        </div>
        <div class="box">
            <div class="g_container">
                <div class="gym_zal">
                    <a href="/mainpage/aboutgym">
                                                <img src="{% static
"gymsite/images/gym_zal.jpg" %}" alt="gym_zal">
                                              <span><strong>Тренарежний
зал</strong></span>
                    </a>
                </div>
                <div class="group_workout">
                    <a href="/mainpage/grouptrain">
                                                <img src="{% static
"gymsite/images/group_train.jpg" %}" alt="group_workout">
```

```
                                                    <span><strong>Групові
тренування</strong></span>
                        </a>
                    </div>
                    <div class="personal_workout">
                        <a href="/mainpage/personaltrain">
                                                    <img  src="{%  static
"gymsite/images/personal_train.jpg" %}" alt="personal_workout">
                                                <span><strong>Персональні
тренування</strong></span>
                        </a>
                    </div>
                    <div class="gym_actions">
                        <a href="/mainpage/activities">
                                                    <img  src="{%  static
"gymsite/images/activities.jpg" %}" alt="gym_actions">
                                                <span><strong>Широкий  вибір
активностей</strong></span>
                        </a>
                    </div>
                </div>
            </div>
            <div id="table" class="table_component">
                <table>
                    <caption><h1>РОЗКЛАД</h1></caption>
                    <thead>
                        <tr>
                            <th><h3>ПН</h3></th>
                            <th><h3>ВТ</h3></th>
                            <th><h3>СР</h3></th>
                            <th><h3>ЧТ</h3></th>
                            <th><h3>ПТ</h3></th>
                        </tr>
                    </thead>
                    <tbody>
                        <tr>
                            <td><h3>Бокс</h3><br><h4>13:00</h4></td>
                            <td><h3>Фітнес</h3><br><h4>12:30</h4></td>
```

```
                                                    <td><h3>Дитяча
Аеробіка</h3><br><h4>14:00</h4></td>
                            <td><h3>Фітнес</h3><br><h4>13:00</h4></td>
                            <td><h3>Бокс</h3><br><h4>14:50</h4></td>
                        </tr>
                        <tr>
                                                    <td><h3>Дитяча
Аеробіка</h3><br><h4>15:30</h4></td>

<td><h3>Стретчинг</h3><br><h4>16:00</h4></td>

<td><h3>Стретчинг</h3><br><h4>16:05</h4></td>
                            <td><h3>Бокс</h3><br><h4>16:00</h4></td>
                                                    <td><h3>Дитяча
Аеробіка</h3><br><h4>15:30</h4></td>
                        </tr>
                        <tr>
                            <td></td>
                            <td></td>
                            <td><h3>Фітнес</h3><br><h4>18:10</h4></td>
                            <td></td>

<td><h3>Стретчинг</h3><br><h4>17:00</h4></td>
                        </tr>
                    </tbody>
                </table>
            </div>

            <div class="discount">
                <div class="dis_img">
                       <img src="{% static "gymsite/images/discount.jpg"
%}" alt="discount">
                </div>
                <div class="dis_text">
                       <h2>Даруємо <span>знижку</span> на перший місяць
занять!</h2>
                    <ul>
                        <li>Відмічайте нас у своїх сторіс</li>
```

```
                                    <li>Очікуйте повідомлення в direct з
промокодом</li>
                    </ul>
                </div>
            </div>
            <div class="coach">
                <div class="c_text">
                    <h4>
                        Наші висококваліфіковані тренери відзначаються
глибоким<br>
                            розумінням індивідуальних потреб кожного
клієнта,<br>
                        надаючи персоналізовані програми тренувань для
досягнення<br>
                    максимальних результатів.
                    </h4>
                    <h4>
                        Завдяки їхньому професіоналізму<br>
                            та підтримці, ви зможете ефективно досягти
своїх<br>
                        спортивних цілей в нашому спортзалі.
                    </h4>
                                    <a  href="/mainpage/trainers"><button
class="button">Тренери</button></a>
                </div>
                <div class="c_img">
                      <img src="{% static "gymsite/images/coach.jpg" %}"
alt="coach">
                </div>
            </div>
            <div id="slider">
                <input type="radio" name="slider" id="slide1" checked>
                <input type="radio" name="slider" id="slide2">
                <input type="radio" name="slider" id="slide3">
                <input type="radio" name="slider" id="slide4">
                <div id="slides">
                   <div id="overflow">
                      <div class="inner">
                         <div class="slide slide_1">
```

```
                        <div class="slide-content">
                            <img width="50px" height="auto" src="{%
static "gymsite/images/quote.png" %}" alt="quote">
                            <p><em>Просто залп енергії!<br>
                            Відмінне обладнання, зручне розташування
і дружня атмосфера.<br>
                            Персонал завжди готовий допомогти.<br>
                            Я відчуваю себе справжнім чемпіоном,
коли тренуюсь тут!</em></p>
                        </div>
                    </div>
                    <div class="slide slide_2">
                        <div class="slide-content">
                            <img width="50px" height="auto" src="{%
static "gymsite/images/quote.png" %}" alt="quote">
                            <p><em>Люблю цей спортзал за його
різноманітність.<br>
                            Тут є все, що потрібно для різних видів
тренувань - від кардіо до силових вправ.<br>
                            Кожного разу знаходжу нові можливості
для покращення своєї фізичної форми!</em></p>
                        </div>
                    </div>
                    <div class="slide slide_3">
                        <div class="slide-content">
                            <img width="50px" height="auto" src="{%
static "gymsite/images/quote.png" %}" alt="quote">
                            <p><em>Найкраще місце для зарядки
енергії!<br>
                            Дивовижна команда і чистий, затишний
простір.<br>
                            Тут завжди приємно проводити час і
працювати над собою.<br>
                            Дякую, що створюєте таке чудове
середовище для тренувань</em></p>
                        </div>
                    </div>
                    <div class="slide slide_4">
                        <div class="slide-content">
```

```
                                    <img width="50px" height="auto" src="{%
static "gymsite/images/quote.png" %}" alt="quote">
                                <p><em>Спортзал - мій другий дім!<br>
                                    Чудова атмосфера, професійні тренери і
безліч можливостей для розвитку.<br>
                                Щоразу, приходячи сюди, відчуваю приплив
енергії та мотивації<br>
                                    досягати нових висот у спорті і житті
взагалі!</em></p>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
            <div id="controls">
                <label for="slide1"></label>
                <label for="slide2"></label>
                <label for="slide3"></label>
                <label for="slide4"></label>
            </div>
            <div id="bullets">
                <label for="slide1"></label>
                <label for="slide2"></label>
                <label for="slide3"></label>
                <label for="slide4"></label>
            </div>
        </div>
    </body>
</html>
{% endblock %}

{% extends "gymsite/base.html" %}
{% load static %}
{% block content %}
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title>OLIMPIAGYM</title>
```

```
            <meta name="description" content="">
                    <meta  name="viewport"  content="width=device-width,
initial-scale=1">
                               <link   rel="stylesheet"   href="{%   static
'gymsite/style_coach.css' %}">
                <link rel="stylesheet" href="{% static 'gymsite/base.css'
%}">
        </head>
        <body>
            <div class="strelka">
                <div class="coach">
                        <img width="200px" height="250" src="{% static
"gymsite/images/trainer1.jpg" %}" alt="trainer1">
                    <div class="info">
                     <div class="main-info">
                         <h1>Гавриленко Аліна</h1>
                         <h4>тренер 1-ї категорії</h4>
                     </div>
                     <div class="other-info">
                         <h2><strong>Досвід роботи:</strong> 1 рік</h2>
                                    <h2><strong>Кількість  проведених
занять:</strong> 75+</h2>
                     </div>
                        <a href="/trainers/gavrylenko"><button>Дізнатися
більше</button></a>
                    </div>
                 </div>
            </div>
            <div class="strelka">
                <div class="coach">
                        <img width="200px" height="250" src="{% static
"gymsite/images/trainer2.jpg" %}" alt="trainer2">
                    <div class="info">
                     <div class="main-info">
                         <h1>Мисан Богдан</h1>
                         <h4>тренер 3-ї категорії</h4>
                     </div>
                     <div class="other-info">
                         <h2><strong>Досвід роботи:</strong> 4 роки</h2>
```

```
                                        <h2><strong>Кількість проведених
занять:</strong> 550+</h2>
                    </div>

                            <a href="/trainers/mysan"><button>Дізнатися
більше</button></a>
                </div>
            </div>
        </div>
        <div class="strelka">
            <div class="coach">
                    <img width="200px" height="250" src="{% static
"gymsite/images/trainer3.jpg" %}" alt="trainer3">
                <div class="info">
                 <div class="main-info">
                    <h1>Ренівський Валентин</h1>
                    <h4>тренер 4-ї категорії</h4>
                 </div>
                 <div class="other-info">
                            <h2><strong>Досвід роботи:</strong> 6
років</h2>
                                    <h2><strong>Кількість проведених
занять:</strong> 1150+</h2>
                 </div>
                    <a href="/trainers/renivskiy"><button>Дізнатися
більше</button></a>
                </div>
            </div>
        </div>
        <div class="strelka">
            <div class="coach">
                    <img width="200px" height="250" src="{% static
"gymsite/images/trainer4.jpg" %}" alt="trainer4">
                <div class="info">
                 <div class="main-info">
                    <h1>Шашкевич Наталія</h1>
                    <h4>тренер 2-ї категорії</h4>
                 </div>
                 <div class="other-info">
```

```
                        <h2><strong>Досвід роботи:</strong> 3 роки</h2>
                                    <h2><strong>Кількість проведених
занять:</strong> 250+</h2>
                    </div>
                      <a href="/trainers/shashkevych"><button>Дізнатися
більше</button></a>
                  </div>
                </div>
            </div>
        </body>
    </html>
    {% endblock %}

    {% extends "gymsite/base.html" %}
    {% load static %}
    {% block content %}
    <html>
        <head>
            <meta charset="utf-8">
            <meta http-equiv="X-UA-Compatible" content="IE=edge">
            <title>OLIMPIAGYM</title>
            <meta name="description" content="">
                    <meta  name="viewport"  content="width=device-width,
initial-scale=1">
              <link rel="stylesheet" href="{% static 'gymsite/base.css'
%}">
                          <link   rel="stylesheet"   href="{%   static
'gymsite/articles_abonem.css' %}">
        </head>
        <body>
            <div class="artcl">
                {% for item in items %}
                    <a href="article/{{item.id}}/" class="article-link">
                        <div class="back">
                            <div class="content-wrapper">
                                <div class="image">
                                        <img src="{{item.image.url}}"
width="250px" height="150px">
                                </div>
```

```
                                <div class="text">
                                    <h2>{{item.description}}</h2>
                                </div>
                            </div>
                        </div>
                    </a>
                {% endfor %}
            </div>
        </body>
    </html>
    {% endblock %}

    {% extends "gymsite/base.html" %}
    {% load static %}
    {% block content %}
    <html>
        <head>
            <meta charset="utf-8">
            <meta http-equiv="X-UA-Compatible" content="IE=edge">
            <title>OLIMPIAGYM</title>
            <meta name="description" content="">
                    <meta  name="viewport"  content="width=device-width,
initial-scale=1">
              <link rel="stylesheet" href="{% static 'gymsite/base.css'
%}">
                            <link   rel="stylesheet"   href="{%   static
'gymsite/articles_abonem.css' %}">
        </head>
        <body>
            <div class="article-container">
                <div class="detail">
                    <h1>{{item.description}}</h1>
                            <img  src="{{item.image.url}}"  width="350px"
height="200px">
                    <p>{{item.text | safe}}</p>
                </div>
            </div>
        </body>
    </html>
```

```
{% endblock %}

{% extends "gymsite/base.html" %}
{% load static %}
{% block content %}
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title>OLIMPIAGYM</title>
        <meta name="description" content="">
                <meta  name="viewport"  content="width=device-width,
initial-scale=1">
            <link rel="stylesheet" href="{% static 'gymsite/base.css'
%}">
                            <link   rel="stylesheet"   href="{%   static
'gymsite/articles_abonem.css' %}">
    </head>
    <body>
        <div class="abon">
            <h1>Абонементи на заняття</h1>
            <div class="exercise">
                <div class="grid-container">
                    {% for item in exercise_abonements %}
                    <div class="grid-item">
                        <div class="ab-head">
                            <div class="name">
                                <h3>{{item.name}}</h3>
                            </div>
                            <div class="price">
                                <h3>{{item.price}}</h3>
                            </div>
                        </div>
                        <div class="info">
                            <p>{{item.description |safe}}</p>
                        </div>
                                                <form  action="{%  url
'create-checkout-session' item.id %}" method="POST">
                                {% csrf_token %}
```

```
                                <button type="submit">Придбати</button>
                            </form>
                        </div>
                        {% endfor %}
                    </div>
                </div>
                <div class="abon">
                    <h1>Місячні абонементи</h1>
                    <div class="exercise">
                        <div class="grid-container">
                            {% for item in monthly_abonements %}
                            <div class="grid-item">
                                <div class="ab-head">
                                    <div class="name">
                                        <h3>{{item.name}}</h3>
                                    </div>
                                    <div class="price">
                                        <h3>{{item.price}}</h3>
                                    </div>
                                </div>
                                <div class="info">
                                    <p>{{item.description |safe}}</p>
                                </div>
                                                    <form  action="{%  url
'create-checkout-session' item.id %}" method="POST">
                                    {% csrf_token %}
                                    <button type="submit">Придбати</button>
                                </form>
                            </div>
                            {% endfor %}
                        </div>
                    </div>
                </div>
            </body>
        </html>
        {% endblock %}

        def index(request):
```

```
        return render(request, "gymsite/mainpage.html")
    def base(request):
        return render(request, "gemsite/base.html")

    def grouptrain(request):
        return render(request, "gymsite/group_train.html")
    def personaltrain(request):
        return render(request, "gymsite/personal_train.html")
    def activities(request):
        return render(request, "gymsite/activities.html")
    def aboutgym(request):
        return render(request, "gymsite/about_gym.html")
    def trainers(request):
        return render(request, "gymsite/trainers.html")
    def gavrylenko(request):
        return render(request, "gymsite/gavrylenko.html")
    def mysan(request):
        return render(request, "gymsite/mysan.html")
    def renivskiy(request):
        return render(request, "gymsite/renivskiy.html")
    def shashkevych(request):
        return render(request, "gymsite/shashkevych.html")
    def create_checkout_session(request, abonement_id):
        abonement = Abonement.objects.get(id=abonement_id)
        try:
                    print("Creating Stripe session with metadata:",
{'abonement_id': str(abonement_id)})
            checkout_session = stripe.checkout.Session.create(
                payment_method_types=['card'],
                line_items=[
                    {
                        'price_data': {
                            'currency': 'uah',
                            'product_data': {
                                'name': abonement.name,
                            },
                            'unit_amount': abonement.price * 100,
                        },
                        'quantity': 1,
```

```
                    },
                ],
                mode='payment',
                customer_email=request.POST.get('email'),
                metadata={'abonement_id': str(abonement_id)},

success_url=request.build_absolute_uri(reverse('success'))              +
'?session_id={CHECKOUT_SESSION_ID}',

cancel_url=request.build_absolute_uri(reverse('cancel')),
            )
            return redirect(checkout_session.url, code=303)
        except Exception as e:
            return HttpResponse(str(e), status=500)

    urlpatterns = [
        path('', index, name="index"),
        path('grouptrain', grouptrain),
        path('personaltrain', personaltrain),
        path('activities', activities),
        path('aboutgym', aboutgym),
        path('trainers', trainers),
        path('gavrylenko', gavrylenko),
        path('mysan', mysan),
        path('renivskiy', renivskiy),
        path('shashkevych', shashkevych),
        path('article', article),
        path('abonement', abonement, name='abonement'),
                    path('create-checkout-session/<int:abonement_id>/',
create_checkout_session, name='create-checkout-session'),
        path('success/', success, name='success'),
        path('cancel/', cancel, name='cancel'),
        path('webhook/', stripe_webhook, name='stripe-webhook'),
    ]

    from django.contrib import admin
    from .models import Articles, Abonement
    admin.site.register(Articles)
    admin.site.register(Abonement)
```

Додаток Б

Код аплікейшина users

```
     {% load static %}
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title>OLIMPIAGYM</title>
        <meta name="description" content="">
                    <meta   name="viewport"   content="width=device-width,
initial-scale=1">
        <link rel="stylesheet" href="{% static 'users/style.css' %}">
    </head>
    <body>
        <div class="logo-container">
            <a href="/mainpage">
                        <img  width="100px"  height="auto"  src="{%  static
'gymsite/images/logo.png' %}" alt="logo"/>
                <p><strong>OLIMPIA GYM</strong></p>
            </a>
        </div>
        <form id="loginForm" method="post">
            {% csrf_token %}
            <div>
                 <input type="text" name="username" placeholder="Нікнейм"
required>
            </div>
            <div>
                                <input  type="password"    name="password"
placeholder="Пароль" required>
            </div>
            {{form.errors}}
            <div class="button-container">
                <button type="submit">Увійти</button>
            </div>
            <div class="link-container">
```

```
                                <a  href="{%  url  'users:register'  %}"
class="login">Зареєструватися</a>
                            <a  href="{%  url  'users:password_reset'  %}"
class="login">Забули пароль ?</a>
            </div>
        </form>
    </body>
</html>


     {% load static %}
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title>OLIMPIAGYM</title>
        <meta name="description" content="">
                    <meta   name="viewport"   content="width=device-width,
initial-scale=1">
        <link rel="stylesheet" href="{% static 'users/style.css' %}">
    </head>
    <body>
        <script async defer>
            window.onload = function() {
                document.getElementById("registrationForm").reset();
            };
        </script>
        <div class="logo-container">
            <a href="/mainpage">
                        <img  width="100px"  height="auto"  src="{%  static
'gymsite/images/logo.png' %}" alt="logo"/>
                <p><strong>OLIMPIA GYM</strong></p>
            </a>
        </div>
        <form id="registrationForm" method="post">
            {% csrf_token %}
            <div>
                 <input type="text" name="username" placeholder="Нікнейм"
required>
            </div>
```

```
            <div>
                    <input type="email" name="email" placeholder="Пошта"
required>
            </div>
            <div>
                                <input  type="password"  name="password1"
placeholder="Пароль" required>
            </div>
            <div>
                                <input  type="password"  name="password2"
placeholder="Підтвердіть пароль" required>
            </div>
            {{form.errors}}
            <button type="submit">Зареєструватися</button>
            <a href="{% url 'login' %}" class="login">Увійти</a>
        </form>
    </body>
</html>

     {% load static %}
<link rel="stylesheet" href="{% static 'users/style.css' %}">
<div class="center">
    <h2>Вкажіть пошту</h2>
</div>
<form method="post">
    {% csrf_token %}
    <div class="email">
        <input type="email" name="email" placeholder="Пошта" required>
    </div>
    <div class="button-container">
        <button type="submit">Зберегти</button>
    </div>
</form>

     {% load static %}
<link rel="stylesheet" href="{% static 'users/style.css' %}">


{% if validlink %}
```

```
    <div class="center">
        <h2>Заміна паролю</h2>
    </div>
    <form method="post">
        {% csrf_token %}
        <div class="email">
                            <input  type="password"  name="new_password1"
placeholder="Пароль" required>
        </div>
        <div class="email">
                           <input  type="password"  name="new_password2"
placeholder="Підтвердіть пароль" required>
        </div>
        <div class="button-container">
            <button type="submit">Змінити пароль</button>
        </div>
    </form>
{% else %}
    <div class="center">
          <p>Посилання для зміни пароля було недійсним, можливо, тому, що
воно вже було використано.
            Будь ласка, надішліть запит на скидання нового пароля.
        </p>
    </div>
{% endif %}

     {% load static %}
<link rel="stylesheet" href="{% static 'users/style.css' %}">
<div class="center">
    <p>
            Ваш пароль був успішно змінений. Ви можете <a href="{% url
'login' %}">Ввійти</a> прямо зараз.
    </p>
</div>

     {% load static %}
<html>
    <head>
        <meta charset="utf-8">
```

```
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title>OLIMPIAGYM</title>
        <meta name="description" content="">
                    <meta   name="viewport"   content="width=device-width,
initial-scale=1">
        <link rel="stylesheet" href="{% static 'users/style.css' %}">
                                                                    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></
script>
    </head>
    <body>
       <header>
        <h1>{{user.username}}</h1>
                <form  action="{%  url  'logout'  %}?next=/users/account"
method="post">
            {% csrf_token %}
            <button type="submit" class="logout">Вийти</button>
        </form>
       </header>
       <div class="main">
            <div class="head">
                <nav>
                    <a href="/users/account"><h4>План тренувань</h4></a>
                </nav>
            </div>
            <div class="create-p">
                {% for plan in user_plans %}
                    <div class="plan" data-plan-id="{{ plan.plan_id }}">
                        <div class="plan-content">
                            <h4>{{ plan.plan_name }}</h4>
                        </div>
                        <div class="plan-button">
                                    <button class="but"><a href="{% url
'users:edit_plan'    %}?plan_id={{    plan.plan_id    }}">Перейти    до
плану</a></button>
                        </div>
                        <div class="delete-p">
                            <img src="{% static 'users/images/delete.png'
%}" width="25px" height="20px">
```

```
                        </div>
                    </div>
                {% endfor %}
                                                <a   href="/users/exercises"
class="create-plan-btn">Створити план</a>
            </div>
        </div>
  </body>
</html>


     {% load static %}
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title>OLIMPIAGYM</title>
        <meta name="description" content="">
                    <meta   name="viewport"   content="width=device-width,
initial-scale=1">
        <link rel="stylesheet" href="{% static 'users/style.css' %}">
                                                                    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></
script>
    </head>
    <body>
<div class="exercise-header">
            <nav>
                <a href="/users/exercises"><h4>Вибрати вправи</h4></a>
                <a href="/users/edit_plan"><h4>Ваш план</h4></a>
                <a href="/users/account"><h4>Особистий кабінет</h4></a>
            </nav>
        </div>
        <div class="container">
            <div class="filter">
                <h3>Тип обладнення</h3>
                            <label><input  type="checkbox"  id="equipment"
name="equipment_type"         value="з         обладненням">         <h4>З
обладненням</h4></label>
```

```
                        <label><input type="checkbox" id="no_equipment"
name="equipment_type"        value="без        обладнення">        <h4>Без
обладнення</h4></label>
                <h3>Тип вправи</h3>
                                <label><input  type="checkbox"  id="legs"
name="exercise_type" value="ноги"> <h4>Для ніг</h4></label>
                                <label><input  type="checkbox"  id="arms"
name="exercise_type" value="руки"> <h4>Для рук</h4></label>
                                <label><input  type="checkbox"  id="back"
name="exercise_type" value="спина"> <h4>Для спини</h4></label>
                                 <label><input  type="checkbox"  id="abs"
name="exercise_type" value="прес"> <h4>Прес</h4></label>
                            <label><input  type="checkbox"  id="shoulders"
name="exercise_type" value="плечі"> <h4>Для плечей</h4></label>
                               <label><input  type="checkbox"  id="chest"
name="exercise_type" value="груди"> <h4>Для грудей</h4></label>
            </div>
            <div class="exercise">
                {% for item in items %}
                <div class="back">
                    <div class="content-wrapper">
                        <div class="icon">
                                <img src="{{item.icon.url}}" width="50px"
height="45px" data-exercise-id="{{ item.id }}">
                        </div>
                        <div class="image">
                              <img src="{{item.image.url}}" width="250px"
height="150px">
                        </div>
                        <div class="text">
                            <h2>{{item.description}}</h2>
                        </div>
                    </div>
                </div>
                {% endfor %}
            </div>
        </div>
    </body>
</html>
```

```
     {% load static %}
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title>OLIMPIAGYM</title>
        <meta name="description" content="">
                    <meta   name="viewport"   content="width=device-width,
initial-scale=1">
        <link rel="stylesheet" href="{% static 'users/style.css' %}">
                                                <link      rel="stylesheet"
href="https://code.jquery.com/ui/1.12.1/themes/base/jquery-ui.css">
                                                                    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></
script>
                                                                    <script
src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>
                                                                    <script
src="https://cdnjs.cloudflare.com/ajax/libs/jqueryui-touch-punch/0.2.3/jq
uery.ui.touch-punch.min.js"></script>
    </head>
    <body>
<div class="exercise-header">
            <nav>
                <a href="/users/exercises"><h4>Вибрати вправи</h4></a>
                            <a  href="/users/edit_plan_temporary"><h4>Ваш
план</h4></a>
                <a href="/users/account"><h4>Особистий кабінет</h4></a>
            </nav>
        </div>
        <div class="exercise">
            <div class="data">
                <form method="POST">
                    {% csrf_token %}
                    <div class="input-container">
                                        <input  type="text"  name="planname"
placeholder="Назва плану" required>
                        <div class="button-container">
                            <button type="submit">Зберегти</button>
```

```
                        </div>
                    </div>
                </form>
            </div>
                                    <input   type="hidden"   id="csrfToken"
name="csrfmiddlewaretoken" value="{{ csrf_token }}">
            <div class="sortable">
                {% for plan_item in user_plan_items %}
                         <div  class="back"  id="item-{{  plan_item.id  }}"
style="order:{{ plan_item.position|default:'0' }}">
                    <div class="content-wrapper">
                        <div class="icon">
                              <img src="{% static 'users/images/menu.png'
%}" width="50px" height="45px">
                        </div>
                        <div class="image">
                             <img src="{{ plan_item.exercise.image.url }}"
width="250px" height="150px">
                        </div>
                        <div class="text">
                            <h2>{{ plan_item.exercise.description }}</h2>
                                    <input type="number" name="quantity"
value="{{    plan_item.quantity    }}"    min="1"    data-plan-item-id="{{
plan_item.exercise_id }}" class="quantity-input">
                        </div>
                    </div>
                </div>
                {% empty %}
                <div class="back">
                    <p>Ви ще не додали жодної вправи до свого плану.</p>
                </div>
                {% endfor %}
            </div>
        </div>
    </body>
</html>

@login_required
def account(request):
```

```
                                                    user_plans            =
UserExercisePlan.objects.filter(user=request.user).values('plan_id',
'plan_name').distinct()
    context = {'user_plans': user_plans}
    return render(request, 'users/account.html', context)

@login_required
def edit_plan(request):
    plan_id = request.GET.get('plan_id')
    if request.method == 'POST':
        plan_name = request.POST.get('planname')
        if plan_id:
                                                          plan_items     =
UserExercisePlan.objects.filter(user=request.user, plan_id=plan_id)
            for item in plan_items:
                item.plan_name = plan_name
                item.save()
        else:
            plan_id = uuid.uuid4()
            temp_plan = request.session.get('temp_plan', [])
            for item in temp_plan:
                exercise = Exercise.objects.get(id=item['exercise_id'])
                UserExercisePlan.objects.create(
                    user=request.user,
                    exercise=exercise,
                    plan_id=plan_id,
                    plan_name=plan_name,
                    quantity=1,
                    position=0
                )
            request.session.pop('temp_plan', None)
        return redirect('users:account')
    else:
        user_plan_items = []
        if plan_id:
                                                      user_plan_items     =
UserExercisePlan.objects.filter(user=request.user,
plan_id=plan_id).order_by('position').select_related('exercise')
```

```
                        return  render(request,  'users/edit_plan.html',
{'user_plan_items': user_plan_items, 'plan_id': plan_id})
        else:
            temp_plan = request.session.get('temp_plan', [])
            user_plan_items = [{
                'id': "temp-" + str(index),
                'exercise_id': item['exercise_id'],
                'image_url': item['image_url'],
                'description': item['description'],
                'quantity': item.get('quantity', 1),
                'position': index
            } for index, item in enumerate(temp_plan)]
                 return  render(request,  'users/edit_plan_temporary.html',
{'user_plan_items': user_plan_items})

     @login_required
@require_POST
def delete_plan(request):
    plan_id = request.POST.get('plan_id')
    if not plan_id:
          return JsonResponse({'status': 'error', 'message': 'Plan ID is
required'}, status=400)
    try:
                plan  =  UserExercisePlan.objects.filter(plan_id=plan_id,
user=request.user)
        plan.delete()
             return  JsonResponse({'status':  'success',  'message':  'Plan
deleted successfully.'})
    except UserExercisePlan.DoesNotExist:
           return  JsonResponse({'status':  'error',  'message':  'Plan  not
found.'}, status=404)

     class CustomPasswordResetView(PasswordResetView):
    template_name = 'users/password_reset_form.html'
    email_template_name = 'users/password_reset_email.html'
    success_url = reverse_lazy('users:password_reset_done')
class CustomPasswordResetDoneView(PasswordResetDoneView):
    template_name = 'users/password_reset_done.html'
```

```
class CustomPasswordResetConfirmView(PasswordResetConfirmView):
    template_name = 'users/password_reset_confirm.html'
    success_url = reverse_lazy('users:password_reset_complete')
class CustomPasswordResetCompleteView(PasswordResetCompleteView):
    template_name = 'users/password_reset_complete.html'

     urlpatterns = [
    path('account/', account, name='account'),
    path('exercises/', exercises, name='exercises'),
    path('edit_plan/', edit_plan, name='edit_plan'),
    path('delete_plan/', delete_plan, name='delete_plan'),
                  path('add_exercise_to_plan/',     add_exercise_to_plan,
name='add_exercise_to_plan'),
          path('remove_exercise_from_plan/',   remove_exercise_from_plan,
name='remove_exercise_from_plan'),
                    path('save_temporary_plan/',     save_temporary_plan,
name='save_temporary_plan'),
          path('remove_temporary_exercise/',   remove_temporary_exercise,
name='remove_temporary_exercise'),
          path('update_temporary_quantity/',   update_temporary_quantity,
name='update_temporary_quantity'),
    path('update_quantity/', update_quantity, name='update_quantity'),
                path('update_exercise_order/',    update_exercise_order,
name='update_exercise_order'),
    path('register/', RegisterView.as_view(), name='register'),
             path('password_reset/',   CustomPasswordResetView.as_view(),
name="password_reset"),
      path('password_reset/done/', CustomPasswordResetDoneView.as_view(),
name="password_reset_done"),
                                          path('reset/<uidb64>/<token>/',
CustomPasswordResetConfirmView.as_view(), name="password_reset_confirm"),
         path('reset/done/',   CustomPasswordResetCompleteView.as_view(),
name="password_reset_complete"),
]

from django.contrib import admin
from .models import Exercise
admin.site.register(Exercise)
```

метадані

Заголовок

Розробка онлайн-платформи для спортзалу із можливістю створювання особистого кабінету користувача

Автор
Писклинець В. Р.

Науковий керівник / Експерт
кандидат технічних наук Олег Пашкевич

підрозділ
King Danylo University

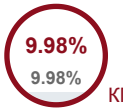
Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

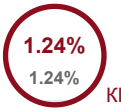
Заміна букв	ß	1
Інтервали	A→	0
Мікропробіли	·	332
Білі знаки	ß	0
Парафрази (SmartMarks)	a	119

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1



КП 2

25
Довжина фрази для коефіцієнта подібності 2

16410
Кількість слів

137346
Кількість символів

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	https://dev.to/joshwizzy/customizing-django-authentication-using-abstractbaseuser-llg	51	0.31 %
2	http://repository.ukd.edu.ua/bitstream/handle/123456789/394/%D0%A0%D1%83%D0%B4%D0%B8%D0%B9%20%D0%90%D0%BD%D0%B4%D1%80%D1%96%D0%B9%20%D0%B4%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0.pdf?sequence=1	36	0.22 %
3	http://repository.ukd.edu.ua/bitstream/handle/123456789/198/%D0%9E%D0%BD%D1%83%D1%84%D1%80%D0%B0%D0%BA%20%D0%86.%20%D0%86..pdf?sequence=1	33	0.20 %
4	http://repository.ukd.edu.ua/bitstream/handle/123456789/394/%D0%A0%D1%83%D0%B4%D0%B8%D0%B9%20%D0%90%D0%BD%D0%B4%D1%80%D1%96%D0%B9%20%D0%B4%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0.pdf?sequence=1	28	0.17 %