Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка віртуального марафону здорового способу життя засобами Python

Виконав: студент IV курсу, групи СНс-42
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

(підпис) Кібук Р. М.
(прізвище та ініціали)

Керівник (підпис) Млинко Б. Б.
(прізвище та ініціали)

Нормоконтроль (підпис) Марценко С.В.
(прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О.
(прізвище та ініціали)

Рецензент (підпис) Гладьо Ю.Б.
(прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ _____
(підпис) (прізвище та ініціали)
«__»_____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Кібуку Ростиславу Максимовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка віртуального марафону здорового способу життя засобами Python

Керівник роботи
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 29 » квітня 2024 року № 4/7-472

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. РОЗДІЛ 1. Постановка задачі розробки марафону здорового способу життя 1.1 Аналіз предметної області, 1.2 Формування вимог розробки веб-сайту 1.3 Пошук актантів та варіантіввикористання веб-сайту 1.3.1 Пошук актантів, 1.3.2 Варіанти використання веб-сайту 1.4 Опис ключових варіантів використання веб-сайту 1.5 Вибір середовища розробки 1.5.1 Мова програмування 1.5.2 База даних 1.6 Обґрунтування використовуваних технологій розробки веб-сайту 1.7 Висновок до першого розділу, 2. Проєктування та реалізація веб-сайту 2.1 Архітектура веб-сайту 2.2 Вибір основних залежностей для розробки віртуального марафону 2.3 Файловаструктура віртуального марафону, 2.4 Структура та особливості використаної БД для розробки віртуального марафону 2.5 Адміністрування віртулаьного марафону 2.6 Розгортання віртуального марафону на хостингових платформах, 2.7 Тестування та використання веб-сайту 2.7.1 Тестування веб-сайту
2.7.2 Використання віртуального марафону 2.8 Висновки до другого розділу, 3. Безпека життєдіяльності, основи хорони праці, 3.1
3.2
3.3 Висновокдо третього розділу, Висновки, Перелік використаних джерел, Додатки.

5. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці			

6. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи		
2.	Пошук джерел щодо розробки та створення веб-		
	сайтів		
3.	Опрацювання обраних джерел		
4.	Аналіз області розробки веб-сайту, підбір технологій		
	для розробки		
5.	Розробка віртуального марафону здорового способу життя		
6.	Оформлення розділу «Постановка задачі розробки		
	Віртуального марафону здорового способу життя»		
7.	Оформлення розділу «Проєктування та реалізація		
	Віртуального марафону здорового способу життя»		
8.	Виконання завдання до підрозділу «Безпека		
	життєдіяльності»		
9.	Виконання завдання до підрозділу «Основи охорони		
	праці»		
10.	Оформлення кваліфікаційної роботи		
11.	Нормоконтроль		
12.	Перевірка на плагіат		
13.	Попередній захист кваліфікаційної роботи		
14.	Захист кваліфікаційної роботи		

Студент _____ Кібук Р.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Млинко Б.Б.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка віртуального марафону з здорового способу життя за допомогою технологій Python // Кваліфікаційна робота освітнього рівня «Бакалавр» // Кібук Ростислав Максимович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНс-42 // Тернопіль, 2024 // С. 71, рис. – 28, табл. – 1, кресл. – 0 , додат. – 7, бібліогр. – 35

Ключові слова: django, python, sql, веб-сайт, health, fitness, nutrition, meditation, yoga, recipes, exercise, mental, health.

Кваліфікаційна робота присвячена розробці віртуального марафону з здорового способу життя за допомогою таких засобів як Python, Django, HTML та CSS.

Метою даної роботи є розробка віртуального марафону здорового способу життя засобами Python.

У першому розділі кваліфікаційної роботи виконано дослідження предметної галузі, визначено ключові вимоги до розробки, обрано відповідні інструменти та описано взаємодію користувачів із системою.

У другому розділі кваліфікаційної роботи розглянуто архітектуру створюваного веб-сайту, описано основні залежності та структуру, розглянуто процес розгортання на хостингових платформах та проведено тестування.

В третьому розділі, було розглянуто правила техніки безпеки при експлуатації обладнання та розрахунок місцевої витяжної вентиляції для верстату чи обладнання.

ANNOTATION

Development of a Virtual Marathon for a Healthy Lifestyle Using Python Technologies // Bachelor's Qualification Work // Rostyslav Maksymovych Kibuk // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Sciences, group SNs-42 // Ternopil, 2024 // Pages 71, Figures – 28, Tables – 1, Drawings – 0, Appendices – 7, Bibliography – 35.

Keywords**:** django, python, sql, website, health, fitness, nutrition, meditation, yoga, recipes, exercise, mental health.

This qualification work is dedicated to the development of a virtual marathon for a healthy lifestyle using tools such as Python, Django, HTML, and CSS.

The aim of this work is to develop a virtual marathon of a healthy childhood using python.

In the first section of the qualification work, the subject area is analyzed, the key requirements for development are defined, the necessary tools are selected, and the interaction of users with the system is described.

In the second section of the qualification work, the architecture of the developed website is examined, the main dependencies and structure are described, the deployment process on hosting platforms is considered, and testing is conducted.

In the third section, the rules of safety during the operation of the equipment and the calculation of local exhaust ventilation for the machine or equipment were considered.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ ІТЕРМІНІВ

HTML – Hypertext Markup Language: Це стандартна мова розмітки для створення веб-сторінок та веб-додатків, яка використовується для визначення структури та розміщення елементів контенту.

CSS – Cascading Style Sheets: Мова стилів, яка використовується для налаштування вигляду та оформлення документів, написаних на HTML чи XML, зокрема задаючи кольори, шрифти та розміщення елементів.

PY – Python Code: Python – це високорівнева мова програмування, яка використовується для автоматизації задач, створення веб-додатків, аналізу даних, написання скриптів, взаємодії з системами та розробки штучного інтелекту.

DB – Database: База даних – це організована колекція структурованої інформації або даних, яка дозволяє ефективно зберігати, керувати, доступатися та аналізувати інформацію.

Стек (Stack): Набір технологій та інструментів, який використовується для розробки веб-сайтів або програм. Стек може включати серверну та клієнтську частини, базу даних, фреймворки та інші компоненти.

Баг (Bug): Програмна помилка або дефект в програмному забезпеченні, який спричиняє неправильне або неочікуване функціонування програми.

DOM – Document Object Model: Інтерфейс програмування, який представляє структуру HTML або XML документів як дерева об'єктів, дозволяючи програмам динамічно змінювати вміст та структуру документів.

SPA – Single Page Application: Веб-додаток, що працює всередині однієї веб-сторінки, де основний контент завантажується один раз, а подальші зміни контенту виконуються без перезавантаження сторінки за допомогою JavaScript.

UI – User Interface: Графічний інтерфейс користувача, що складається з елементів, таких як кнопки, меню, іконки, які дозволяють користувачу взаємодіяти з програмним забезпеченням чи апаратним пристроєм.

Крос-платформеність: Властивість програмного забезпечення працювати на різних операційних системах або апаратних платформах без необхідності значних змін у коді.

Use case: Опис сценарію або діаграма, що ілюструє взаємодію користувачів із системою для досягнення певної цілі, що допомагає визначити функціональні вимоги системи.

Шарінг (Share): Процес обміну або розповсюдження інформації, файлів чи ресурсів між користувачами або системами.

Трек (Track): Аудіофайл або музичний запис, який може включати різні звукові доріжки для створення пісень або інших аудіовиробів.

API – Application Programming Interface: Набір інструментів та протоколів для створення програмного забезпечення, який дозволяє різним програмам взаємодіяти одна з одною.

REST – Representational State Transfer: Архітектурний стиль для створення веб-служб, що дозволяє взаємодію між системами за допомогою HTTP запитів, таких як GET, POST, PUT та DELETE.

JSN – JavaScript Object Notation: Легкий формат обміну даними, який зручно читати та писати людям, а також легко парсити та створювати програмам. Використовується для передачі даних між сервером та клієнтом.

SQL – Structured Query Language: Мова програмування, що використовується для управління та маніпуляції базами даних. Дозволяє виконувати запити для отримання, вставки, оновлення та видалення даних.

HTTP – Hypertext Transfer Protocol: Протокол передачі гіпертексту, який є основою для обміну даними у Всесвітній павутині. Використовується для передачі документів HTML, зображень, відео та інших ресурсів.

OAuth – Open Authorization: Протокол авторизації, який дозволяє стороннім додаткам отримувати обмежений доступ до ресурсів користувача без передачі пароля. Використовується для безпечного доступу до API.

CI/CD – Continuous Integration and Continuous Deployment: Практики автоматизації, що дозволяють командам розробки частіше інтегрувати зміни коду

та автоматично розгортати його в середовищі. Це допомагає забезпечити швидке та якісне оновлення програмного забезпечення.

MVC – Model-View-Controller: Архітектурний шаблон для створення програмного забезпечення, який розділяє програму на три основні компоненти: Модель (Model), Вид (View) та Контролер (Controller). Це спрощує управління складними програмами.

UX – User Experience: Враження та емоції користувача від взаємодії з продуктом або сервісом. Включає зручність використання, ефективність, доступність та задоволеність користувача.

WebSocket: Протокол комунікації, що забезпечує двосторонній зв'язок між клієнтом та сервером в реальному часі. Використовується для створення інтерактивних веб-додатків, таких як чати та онлайн-ігри.

JSON-RPC: Протокол віддаленого виклику процедур, який використовує JSON для передачі даних. Дозволяє клієнтам виконувати методи на віддалених серверах та отримувати результати у форматі JSON.

JWT – JSON Web Token: Стандарт для створення безпечних токенів, які можуть бути використані для аутентифікації та авторизації користувачів. Токени містять закодовану інформацію та можуть бути перевірені на автентичність без звернення до бази даних.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі здоровий спосіб життя стає все більш популярним та важливим для багатьох людей. Враховуючи швидкий темп життя та постійний стрес, багато хто прагне знайти ефективні способи підтримки фізичного та психічного здоров'я. Одним із таких способів є участь у марафонах, які сприяють підвищенню фізичної активності, зміцненню здоров'я та розвитку дисципліни.

Розвиток інформаційних технологій відкриває нові можливості для організації та проведення таких заходів. Віртуальні марафони, зокрема, набувають все більшої популярності завдяки зручності та доступності для широкого кола учасників. Цей проект присвячений розробці віртуального марафону, орієнтованого на здоровий спосіб життя, за допомогою технологій Python.

Проект спрямований на популяризацію здорового способу життя та залучення більшої кількості людей до активних занять спортом. Використання інноваційних технологій дозволить зробити участь у марафоні зручною та привабливою для користувачів, незалежно від їхнього місцезнаходження та рівня фізичної підготовки.

Мета і задачі дослідження. Основна мета проекту – створення інтерактивного веб-сайту, який буде служити платформою для проведення віртуальних марафонів. Веб-сайт надаватиме користувачам можливість реєстрації, участі у марафонах, відстеження свого прогресу, обміну досягненнями та отримання корисної інформації про здоровий спосіб життя. В рамках проекту буде використано сучасні технології розробки, такі як Python, Django, HTML та CSS.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ ВІРТУАЛЬНОГО МАРАФОНУ ЗДОРОВОГО СПОСОБУ ЖИТТЯ

1.1 Аналіз предметної області

Аналіз предметної області є ключовим етапом у розробці будь-якого програмного забезпечення, оскільки він дозволяє зрозуміти основні аспекти, вимоги та виклики, з якими може зіткнутися команда розробників. У випадку з даним проектом, що передбачає створення віртуального марафону для здорового способу життя за допомогою технологій Python, аналіз предметної області охоплює кілька важливих аспектів: тенденції у сфері здорового способу життя, особливості віртуальних марафонів, сучасні технології розробки веб-додатків та вимоги користувачів.

Тенденції у сфері здорового способу життя. У сучасному суспільстві все більше людей звертають увагу на свій фізичний та психічний стан. Популярність здорового способу життя зростає завдяки усвідомленню його важливості для підтримки високого рівня енергії, продуктивності та загального благополуччя. Це включає збалансоване харчування, регулярні фізичні вправи, контроль стресу та достатню кількість сну. Різноманітні ініціативи та програми, такі як фітнес-клуби, йога-студії, додатки для відстеження здоров'я, стають все більш популярними.

Особливості віртуальних марафонів.Віртуальні марафони - це відносно новий, але стрімко розвиваючийся формат спортивних змагань. На відміну від традиційних марафонів, віртуальні не вимагають фізичної присутності учасників у певному місці в певний час. Учасники можуть брати участь з будь-якого місця, використовуючи свої пристрої для відстеження пробігу та досягнень. Це дозволяє залучати широку аудиторію, незалежно від їхнього географічного розташування та рівня підготовки.

Віртуальні марафони мають кілька переваг:

- Гнучкість: Учасники можуть обирати зручний для них час та місце для участі.
- Доступність: Відсутність необхідності подорожувати дозволяє брати участь людям з різних куточків світу.
- Мотивація: Можливість спільного досягнення цілей та обмін результатами сприяє підвищенню мотивації.

Сучасні технології розробки веб-додатків.Для розробки нашого віртуального марафону будуть використані сучасні технології веб-розробки, зокрема Python та Django. Python - це потужна та популярна мова програмування, яка широко використовується для створення різноманітних додатків, включаючи веб-додатки. Django - це високорівневий веб-фреймворк для Python, який забезпечує швидку розробку та чистий, прагматичний дизайн.

- Python: Проста у вивченні та використанні мова, що забезпечує високу продуктивність розробки.
- Django: Фреймворк, що сприяє створенню безпечних та масштабованих веб-додатків з мінімальними витратами часу.

Крім того, для розробки інтерфейсу користувача будуть використані HTML та CSS, що дозволить створити зручний та привабливий дизайн.

Вимоги користувачів.Важливим аспектом успішного проекту є розуміння потреб та очікувань користувачів. У випадку з марафоном для здорового способу життя, основними вимогами користувачів є:

- Зручність використання: Інтуїтивно зрозумілий інтерфейс, що дозволяє легко реєструватися, брати участь у марафонах та відстежувати результати.
- Мотивуючий контент: Інформаційні матеріали, поради щодо здорового способу життя, мотиваційні історії успіху учасників.
- Точність відстеження: Надійні інструменти для вимірювання пробігу та інших фізичних параметрів.
- Соціальні функції: Можливість взаємодії з іншими учасниками, обмін результатами, підтримка та спільне досягнення цілей.

Аналіз конкурентів. Ще один важливий аспект предметної області – це аналіз існуючих рішень та конкурентів. На ринку вже є кілька популярних платформ для віртуальних марафонів, таких як RunKeeper, Strava та інші. Аналіз їхніх функцій, переваг та недоліків допоможе визначити, які можливості необхідно включити в наш проект, а також знайти унікальні підходи для виділення серед конкурентів.

Висновки з аналізу.

На основі проведеного аналізу предметної області можна зробити наступні висновки:

- Здоровий спосіб життя є актуальною та затребуваною темою, що створює великий потенціал для залучення користувачів.
- Віртуальні марафони мають низку переваг перед традиційними, що робить їх привабливими для широкої аудиторії.
- Використання сучасних технологій, таких як Python та Django, дозволить створити ефективний, надійний та зручний у використанні веб-додаток.

Розуміння потреб користувачів та аналіз конкурентів допоможуть розробити продукт, що задовольнить вимоги ринку та приверне увагу потенційних учасників.

Аналіз предметної області є фундаментом для подальшої розробки проекту. Він забезпечує розуміння контексту, визначає ключові вимоги та допомагає обрати оптимальні технології та підходи для успішного втілення ідеї.

1.2 Формування вимог розробки веб-сайту

Формування вимог до розробки веб-сайту є критично важливим етапом, який визначає основні функціональні та нефункціональні можливості майбутньої системи. Для створення ефективного та зручного у використанні віртуального марафону для здорового способу життя необхідно врахувати вимоги з сторони користувачів (клієнтів) та адміністраторів сайту.

З сторони клієнта веб-сайт повинен забезпечувати чимало основних функцій, за допомогою яких буде продуктивно та якісно функціонувати віртуальний марафон здорового способу життя:

- Можливість реєстрації: Форма реєстрації: Користувачі повинні мати можливість зареєструватися на сайті, надаючи базову інформацію (ім'я, електронна пошта, пароль, тощо). Аутентифікація: Після реєстрації користувачі повинні мати можливість входити до системи за допомогою своїх облікових даних.

- Перегляд товару: Каталог товарів: Користувачі повинні мати доступ до перегляду каталогу товарів, які пропонуються на сайті. Детальна інформація про товар: Кожен товар повинен мати окрему сторінку з детальним описом, фотографіями, ціною та іншими характеристиками.Пошук та фільтрація: Користувачі повинні мати можливість здійснювати пошук товарів за ключовими словами та фільтрувати результати за різними критеріями (категорія, ціна, популярність, тощо).

- Головна сторінка: Навігація: Головна сторінка повинна забезпечувати зручну навігацію до основних розділів сайту (каталог товарів, розклад подій, особистий кабінет, контактна інформація). Інформаційний контент: На головній сторінці повинна бути розміщена інформація про сайт, його мету, актуальні новини та пропозиції.

- Розклад спортивних подій: Календар подій: Користувачі повинні мати доступ до календаря з розкладом майбутніх спортивних подій, тренувань та марафонів. Деталі подій: Кожна подія повинна мати сторінку з детальною інформацією про місце, час, умови участі та іншу необхідну інформацію.

З сторони адміністратора веб-сайт повинен забезпечувати такі основні функції:

- Адміністративна панель: Доступ до адмін-панелі: Адміністратори повинні мати захищений доступ до адміністративної панелі для управління сайтом. Управління вмістом: Адміністратори повинні мати можливість редагувати, додавати та видаляти контент на сайті (товари, статті, події, новини). Управління користувачами: Адміністратори повинні мати можливість переглядати, редагувати та видаляти облікові записи користувачів, а також керувати їхніми правами доступу.

- Робота з базою даних користувачів: Перегляд даних користувачів: Адміністратори повинні мати доступ до повної бази даних користувачів, включаючи їх реєстраційну інформацію, історію активності та участі в подіях.

- Аналіз даних: Адміністратори повинні мати інструменти для аналізу користувацьких даних, що допоможе у прийнятті рішень щодо поліпшення сервісу та маркетингових стратегій.

- Безпека даних: Всі дані користувачів повинні бути захищені відповідно до сучасних стандартів безпеки, щоб запобігти несанкціонованому доступу та втраті інформації.

- Крім функціональних вимог, важливо також визначити нефункціональні вимоги, які забезпечать загальну якість та надійність системи:

- Продуктивність: Швидкість завантаження сторінок: Веб-сайт повинен швидко завантажуватися та реагувати на дії користувачів, забезпечуючи позитивний досвід взаємодії. Масштабованість: Система повинна мати можливість ефективно обробляти велику кількість користувачів та даних без зниження продуктивності.

- Безпека: Аутентифікація та авторизація: Всі операції, що вимагають авторизації, повинні бути захищені надійними методами аутентифікації. Захист даних: Особисті дані користувачів повинні зберігатися та передаватися у зашифрованому вигляді для запобігання витокам та несанкціонованому доступу.

- Юзабіліті: Інтуїтивно зрозумілий інтерфейс: Інтерфейс користувача повинен бути простим у використанні, з чіткими та логічними навігаційними елементами. Адаптивний дизайн: Веб-сайт повинен коректно відображатися на різних пристроях (комп'ютери, планшети, смартфони), забезпечуючи комфортне користування незалежно від розміру екрану.

- Надійність: Безперервна робота: Веб-сайт повинен забезпечувати високу доступність та безперебійну роботу, мінімізуючи простої та технічні збої. Відновлюваність: Система повинна мати механізми для швидкого відновлення після можливих збоїв або атак.

Формування вимог розробки веб-сайту є необхідним етапом для забезпечення його відповідності потребам користувачів та адміністрації, а також для досягнення високих стандартів якості, продуктивності та безпеки.

1.3 Пошук актантів та варіантів використання віртуального марафону здорового способу життя

Тепер необхідно зосередитись на пошуку актантів та варіантів використання віртуального марафону здорового способу життя. Актанти, які взаємодіють з таким марафоном, можуть бути різноманітні - від самого учасника марафону до його організаторів та адміністраторів.

Для учасників марафону основним варіантом використання є можливість прийняття участі в спортивних заходах, тренуваннях та інших активностях, які сприяють покращенню їхнього фізичного та психічного здоров'я. Вони можуть реєструватися на події, стежити за своїми досягненнями, обмінюватися досвідом з іншими учасниками, а також отримувати корисні поради та інформацію щодо здорового способу життя. Для учасників марафону також важливим варіантом використання є доступ до різноманітних ресурсів і інформації, які допомагають їм підтримувати здоровий спосіб життя. Це може включати в себе статті, відео та інші матеріали про правильне харчування, фізичну активність, психологічне благополуччя та інші аспекти здорового життя. Учасники можуть користуватися цими ресурсами для отримання нових знань та натхнення для своєї особистої здорової поведінки.

Для організаторів варіантів використання може бути набагато більше. Вони відповідають за планування та організацію марафону, включаючи вибір маршрутів, розклад подій, залучення спонсорів та партнерів, рекламу та просування заходу, а також підготовку необхідного обладнання та персоналу. Організатори марафону, крім того, можуть використовувати платформу для залучення спікерів та експертів з різних галузей здоров'я та фітнесу для проведення вебінарів, майстер-класів та інших освітніх заходів. Це дозволяє учасникам отримувати доступ до ексклюзивної інформації та досвіду, який допомагає їм покращити своє здоров'я та самопочуття.

Адміністратори веб-сайту марафону також мають свої варіанти використання. Вони відповідають за технічну підтримку та адміністрування самого сайту, включаючи оновлення контенту, вирішення технічних проблем, моніторинг безпеки та забезпечення надійності роботи платформи. Адміністратори платформи також можуть використовувати дані, зібрані під час марафону, для аналізу та вдосконалення майбутніх заходів. Вони можуть вивчати поведінку учасників, їхні вподобання та інтереси, щоб підготувати більш цільовану та ефективну програму для майбутніх марафонів.

Узагальнений варіант використання для всіх актантів полягає в створенні та використанні інтерактивної та зручної платформи для організації та участі у віртуальних марафонах здорового способу життя, що сприяє підвищенню свідомості про здоровий спосіб життя та стимулює активний спосіб життя.

1.3.1 Варіанти використання віртуального марафону здорового способу життя

Варіанти використання веб-сайтів зазвичай розглядають різноманітні можливості, які ці платформи надають своїм користувачам. Він може включати опис можливостей для різних категорій актантів, таких як користувачі, адміністратори, організатори подій тощо. У цьому розділі можуть бути висвітлені функціональність сайту, можливості взаємодії користувачів з платформою, можливості персоналізації контенту, а також інструменти для спілкування та обміну інформацією. Також можуть бути включені приклади використання, сценарії взаємодії та потенційні переваги, які користувачі можуть отримати від використання веб-сайту. У цьому розділі можуть також розглядатися варіанти використання платформи з точки зору різних зацікавлених сторін, включаючи користувачів, партнерів, рекламодавців та інших учасників екосистеми веб-сайту. В цілому, розділ про варіанти використання веб-сайтів дозволяє розкрити широкий спектр можливостей, які надають ці платформи, та їхню потенційну вартість для різних сторін.

З варіантів використання веб-сайту "Healthy Living Marathon" можна розглянути широкий спектр можливостей, які надаються учасникам та організаторам марафону. Одним з ключових варіантів використання є доступ до персоналізованих тренувальних програм. Кожен учасник може отримати індивідуальний план тренувань, розроблений на основі його фізичних можливостей, мети участі та інших факторів. Це дозволяє кожному учаснику підібрати оптимальну програму для досягнення своїх цілей з покращення фізичного стану та здоров'я.

Крім того, веб-сайт надає можливість перегляду розкладу подій та тренувань, що дозволяє учасникам вчасно планувати свій час та обирати події, які найбільше відповідають їхнім потребам та інтересам. Такий розклад може включати різноманітні

заходи, такі як тренування з фітнесу, йоги, пілатесу, аеробіки, а також лекції та воркшопи з питань здорового способу життя.

Додатково, веб-сайт створює сприятливу спільноту для учасників, де вони можуть обмінюватися досвідом, порадами та підтримкою один одного. Через форуми, чати та інші комунікаційні інструменти учасники можуть ділитися своїми досягненнями, обговорювати труднощі та отримувати підтримку від співучасників та тренерів. Це створює додатковий мотиваційний фактор для досягнення цілей та підтримує позитивну атмосферу серед учасників марафону.

Зазначені можливості веб-сайту "Healthy Living Marathon" допомагають учасникам ефективно планувати та виконувати свої тренування, отримувати корисну інформацію та підтримку від спільноти, що сприяє покращенню їхнього фізичного та психічного здоров'я.

Короткий опис варіантів використання наведено в таблиці 1.1.

Таблиця 1.1 – Варіантів використання веб - сайту

Актор	Найменування	Формулювання
Користувач без авторизації	Увійти / зареєструватися	Для забезпечення доступу, використання функціональних можливостей веб-сайту, а також для автентифікації користувачів
Користувач авторизований	Покупка послуг, реєстрація на події	Перегляд, купівля, користування кошиком, порівняння

Адміністратор	Огляд, додавання, редагування, видалення	Огляд, додавання, редагування та видалення, включаючи дії щодо інших користувачів

Таким чином, виходячи з табличних даних варіантів використання веб-сайту, можна зрозуміти як кожен з акторів повинен взаємодіяти з системою.

1.4 Опис ключових варіантів використання віртуального марафону

Взаємодія кожного з акторів на веб-сайті відбувається по-різному, кожен актор має свою власну схему взаємодії та очікує від системи конкретної поведінки та реакції. Цю взаємодію акторів можна проілюструвати за допомогою UML діаграми.

Таким чином, для неавторизованого користувача UML діаграма виглядатиме наступним чином:

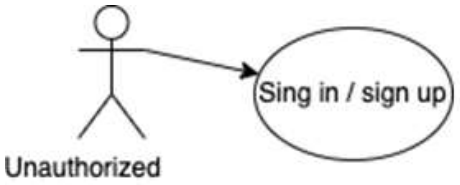

Рисунок 1.1 – Діаграма варіантів використання не авторизованого користувача

Веб-сайт віртуального марафону здорового способу життя має різні функції та можливості для авторизованих та неавторизованих користувачів, що забезпечує гнучкість у використанні платформи залежно від статусу користувача.

Для неавторизованих користувачів основні функції включають доступ до загальної інформації про марафон, перегляд основних подій, тренувань та освітнього контенту, який доступний без реєстрації. Вони можуть ознайомитися з розкладом

заходів, прочитати статті про здоровий спосіб життя, переглянути відео та інші ресурси, що надаються платформою. Однак, неавторизовані користувачі не мають доступу до персоналізованих функцій, таких як створення профілю, реєстрація на події, участь у спільнотах та отримання персоналізованих рекомендацій.

Авторизовані користувачі мають значно ширший доступ до функціоналу веб-сайту. Після реєстрації та входу в систему вони можуть створити персональний профіль, де можна зберігати інформацію про свій стан здоров’я, результати тренувань та досягнення. Цей профіль дозволяє користувачам відслідковувати свій прогрес у марафоні, встановлювати та керувати своїми цілями, а також отримувати персоналізовані рекомендації щодо тренувань та харчування.

Авторизовані користувачі можуть реєструватися на заходи, тренування та інші події, які проводяться на платформі. Вони мають можливість брати участь у віртуальних змаганнях, отримувати доступ до ексклюзивного контенту, такого як спеціалізовані відеоуроки, вебінари та інтерв’ю з експертами. Вони також можуть взаємодіяти з іншими учасниками через форуми, чати та соціальні мережі на платформі, що створює спільноту підтримки та мотивації.

Для адміністраторів веб-сайт надає додаткові інструменти для управління контентом та учасниками. Адміністратори можуть контролювати доступ до різних розділів сайту, управляти реєстраціями на події, моніторити активність користувачів та аналізувати дані для покращення функціоналу платформи. Вони також відповідають за забезпечення безпеки та конфіденційності даних, а також за оновлення та підтримку технічної частини веб-сайту.

Таким чином, авторизовані користувачі отримують доступ до більш персоналізованих функцій і інструментів, що значно розширює їхні можливості для участі в марафоні та підтримки здорового способу життя. В той час як неавторизовані користувачі можуть ознайомитися з основними аспектами марафону, але для більш глибокої інтеграції та взаємодії з платформою необхідна реєстрація та авторизація.

1.5 Вибір середовища розробки веб-сайту

Вибір середовища для розробки веб-сайту є критичним кроком у створенні успішного проекту. У цьому розділі ми розглянемо різні варіанти середовищ для розробки, включаючи їхні характеристики та переваги, щоб зрозуміти, чому Visual Studio Code є оптимальним вибором для розробки віртуального марафону здорового способу життя.

Visual Studio Code (VS Code): Visual Studio Code - це легкий, але потужний редактор коду від Microsoft, який підтримує численні мови програмування та розширення. Він надає інтегровану підтримку для редагування, налагодження та управління проектами. VS Code має розширену екосистему плагінів, що дозволяє користувачам налаштовувати середовище під свої потреби. Однією з головних переваг є підтримка інтеграції з Git, що полегшує систему контролю версій. Завдяки своїм можливостям, зручному інтерфейсу та активному розвитку, VS Code є чудовим вибором для розробки веб-сайтів, включаючи складні проекти, як віртуальний марафон здорового способу життя. [4]

PyCharm: PyCharm - це потужне середовище розробки від JetBrains, яке спеціалізується на Python. Воно надає безліч функцій, таких як інтеграція з системами контролю версій, потужний налагоджувач, підтримка фреймворків Django та Flask, а також вбудовані інструменти для тестування. PyCharm особливо корисний для розробки з Python, але може бути важким для менших проектів через свої великі системні вимоги.

Sublime Text: Sublime Text - це легкий та швидкий текстовий редактор з великою кількістю плагінів, що підтримують різні мови програмування. Він забезпечує простий інтерфейс та високу продуктивність, але для складних проектів може не вистачати вбудованих інструментів для налагодження та управління проектами. Sublime Text добре підходить для швидкого редагування коду та невеликих проектів.

Atom: Atom - це текстовий редактор від GitHub, який пропонує велику кількість плагінів та модульність для налаштування під конкретні потреби розробника. Він має інтуїтивно зрозумілий інтерфейс, але може бути менш стабільним при роботі з великими проектами або при використанні складних розширень. Atom також підтримує інтеграцію з Git, але його продуктивність може знижуватися при великій кількості відкритих вкладок.

WebStorm: WebStorm - це IDE від JetBrains, спеціалізоване на веб-розробці з підтримкою JavaScript, HTML, CSS та популярних фреймворків, таких як React, Angular та Vue.js. Хоча WebStorm є потужним інструментом для веб-розробки, він орієнтований більше на JavaScript і може бути менш ефективним для проектів, де основна мова програмування - Python.

Після аналізу всіх варіантів, Visual Studio Code є найбільш оптимальним вибором для розробки віртуального марафону здорового способу життя. Його легкість, підтримка Python та Django, велика кількість доступних розширень, а також зручність інтеграції з Git роблять його ідеальним інструментом для цього проекту. Visual Studio Code забезпечує всі необхідні функції для ефективної розробки, налагодження та управління проектом, що робить його найкращим вибором для реалізації віртуального марафону здорового способу життя.

1.5.1 Вибір мови програмування

Вибір мови програмування є ключовим етапом у розробці будь-якого програмного продукту, оскільки він визначає не лише технічну реалізацію проекту, але й можливості його подальшого розвитку та підтримки. У даному проекті, спрямованому на створення віртуального марафону здорового способу життя, вибір мови програмування має важливе значення для забезпечення ефективної та зручної реалізації всіх функціональних вимог.

Однією з основних мов, яку буде розглянуто, є Python. Python є високорівневою, інтерпретованою мовою програмування, яка відома своєю простотою і зрозумілістю синтаксису. Це дозволяє розробникам швидко створювати та тестувати код, що є важливим аспектом при розробці складних веб-додатків, таких як віртуальний марафон здорового способу життя.

Python підтримує численні бібліотеки та фреймворки, що значно спрощує розробку веб-сайтів. Серед них особливе місце займає фреймворк Django, який забезпечує розробникам потужні інструменти для створення високонавантажених та безпечних веб-додатків. Django забезпечує швидкий старт проекту завдяки своїм

готовим рішенням для управління базами даних, аутентифікації, маршрутизації, а також безліччю інших функцій, що значно зменшує час розробки.

Однією з ключових переваг Python є його велика і активна спільнота. Це забезпечує наявність великої кількості бібліотек, інструментів та ресурсів для навчання, що значно спрощує розробку і підтримку програмного забезпечення. Крім того, спільнота Python активно розвиває нові технології та інновації, що дозволяє легко інтегрувати сучасні технологічні рішення у ваш проект.

Python також підтримує інтеграцію з різними базами даних, що є важливим аспектом для нашого проекту віртуального марафону. За допомогою Python можна працювати з SQL базами даних, такими як PostgreSQL, MySQL, а також з NoSQL базами даних, такими як MongoDB. Це дозволяє забезпечити ефективне зберігання та обробку великого обсягу даних, що є необхідним для функціонування платформи марафону.

Ще однією важливою особливістю Python є його мультиплатформність, що дозволяє розробляти додатки, які можуть працювати на різних операційних системах, таких як Windows, Linux і macOS. Це забезпечує гнучкість у виборі робочого середовища та дозволяє легко тестувати та впроваджувати зміни у різних середовищах.

Окрім Python, для веб-розробки можуть бути використані й інші мови програмування, такі як JavaScript, яка є незамінною для створення інтерактивного та динамічного інтерфейсу користувача. Однак, основна мова для серверної логіки в нашому проекті залишиться Python, завдяки його простоті, потужності та ефективності у вирішенні завдань, пов’язаних з обробкою даних та інтеграцією з іншими системами.

Для створення віртуального марафону з здорового способу життя було обрано Vue 3 – сучасний та потужний фреймворк для розробки інтерфейсів користувача. Vue 3 відомий своєю гнучкістю, продуктивністю та легкістю у використанні, що дозволяє забезпечити найкращий користувацький досвід для учасників.

Vue 3 використовує новий механізм реактивності, що значно підвищує продуктивність та швидкість роботи додатка. Це дозволяє користувачам миттєво бачити результати своїх дій та отримувати найсвіжішу інформацію без затримок.

Компонентний підхід, який пропонує Vue 3, дозволяє розбивати додаток на невеликі, незалежні частини, що спрощує розробку, тестування та обслуговування. Кожен компонент виконує чітко визначену функцію, що робить код більш організованим та зрозумілим.

Однією з головних переваг Vue 3 є його легка інтеграція з іншими бібліотеками та існуючими проєктами. Це дозволяє використовувати його разом з іншими технологіями, забезпечуючи гнучкість і масштабованість рішення. Крім того, Vue 3 підтримує TypeScript, що додає статичну типізацію до JavaScript, підвищуючи надійність та передбачуваність коду.

Також варто відзначити, що Vue 3 має велику та активну спільноту розробників, що забезпечує доступ до великої кількості ресурсів, навчальних матеріалів та готових рішень. Це робить процес навчання та впровадження фреймворку значно простішим та швидшим.

Для створення зручного та інтуїтивно зрозумілого інтерфейсу користувача віртуального марафону було обрано саме Vue 3. Завдяки цьому фреймворку вдалося реалізувати всі необхідні функціональні можливості, забезпечити високу продуктивність додатка та створити позитивний досвід для користувачів, які прагнуть покращити своє здоров'я та якість життя.

Таким чином, Python з Django та JavaScript з Vue3 є оптимальним вибором для розробки віртуального марафону здорового способу життя. Його простота, потужність, підтримка великої кількості бібліотек та інструментів, а також активна спільнота розробників роблять його ідеальним інструментом для створення високоякісного та ефективного веб-сайту, що відповідає всім сучасним вимогам до безпеки, продуктивності та зручності використання.

1.5.2 Вибір бази даних

Бази даних є основними структурами для зберігання, обробки та управління даними в інформаційних системах. Вони дозволяють ефективно організовувати, зберігати та отримувати великі обсяги інформації, що є критичним для функціонування багатьох сучасних веб-додатків. У цьому розділі ми розглянемо

основні поняття баз даних і підведемо до вибору SQLite3 для реалізації проекту віртуального марафону здорового способу життя.

Основні типи баз даних включають реляційні бази даних (RDBMS), об'єктно-орієнтовані бази даних (OODBMS), документо-орієнтовані бази даних (NoSQL), графові бази даних та інші. Реляційні бази даних, такі як PostgreSQL, MySQL, і SQLite, використовують таблиці для зберігання даних, які можуть бути зв'язані між собою через ключі. Це дозволяє ефективно виконувати складні запити та обробляти великі обсяги інформації.

Реляційні бази даних мають кілька основних характеристик, включаючи використання структурованих таблиць з визначеними схемами, підтримку транзакцій для забезпечення цілісності даних, а також можливість виконання складних запитів за допомогою мови SQL (Structured Query Language). SQL дозволяє створювати, оновлювати, видаляти та отримувати дані з бази даних, що робить його потужним інструментом для взаємодії з даними.

Серед усіх реляційних баз даних, SQLite є особливим рішенням завдяки своїй легкості, простоті у використанні та низьким вимогам до системних ресурсів. SQLite є вбудованою базою даних, що означає, що вона не потребує окремого сервера бази даних і може бути інтегрована безпосередньо в додаток. Це робить SQLite ідеальним вибором для невеликих і середніх проектів, де потрібна проста та ефективна система управління даними. [2]

SQLite має ряд переваг, які роблять його оптимальним для проекту віртуального марафону здорового способу життя. По-перше, він є легким та простим у налаштуванні, що дозволяє швидко інтегрувати базу даних у веб-додаток без потреби в складних конфігураціях чи додаткових ресурсах. По-друге, SQLite забезпечує високу продуктивність при роботі з малими та середніми обсягами даних, що відповідає потребам нашого проекту, де велике значення має ефективне оброблення даних про користувачів, тренування та результати. [2]

SQLite також забезпечує вбудовану підтримку транзакцій, що гарантує цілісність даних навіть при одночасному доступі кількох користувачів. Це особливо важливо для віртуального марафону, де користувачі можуть оновлювати свої профілі, реєструватися на заходи та відправляти результати тренувань в реальному часі. SQLite

забезпечує надійне збереження всіх операцій, що запобігає втраті даних або виникненню конфліктів при одночасному доступі.

Ще однією важливою перевагою SQLite є його можливість працювати без потреби у встановленні окремого серверного програмного забезпечення, що спрощує розгортання та підтримку системи. Це дозволяє знизити витрати на інфраструктуру та спростити процес розробки та підтримки додатку.

Для забезпечення ефективної взаємодії між бекендом на Django та фронтендом на Vue.js використовують API (Application Programming Interface), зокрема REST API або GraphQL. Django Rest Framework (DRF) є популярним інструментом для створення RESTful API в проєктах на Django. DRF дозволяє легко створювати API-ендпоінти, які фронтенд може використовувати для отримання або відправлення даних.

Коли користувач взаємодіє з інтерфейсом на Vue.js, фронтенд надсилає HTTP-запити до API-ендпоінтів, створених на Django. Ці запити можуть бути GET-запитами для отримання даних, POST-запитами для створення нових записів, PUT або PATCH-запитами для оновлення існуючих записів та DELETE-запитами для видалення даних. Django обробляє ці запити, виконує відповідні операції з базою даних і повертає результати у вигляді JSON-відповідей.

Завдяки своїм характеристикам, ці технології стають ідеальним вибором для реалізації проекту віртуального марафону здорового способу життя, забезпечуючи високу продуктивність, простоту використання та надійність. Їхня інтеграція в додаток дозволяє ефективно управляти даними користувачів, тренувань та результатів, що є критично важливим для успішного функціонування платформи.

1.6 Обґрунтування використовуваних технологій розробки віртуального марафону

Розробка віртуального марафону здорового способу життя вимагає ретельного підходу до вибору технологій, які забезпечать не лише функціональність, але й ефективність, масштабованість та безпеку системи. У сучасному світі, де здоров'я та

добробут стають пріоритетами для багатьох людей, створення платформи, яка дозволяє користувачам брати участь у марафонах, слідкувати за своїм прогресом та отримувати підтримку, є надзвичайно актуальним завданням. Вибір правильних технологій є критичним для забезпечення успіху цього проекту, оскільки від нього залежить здатність системи ефективно обробляти дані, інтегруватися з іншими системами та забезпечувати зручний і безпечний доступ для користувачів.

У цьому контексті, кожна технологія, яку ми обираємо для реалізації нашого проекту, повинна відповідати ряду критеріїв: вона має бути зручною для розробки, забезпечувати високу продуктивність, підтримувати масштабування та інтеграцію з іншими системами, а також гарантувати безпеку даних користувачів. Це обумовлює необхідність ретельного аналізу та обґрунтування вибору кожної технології, що буде використовуватись у проекті.

Вибір конкретних технологій базується на їхній здатності відповідати цим вимогам, а також на їхній популярності та підтримці в спільноті розробників. У нашому випадку, мова програмування Python, фреймворк Django, база даних SQLite, а також веб-технології HTML, CSS та JavaScript були обрані з урахуванням їхніх сильних сторін у контексті нашого проекту. Тепер ми розглянемо кожну з цих технологій детальніше та пояснимо, чому вони є оптимальними вибором для створення нашого віртуального марафону.

Розробка віртуального марафону здорового способу життя є складним і багатоаспектним процесом, який вимагає ретельного підбору та інтеграції різних технологій для забезпечення ефективного функціонування системи. У цьому розділі ми розглянемо обґрунтування вибору ключових технологій, таких як мова програмування Python, фреймворк Django, база даних SQLite, а також веб-технології HTML, CSS та JavaScript, що використовуються в проекті.

- Мова програмування Python: Python є однією з найбільш популярних мов програмування завдяки своїй простоті, зручності синтаксису та потужності. Python дозволяє розробникам швидко писати чистий та зрозумілий код, що знижує час на розробку та тестування додатків. Важливою перевагою Python є велика кількість бібліотек та фреймворків, що дозволяє легко інтегрувати різноманітні функції без необхідності розробки з нуля. У контексті нашого проекту Python забезпечує простоту роботи з даними,

інтеграцію з веб-фреймворком Django, а також підтримку асинхронних операцій, що є критично важливим для обробки великого обсягу одночасних запитів користувачів.

- Фреймворк Django: Django є одним із найбільш популярних веб-фреймворків для Python, який забезпечує зручне середовище для розробки веб-додатків. Однією з ключових особливостей Django є його здатність значно скоротити час розробки завдяки своїм «з коробки» рішенням, таким як система аутентифікації, система управління базами даних, підтримка шаблонів та обробка запитів. Django забезпечує високий рівень безпеки, що є важливим аспектом для захисту даних користувачів в нашому віртуальному марафоні. Крім того, Django підтримує масштабування, що дозволяє легко розширювати додаток при зростанні числа користувачів. [1]

- База даних SQLite: SQLite є легкою та потужною вбудованою базою даних, яка забезпечує високий рівень продуктивності при роботі з невеликими та середніми обсягами даних. Вибір SQLite для проекту обґрунтовується його простотою у використанні, низькими вимогами до системних ресурсів та відсутністю необхідності у окремому сервері бази даних. SQLite забезпечує швидке зберігання та обробку даних, що є критично важливим для забезпечення високої швидкості відповіді системи при одночасному доступі великої кількості користувачів. Крім того, SQLite добре інтегрується з Django, що значно спрощує процес налаштування та використання бази даних.

Django, як фреймворк на Python, забезпечує розробникам можливість швидко створювати веб-додатки, використовуючи ORM (Object-Relational Mapping) для взаємодії з базою даних. Django ORM дозволяє розробникам працювати з базами даних, використовуючи об'єкти Python замість SQL-запитів. Це спрощує процес створення, читання, оновлення та видалення записів у базі даних. Водночас Vue.js використовується для створення динамічних інтерфейсів користувача, забезпечуючи реактивний досвід роботи з додатком.

- Веб-технології HTML, CSS та JavaScript: HTML, CSS та JavaScript є основними технологіями для створення сучасних веб-додатків. HTML забезпечує структурування вмісту сторінок, CSS відповідає за їхню візуальну стилізацію, а JavaScript додає інтерактивні функції та динамічні елементи. Використання цих технологій дозволяє створити зручний та привабливий інтерфейс користувача для нашого віртуального марафону. HTML забезпечує базову структуру сторінок, CSS дозволяє налаштувати вигляд

і макет сторінок відповідно до сучасних стандартів веб-дизайну, а JavaScript забезпечує інтерактивність, що дозволяє користувачам легко взаємодіяти з платформою, реєструватися на події, переглядати результати та отримувати персоналізовані рекомендації.

- Інші технології та інструменти: Крім основних технологій, проект використовує різноманітні інші інструменти та бібліотеки для забезпечення повноцінного функціонування веб-додатку. Наприклад, бібліотеки для роботи з формами та валідацією даних, інструменти для тестування коду, а також сервіси для розгортання та хостингу додатка. Всі ці компоненти інтегруються з основними технологіями, забезпечуючи стабільність, масштабованість та ефективність системи.

Вибір зазначених технологій обґрунтовується їхньою здатністю забезпечити високу продуктивність, зручність у використанні та підтримку масштабування, що є критично важливим для створення віртуального марафону здорового способу життя. Завдяки цим технологіям ми можемо створити стабільну, безпечну та ефективну платформу, яка відповідає всім сучасним вимогам до веб-додатків.

Використання Django та Vue.js дозволяє створювати сучасні веб-додатки, які поєднують потужний бекенд для обробки даних та гнучкий фронтенд для взаємодії з користувачами. Django забезпечує надійну роботу з базою даних, тоді як Vue.js надає можливості для створення динамічних та інтерактивних інтерфейсів. Використання API забезпечує ефективну взаємодію між цими двома частинами додатка, дозволяючи створювати швидкі, надійні та масштабовані веб-рішення.

1.7 Висновок до першого розділу

У рамках розробки віртуального марафону здорового способу життя було ретельно підібрано та обґрунтовано вибір технологій, що забезпечують ефективне та надійне функціонування системи. Аналіз предметної області дозволив визначити основні вимоги до проекту, що зумовили вибір мови програмування Python і фреймворку Django для реалізації серверної частини. Python, завдяки своїй простоті, потужності та підтримці великої кількості бібліотек, був обраний як основна мова для розробки, що забезпечує високу продуктивність та зручність у створенні веб-додатків.

Django, як потужний веб-фреймворк, забезпечує швидкий старт розробки завдяки своїм готовим рішенням для аутентифікації, управління базами даних та обробки запитів. Цей вибір дозволяє значно скоротити час розробки та зосередитися на реалізації унікальних функцій проекту. SQLite була обрана для зберігання даних завдяки своїй легкості, простоті інтеграції та низьким вимогам до системних ресурсів, що забезпечує ефективне зберігання та обробку даних користувачів. [1]

Веб-технології HTML, CSS та JavaScript були використані для створення інтерфейсу користувача, що забезпечує зручність та інтерактивність взаємодії з платформою. Ці технології дозволяють створити привабливий та функціональний інтерфейс, який відповідає сучасним стандартам веб-дизайну та забезпечує високу якість користувацького досвіду.

У результаті, обрана технологічна стека забезпечує всі необхідні аспекти для створення успішного віртуального марафону здорового способу життя, включаючи ефективне зберігання даних, зручний і безпечний інтерфейс користувача, а також підтримку масштабування та інтеграції з іншими системами. Завдяки ретельному підбору та обґрунтуванню використаних технологій, проект готовий забезпечити високу продуктивність, надійність і зручність для користувачів, сприяючи популяризації здорового способу життя та залученню більшої кількості людей до активної участі у марафонах.

2. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВІРТУАЛЬНОГО МАРАФОНУ ЗДОРОВОГО СПОСОБУ ЖИТТЯ

2.1 Клієнт-серверна архітектура веб-сайту

Клієнт-серверна архітектура є основою для багатьох сучасних веб-додатків. Вона передбачає розділення системи на дві основні частини: клієнтську (фронтенд) і серверну (бекенд). Така архітектура дозволяє створювати масштабовані, гнучкі та ефективні веб-додатки, що забезпечують якісний користувацький досвід. У цьому розділі буде розглянуто архітектури фронтенду на основі Vue 3 та бекенду з використанням Django.

Vue 3 є сучасним фреймворком для створення користувацьких інтерфейсів. Він відомий своєю реактивністю, продуктивністю та легкістю у використанні. Vue 3 використовує компонентний підхід, який дозволяє розробникам створювати веб-додатки, розділяючи їх на незалежні, повторно використовувані компоненти

Технічно Vue.js зосереджений на рівні ViewModel шаблону MVVM. Він з’єднує представлення та модель двостороннім зв’язуванням даних. Фактичні маніпуляції DOM і форматування виводу абстрагуються в директивах і фільтрах . (див. рис. 2.1). [3]

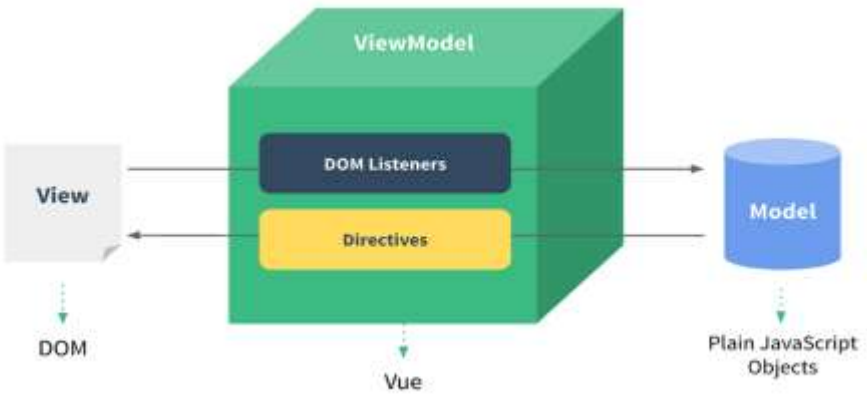

Рисунок 2.1 – Архітектура роботи моделей Vue3

У Vue 3 компоненти є основними будівельними блоками додатка. Кожен

компонент відповідає за певну частину інтерфейсу і включає в себе логіку, шаблони та стилі. Це робить код більш організованим та підтримуваним. Компоненти можуть взаємодіяти один з одним через властивості (props) та події (events), що дозволяє створювати складні й інтерактивні інтерфейси.

Однією з ключових особливостей Vue 3 є реактивність. Фреймворк автоматично відстежує зміни в стані даних і оновлює користувацький інтерфейс відповідно до цих змін. Це дозволяє створювати швидкі та інтерактивні веб-додатки з мінімальними зусиллями.

Django є потужним фреймворком для веб-розробки на Python. Він надає широкий набір інструментів для створення веб-додатків, включаючи ORM (Object-Relational Mapping) для роботи з базами даних, систему аутентифікації користувачів, засоби для обробки запитів та багато іншого. Django дотримується принципу "батарейки включені" (batteries included), що означає, що більшість необхідних функціональностей вже вбудовані в фреймворк (див. рис. 2.2).

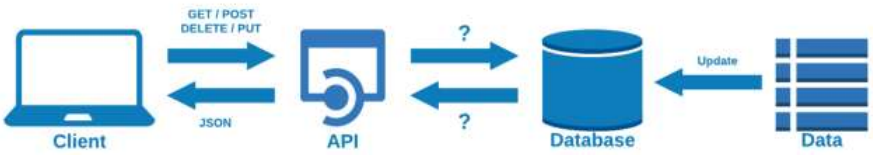

Рисунок 2.2 – Архітектура роботи моделей Vue3

Django використовує модель MVC (Model-View-Controller), де модель відповідає за роботу з базою даних, представлення (view) обробляє запити та повертає відповіді, а контролер (в Django це так званий "view") зв'язує модель і представлення, керуючи логікою додатка.

Django також надає зручний інтерфейс адміністрування, який автоматично генерується на основі моделей бази даних. Це дозволяє розробникам швидко створювати та керувати контентом без необхідності писати додатковий код.

Клієнт-серверна архітектура передбачає взаємодію між фронтендом на Vue 3 та бекендом на Django через API (Application Programming Interface). Для цього часто використовують REST API або GraphQL. Django Rest Framework (DRF) є популярним

інструментом для створення RESTful API в проєктах на Django. DRF дозволяє легко створювати API-ендпоінти, які фронтенд може використовувати для отримання або відправлення даних (див. рис. 2.3).

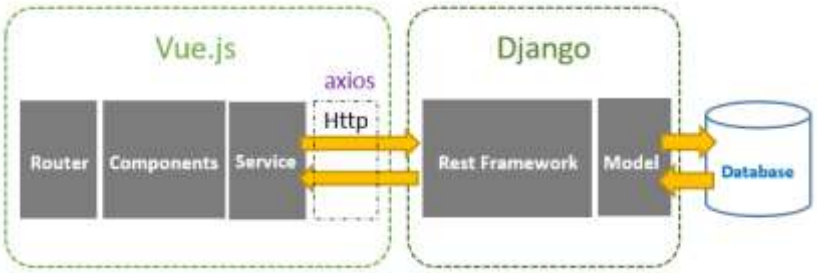

Рисунок 2.3 – Взаємодія vue3 з Django

Коли користувач взаємодіє з інтерфейсом на Vue 3, фронтенд надсилає HTTP-запити до API-ендпоінтів, створених на Django. Ці запити можуть бути GET-запитами для отримання даних, POST-запитами для створення нових записів, PUT або PATCH-запитами для оновлення існуючих записів та DELETE-запитами для видалення даних. Django обробляє ці запити, виконує відповідні операції з базою даних і повертає результати у вигляді JSON-відповідей. Vue 3 отримує ці відповіді і оновлює інтерфейс користувача відповідно до нових даних.

Клієнт-серверна архітектура, яка поєднує фронтенд на Vue 3 та бекенд на Django, дозволяє створювати сучасні, швидкі та надійні веб-додатки. Vue 3 забезпечує динамічний та реактивний інтерфейс користувача, тоді як Django надає потужні інструменти для обробки даних та керування серверною логікою. Використання API для взаємодії між фронтендом та бекендом дозволяє легко масштабувати додаток, розширювати його функціональність та забезпечувати високий рівень продуктивності та зручності для користувачів.

2.2 Вибір основних залежностей для розробки віртуального марафону здорового способу життя

Розробка сучасного веб-сайту потребує ретельного вибору основних залежностей, які забезпечать надійність, продуктивність та гнучкість додатка.

Залежності включають фреймворки, бібліотеки та інструменти, що використовуються для створення та підтримки веб-сайту.

Для фронтенду було обрано Vue 3, сучасний фреймворк для розробки користувацьких інтерфейсів. Vue 3 відомий своєю гнучкістю, продуктивністю та легкістю у використанні. Основні залежності для фронтенд-розробки з Vue 3 включають:

- Vue CLI: Інструмент для швидкого створення та налаштування проектів на Vue.js. Vue CLI дозволяє легко генерувати структуру проекту, додавати необхідні плагіни та налаштування.
- Vue Router: Бібліотека для керування маршрутизацією у Vue-додатках. Вона дозволяє створювати односторінкові додатки з динамічною навігацією без перезавантаження сторінки.
- Vuex: Становий менеджер для Vue-додатків, що забезпечує централізоване керування станом додатка. Це особливо корисно для великих додатків з багатьма компонентами, що потребують обміну даними.
- Axios: Бібліотека для виконання HTTP-запитів, що дозволяє фронтенду взаємодіяти з бекендом через API. Axios забезпечує простий та гнучкий спосіб роботи з асинхронними запитами та обробкою відповідей.

Для бекенду було обрано Django, потужний фреймворк для веб-розробки на Python. Django забезпечує надійну роботу з базою даних, обробку запитів та керування серверною логікою. Основні залежності для бекенд-розробки з Django включають:

- Django Rest Framework (DRF): Бібліотека для створення RESTful API в проектах на Django. DRF дозволяє легко створювати API-ендпоінти для взаємодії з фронтендом та обробки даних.
- Django ORM: Вбудований інструмент для роботи з базами даних, що дозволяє розробникам взаємодіяти з базою даних за допомогою об'єктів Python замість написання SQL-запитів. Django ORM спрощує процес створення, читання, оновлення та видалення записів у базі даних.

- Django Allauth: Бібліотека для керування аутентифікацією та реєстрацією користувачів. Вона підтримує соціальну аутентифікацію через популярні сервіси, такі як Google, Facebook та інші, що робить процес входу для користувачів зручним та швидким.
- Django CORS Headers: Додаток для керування заголовками CORS (Cross-Origin Resource Sharing), що дозволяє бекенду обробляти запити з інших доменів. Це особливо важливо для взаємодії між фронтендом та бекендом, розміщеними на різних серверах.

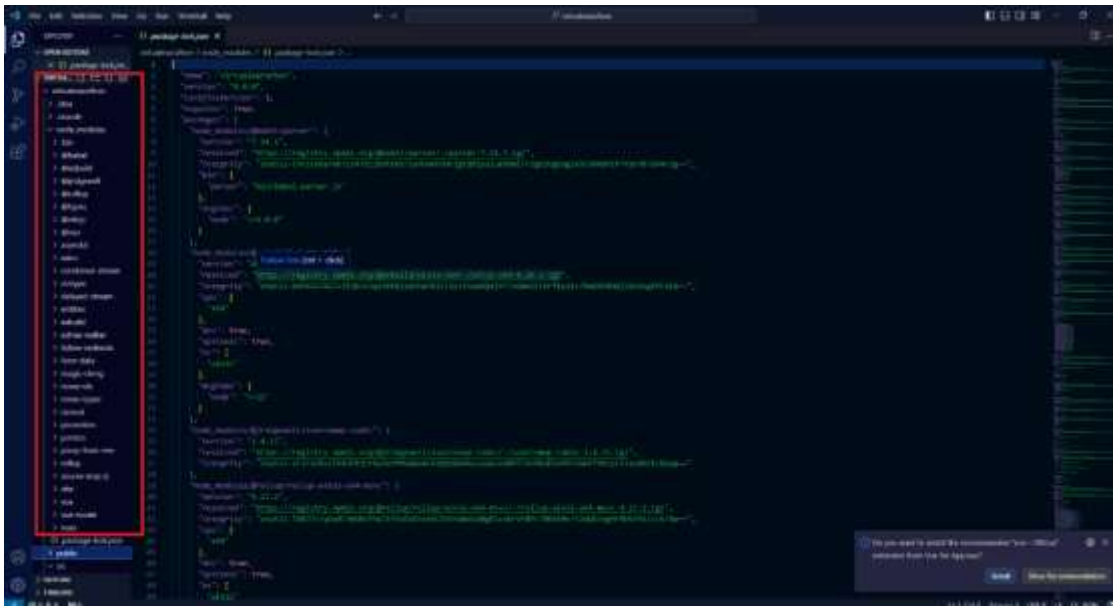

Рисунок 2.4 – Список залежностей які є в проекті

Окрім основних залежностей для фронтенду та бекенду, для ефективної розробки та розгортання веб-сайту використовуються додаткові інструменти:

- Git: Система керування версіями, що дозволяє розробникам відстежувати зміни в коді, працювати в командах та керувати різними версіями проекту. Git забезпечує безпеку коду та спрощує процеси розробки та інтеграції.
- Docker: Платформа для контейнеризації, що дозволяє розробникам створювати, розгортати та запускати додатки в ізольованих середовищах.
- PostgreSQL: Потужна реляційна база даних, яка часто використовується з Django. PostgreSQL забезпечує високу продуктивність, надійність та масштабованість, що робить її відмінним вибором для зберігання даних веб-додатків.

- Nginx: Веб-сервер та зворотний проксі-сервер, що забезпечує ефективне оброблення запитів та балансування навантаження. Nginx часто використовується для розгортання додатків, що працюють на Django.
- Webpack: Інструмент для збірки модулів JavaScript, який використовується з Vue.js. Webpack дозволяє оптимізувати ресурси, такі як JavaScript, CSS та зображення, для забезпечення швидкого завантаження та ефективної роботи додатка

Вибір основних залежностей для розробки веб-сайту є критичним етапом, який визначає надійність, продуктивність та зручність розробки та підтримки додатка.

2.3 Файлова структура віртуального марафону здорового способу життя

Віртуальний марафон здорового способу життя структурований таким чином, щоб відобразити поділ на фронтенд і бекенд, що сприяє чіткому розподілу відповідальностей і спрощує керування кодовою базою (див. рис. 2.5).

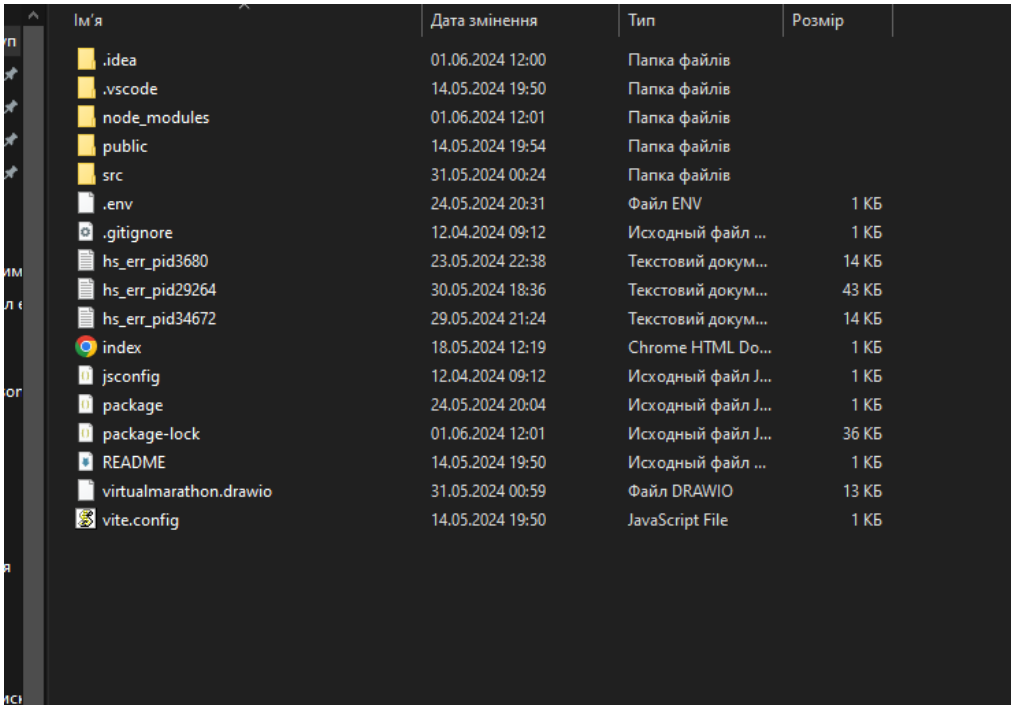

Рисунок 2.5 – Файлова структура проекту

Загальна структура проекту складається з кількох основних директорій: backend, frontend, config, docs, scripts, tests, а також головного файлу README.md.

Кожна з цих директорій має своє призначення та містить відповідні файли і піддиректорії, необхідні для роботи проекту.

Бекендова частина проекту реалізована за допомогою Django і зберігається в директорії backend. Тут міститься код серверної логіки, роботи з базою даних та API.

Головна конфігураційна директорія бекенду, відома як marathon, містить файли налаштувань, маршрутизації та файли для запуску проекту, такі як settings.py, urls.py, wsgi.py і asgi.py.

У директорії apps розміщені додатки проекту. Кожен додаток має свою власну піддиректорію, наприклад, users для управління користувачами та events для управління подіями марафону. Кожен додаток містить файли моделей, адміністрування, представлень (views), серіалізаторів та маршрутів.

Статичні файли, такі як CSS, JavaScript та зображення, які використовуються на серверній стороні, зберігаються в директорії static. Шаблони HTML, що використовуються для рендерингу сторінок на серверній стороні, містяться в директорії templates. Файл manage.py є скриптом для керування Django-проектом, що дозволяє запускати сервер, виконувати міграції бази даних та інші адміністративні завдання.

Фронтендова частина проекту реалізована на Vue 3 і зберігається в директорії frontend. Тут розміщений код користувацького інтерфейсу та логіка взаємодії з бекендом.

Директорія public містить статичні файли, які обслуговуються веб-сервером безпосередньо. Основним файлом тут є index.html, який виступає головним шаблоном сторінки.

Основна частина вихідного коду фронтенду знаходиться в директорії src. У цій директорії є піддиректорії assets, components, router, store, та pages. Директорія assets містить статичні ресурси, такі як стилі та зображення. У директорії components зберігаються повторно використовувані компоненти Vue. Директорія router містить налаштування маршрутизації для Vue Router, а store - налаштування станового менеджера Vuex. Компоненти, які відповідають за цілі сторінки або великі секції сторінок, зберігаються в директорії views. Директорія pages містить утиліти та

допоміжні функції. Головний компонент додатку зберігається у файлі App.vue, а точка входу додатку - у файлі main.js.

Директорія docs містить документацію проекту, включаючи технічні описи, інструкції для розробників та користувачів. У директорії scripts розміщені сценарії для автоматизації завдань, таких як розгортання, збірка та тестування. Директорія tests містить тести для проекту, охоплюючи як фронтенд, так і бекенд. Головний файл README.md містить інформацію про проект, інструкції по встановленню та використанню.

Організована файлова структура допомагає розробникам ефективно працювати з кодовою базою, забезпечує чіткий розподіл відповідальностей та спрощує масштабування і підтримку проекту. Використання чітко визначених директорій для бекенду, фронтенду та додаткових інструментів сприяє кращій організації роботи і забезпечує високу якість розробки віртуального марафону здорового способу життя.

2.4 Структура та особливості використаної БД для розробки веб-сайту

База даних є ключовим елементом для зберігання та управління даними веб-сайту. Для віртуального марафону здорового способу життя використовується база даних SQLite, яка забезпечує простоту використання і швидкість налаштування. У цьому розділі буде розглянута структура бази даних, її основні компоненти та особливості використання в контексті даного веб-сайту.

Для віртуального марафону була обрана SQLite, з таких причин як:

- Простота налаштування: SQLite не вимагає окремого серверного процесу, що робить її ідеальним вибором для розробки та невеликих додатків.
- Легкість інтеграції: SQLite інтегрується безпосередньо у Django без необхідності додаткових налаштувань, що спрощує розробку.
- Відсутність потреби в масштабуванні: Для проекту, який розрахований на невелику кількість користувачів і середній обсяг даних, SQLite забезпечує достатню продуктивність.

У фрагменті коду з файлу налаштувань settings.py визначається, що буде використовуватись база даних SQLite. Це видно з блоку налаштувань у секції DATABASES:

- ENGINE: 'django.db.backends.sqlite3' вказує на те, що використовується SQLite як система управління базами даних.
- NAME: BASE_DIR / 'db.sqlite3' визначає місце зберігання бази даних, яка знаходиться у файлі db.sqlite3 в кореневій директорії проекту.

Таким чином, для цього проекту Django налаштовано використання SQLite як основної бази даних.

База даних проекту організована у вигляді файлу db.sqlite3, який знаходиться в кореневій директорії проекту. Основні таблиці та їх структура визначаються через моделі Django (див. рис. 2.6).

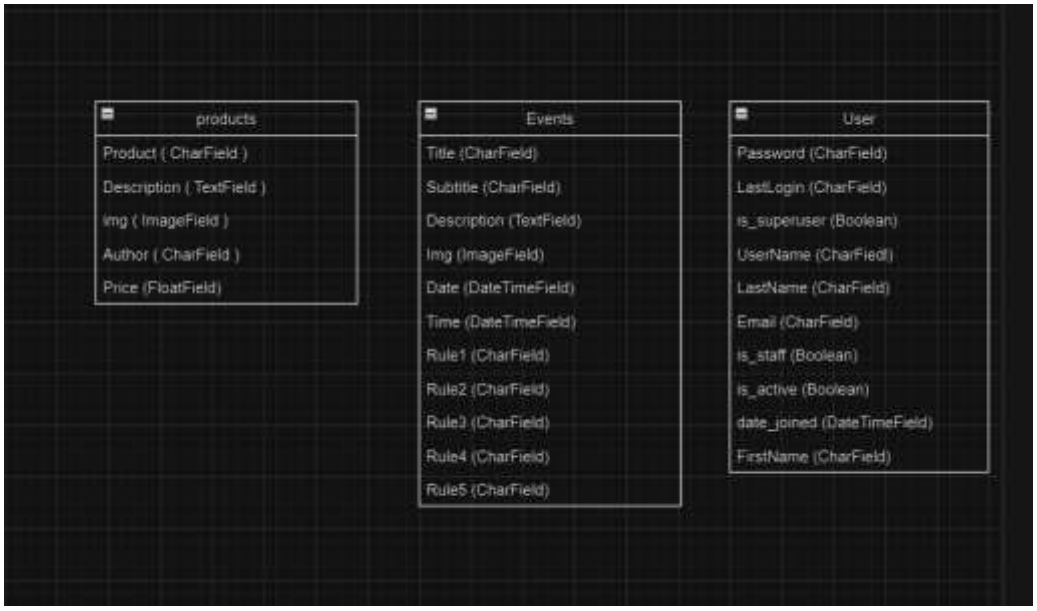

Рисунок 2.6 – Концептуальна схема бази даних віртуального марафону

Таблиця користувачів зберігає інформацію про зареєстрованих користувачів. Поля включають ідентифікатор користувача, ім'я, електронну пошту, зашифрований пароль, дату створення та оновлення облікового запису. Ця таблиця також може містити додаткову інформацію, таку як роль користувача (адміністратор, учасник) та налаштування профілю.

Таблиця подій зберігає інформацію про всі заплановані події марафону. Поля включають ідентифікатор події, назву, опис, дату початку та завершення, місце проведення та організатора. Ця таблиця також може зберігати інформацію про статус події (активна, завершена, скасована).

Таблиця учасників зберігає інформацію про користувачів, які зареєструвалися для участі в подіях. Поля включають ідентифікатор учасника, ідентифікатор події, до якої вони приєдналися, дату реєстрації та статус участі.

Таблиця активностей зберігає записи про фізичну активність користувачів під час марафону. Поля включають ідентифікатор активності, ідентифікатор користувача, тип активності (біг, ходьба, велоспорт), тривалість, дистанцію та дату.

Django ORM (Object-Relational Mapping) дозволяє розробникам взаємодіяти з базою даних через об'єкти Python, що значно спрощує розробку. Django ORM автоматично генерує SQL-запити на основі визначених моделей, що знижує ризик помилок і підвищує продуктивність розробки.

Django підтримує потужний механізм міграцій, який дозволяє керувати змінами структури бази даних. Розробники можуть створювати, застосовувати та відкочувати міграції, що спрощує процес розширення функціональності. Міграції створюються автоматично на основі змін у моделях Django і зберігаються в спеціальних файлах міграцій.

Для забезпечення високої продуктивності веб-сайту використовуються різні методи оптимізації бази даних, такі як індексація, кешування та оптимізація запитів. Індексація полів, які часто використовуються у фільтрації та сортуванні, значно підвищує швидкість виконання запитів. Кешування зменшує навантаження на базу даних, прискорюючи доступ до часто використовуваних даних. Оптимізація запитів включає написання ефективних SQL-запитів та використання методів попереднього завантаження даних для зменшення кількості запитів до бази даних.

Використання SQLite у поєднанні з Django ORM забезпечує просту, але надійну систему управління даними для віртуального марафону здорового способу життя. Чітка структура бази даних і ефективне використання ORM сприяють успішному функціонуванню веб-сайту, забезпечуючи високу якість обробки даних і зручність у розробці.

2.5 Адміністрування віртуального марафону здорового способу життя

Адміністрування сайту є важливою частиною підтримки та управління веб-платформою, що дозволяє контролювати та модифікувати різні аспекти сайту. Це включає в себе управління користувачами, контентом, налаштуваннями та іншими елементами, які впливають на функціональність і безпеку сайту. У випадку з Django, адміністрування сайту може бути здійснене через вбудовану адміністративну панель, яка значно спрощує цей процес.

Щоб потрапити в адміністративну панель Django, необхідно спочатку увійти в систему. Це зазвичай робиться через веб-браузер, використовуючи URL-адресу, яку визначає конфігурація проекту. URL-адреса для адміністративної панелі віртуального марафону виглядає як https://server.testnetwork.top/admin (див. рис. 2.7).

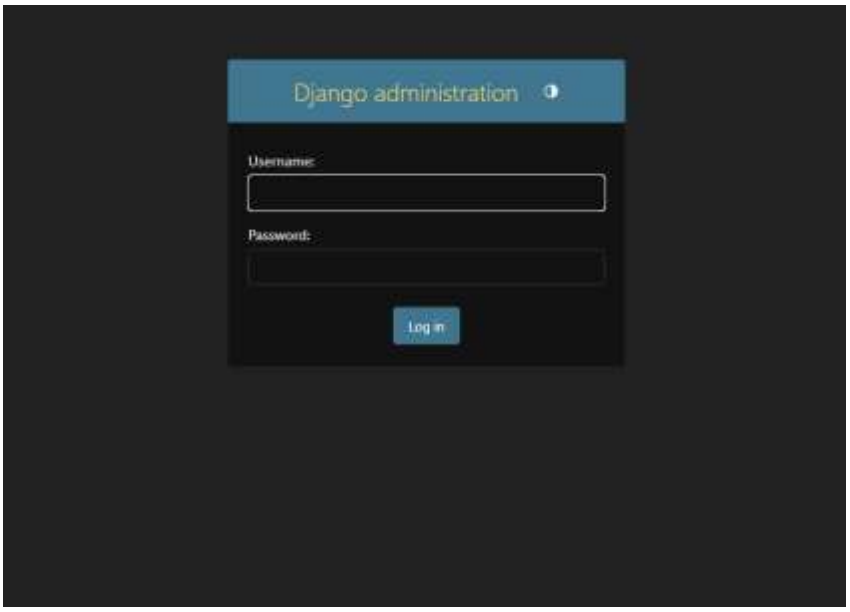

Рисунок 2.7 – Вхід в адмін-панель Django

Щоб мати доступ до адміністративної панелі, користувач повинен бути зареєстрованим і мати відповідні права доступу. Для цього під час реєстрації або додавання користувача в Django потрібно вказати необхідні права (наприклад, доступ до адмін-панелі).

Адміністративна панель Django автоматично генерується на основі моделей, визначених у проекті. Це потужний інструмент, який дозволяє виконувати різні операції з даними без необхідності писати додатковий код. Панель включає такі функції, як перегляд, додавання, редагування і видалення записів у базі даних (див. рис. 2.8). [10]

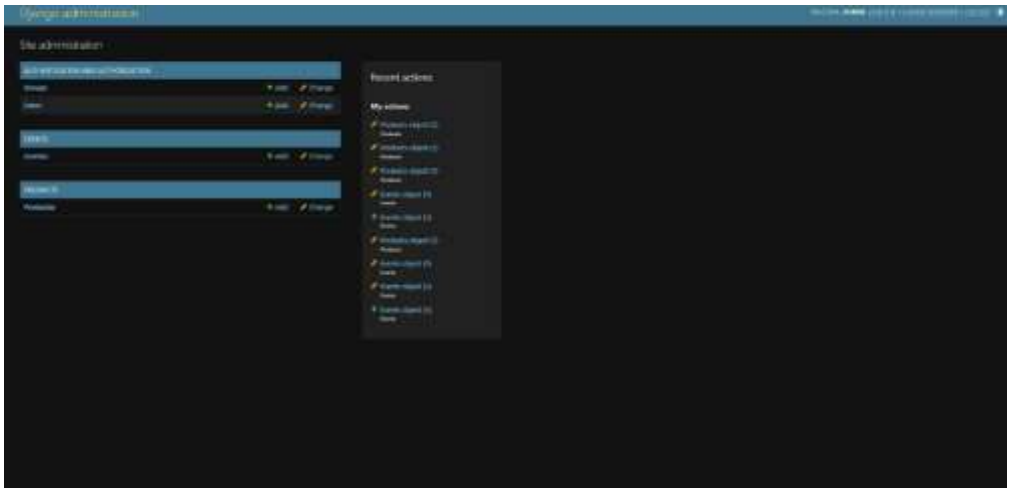

Рисунок 2.8 – Адмін-панель Django

- Перегляд даних: Вся інформація, яка зберігається в базі даних, відображається у вигляді таблиць з можливістю фільтрації і сортування.
- Додавання нових записів: Легке додавання нових записів у таблиці, що дозволяє швидко оновлювати контент.
- Редагування записів: Зміна даних у вже існуючих записах без необхідності вносити зміни в код.
- Видалення записів: Видалення небажаних записів з бази даних.

У адміністративній панелі Django для управління віртуальним марафоном доступні наступні основні категорії: Groups, Events, Users, Products.

Groups. У цьому розділі адміністратор може створювати, редагувати та видаляти групи, які можуть бути використані для організації учасників за інтересами чи іншими критеріями. Це може включати групи за типом активності, рівнем підготовки або географічним положенням (див. рис 2.9).

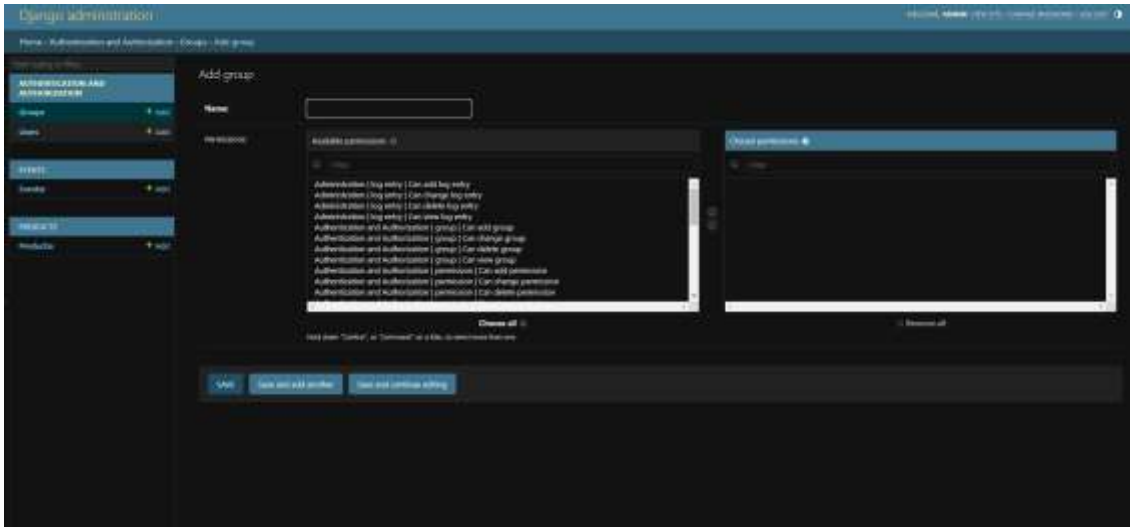

Рисунок 2.9 – Редагування в розділі Groups

Events. В розділі подій адміністратор може управляти всіма аспектами заходів, що проводяться в рамках марафону. Це включає створення нових подій, редагування існуючих подій, встановлення дати і часу, опису та місця проведення. Адміністратор може також змінювати статус подій, наприклад, активна, завершена або скасована (див. рис 2.10).

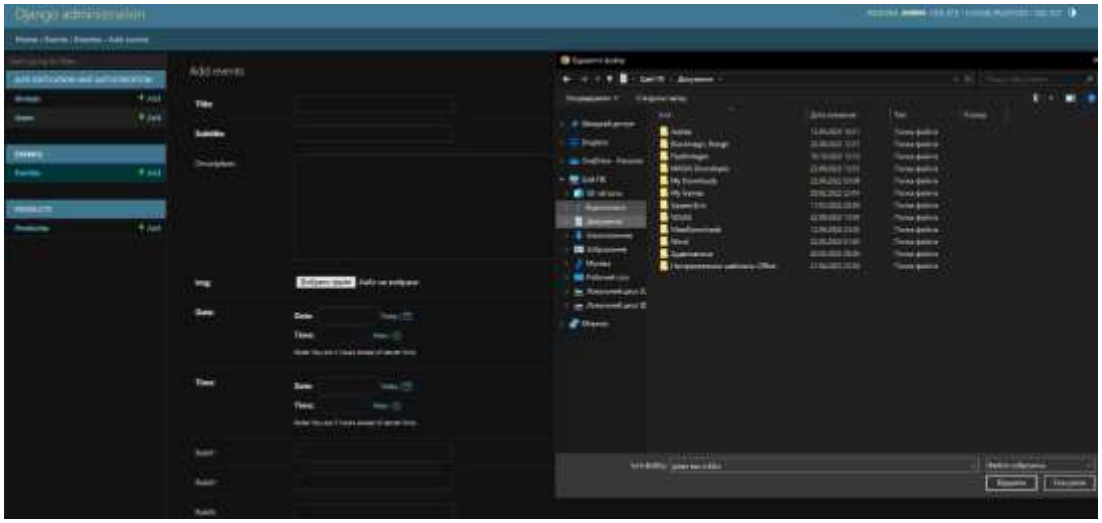

Рисунок 2.10 – Редагування в розділі Events

Users. Цей розділ дозволяє управляти інформацією про користувачів, які зареєстровані на сайті. Адміністратор може додавати нових користувачів, змінювати їхні профілі, призначати їм ролі і доступ до різних частин сайту (див. рис 2.11).

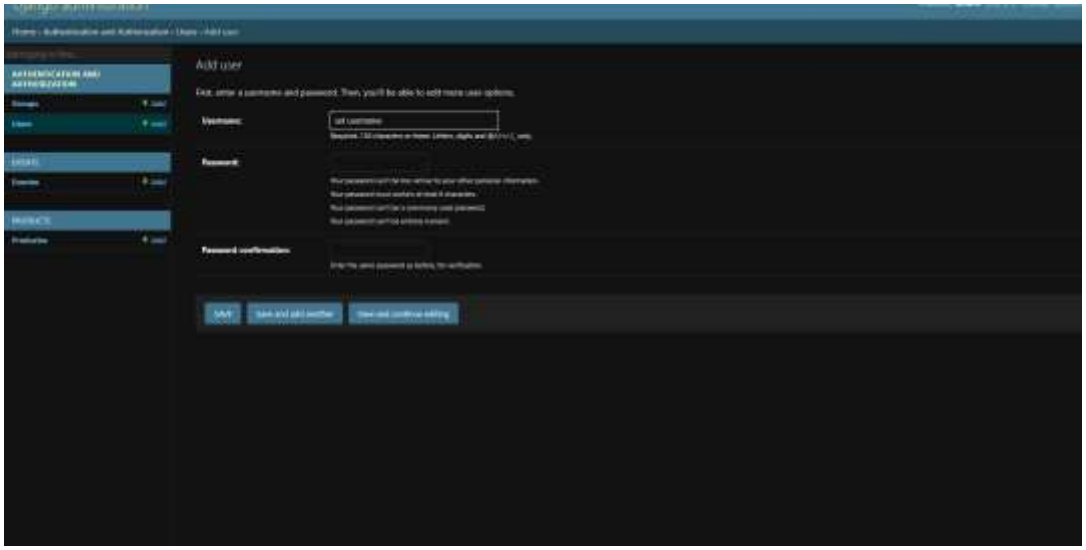

Рисунок 2.11 – Редагування в розділі Users

Products. У цьому розділі адміністратор може керувати товарами або послугами, що пропонуються в рамках марафону. Це включає додавання нових продуктів, редагування описів, ціни, кількості на складі та інших характеристик. Адміністратор може також переглядати замовлення, пов'язані з продуктами, і змінювати їх статус (див. рис 2.12).

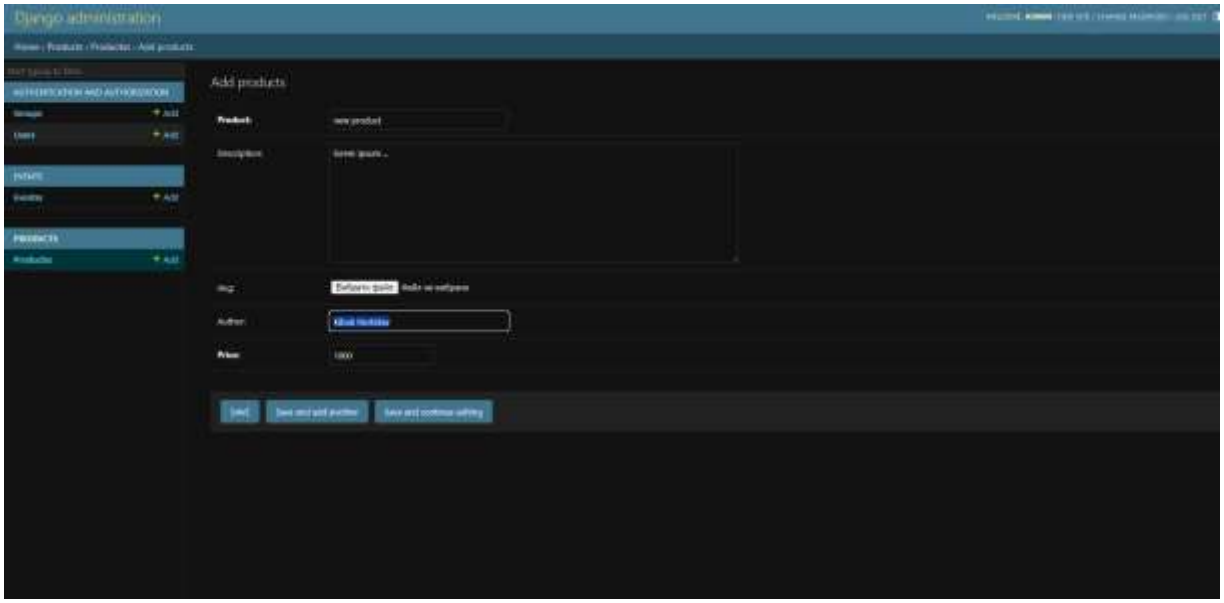

Рисунок 2.12 – Редагування в розділі Products

Адміністрування сайту через адміністративну панель Django є потужним і зручним інструментом для управління різними аспектами веб-сайту. Це дозволяє легко та ефективно управляти даними, забезпечуючи швидке внесення змін і

підтримку актуальності контенту, що є критично важливим для успішного функціонування віртуального марафону здорового способу життя.

2.6 Розгортання веб-сайту на хостингових платформах

Розгортання веб-сайту на хостингових платформах включає в себе процес перенесення коду проекту на сервер, щоб він був доступний для користувачів через Інтернет. Це комплексний процес, який включає налаштування серверного середовища, установки необхідного програмного забезпечення, а також налаштування параметрів безпеки і доступу. Хостинг платформи надають інструменти та сервіси, які полегшують цей процес, забезпечуючи необхідні ресурси для розміщення веб-сайтів.

Для розгортання віртуального марафону здорового способу життя було обрано Digital Ocean. Digital Ocean є популярною платформою для хостингу, яка надає інфраструктуру як сервіс (IaaS) і пропонує прості у використанні рішення для розгортання, які підходять для різноманітних проектів, від малих стартапів до великих додатків. Основними причинами вибору Digital Ocean є її надійність, простота налаштування, масштабованість та доступні ціни, що робить її ідеальним вибором для середніх проектів, які потребують стабільної роботи та високої продуктивності. [12]

Розгортання веб-сайту на Digital Ocean включала кілька ключових кроків, починаючи від створення нового сервера до налаштування веб-сервера та бази даних. Основні етапи, які було виконано:

Крок 1: Створення Droplet. Droplet — це віртуальний сервер на Digital Ocean. Для початку необхідно створити новий Droplet, який буде використовуватись для хостингу веб-сайту. Після реєстрації на платформі та входу в особистий кабінет потрібно вибрати параметри для Droplet, такі як розмір серверу, регіон розташування і операційну систему. Для Django проекту зазвичай обирають Ubuntu через його сумісність з Django і великою кількістю доступних інструкцій та ресурсів (див. рис. 2.13).

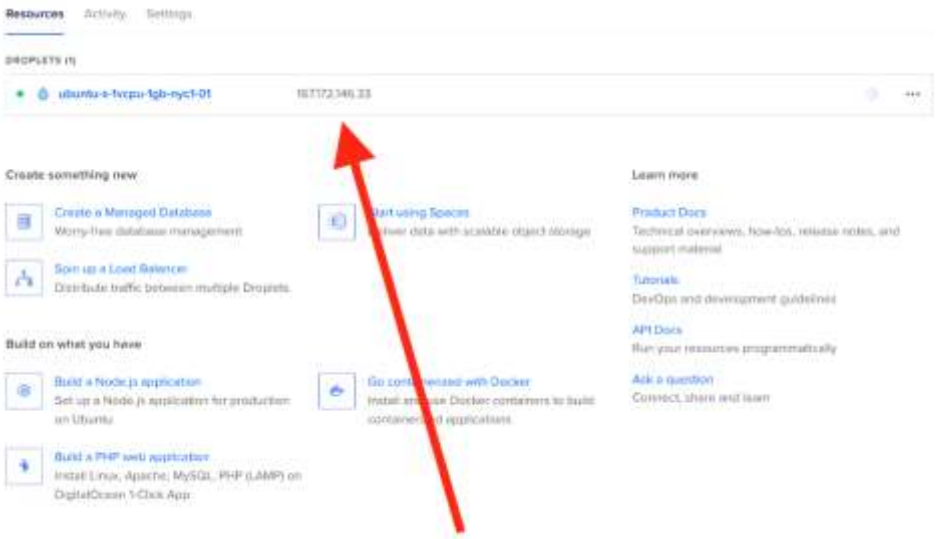

Рисунок 2.13 – Створення Droplet

Крок 2: Налаштування серверу. Після створення Droplet, необхідно підключитися до сервера через SSH. Це можна зробити, використовуючи термінал або будь-який SSH-клієнт. Після підключення можна виконувати різні налаштування, включаючи оновлення системи, налаштування firewall (за допомогою UFW або іншого інструменту) та встановлення необхідного програмного забезпечення, такого як Python, pip, virtualenv, і web server (наприклад, Gunicorn або uWSGI) (див. рис. 2.14).

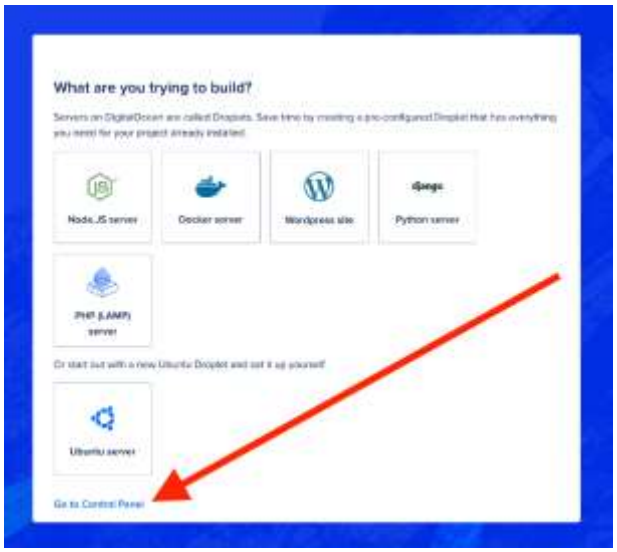

Рисунок 2.14 – Перехід до контрольної панелі серверу

Крок 3: Налаштування бази даних. Для проекту, що використовує Django, важливо налаштувати базу даних. На Digital Ocean можна створити додатковий Droplet для бази даних або використовувати Managed Database сервіси, якщо вони

доступні. Для SQLite, який за замовчуванням використовується в Django, необхідно лише скопіювати файл бази даних на сервер. Для більш складних баз даних, таких як PostgreSQL або MySQL, слід встановити відповідний сервер бази даних і налаштувати його відповідно до вимог проекту (див. рис. 2.15).

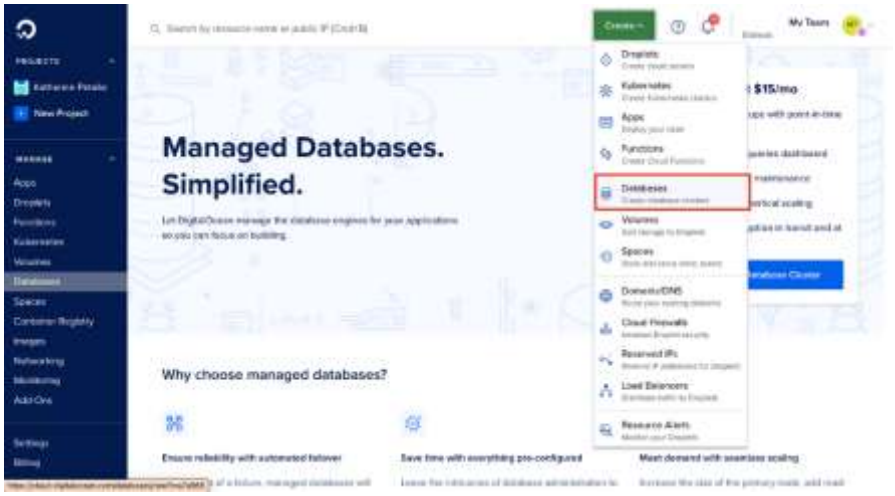

Рисунок 2.15 – Налаштування бази даних на digital ocean

Крок 4: Налаштування веб-сервера. Для обробки HTTP-запитів і сервування файлів сайту слід налаштувати веб-сервер. Gunicorn або uWSGI можна використовувати для запуску Django додатків, а Nginx або Apache як зворотний проксі сервер. Це дозволяє розділити обробку запитів на сервері і обробку додатків, що забезпечує більшу продуктивність і безпеку.

Крок 5: Налаштування доменного імені. Щоб веб-сайт був доступний через доменне ім'я, необхідно налаштувати DNS-записи для доменного імені. На Digital Ocean можна налаштувати DNS через панель управління або використовувати сторонні DNS-провайдери. Зазвичай це включає додавання записів A або CNAME, які вказують на IP-адресу Droplet (див. рис. 2.16).

Рисунок 2.16 – Налаштування доменного імені

Крок 6: Деплой Django додатка. Для деплою Django додатка на сервер можна використовувати Git або інші системи контролю версій. Після завантаження коду на сервер слід налаштувати перемінні середовища, включаючи налаштування бази даних, секретні ключі та інші параметри. За допомогою інструментів як gunicorn або uwsgi можна запустити додаток, а за допомогою Nginx налаштувати зворотний проксі, щоб направляти HTTP-запити до Gunicorn або uWSGI.

Крок 7: Налаштування HTTPS. Для забезпечення безпеки веб-сайту рекомендується використовувати HTTPS. Це можна зробити за допомогою Let's Encrypt, який пропонує безкоштовні SSL-сертифікати. Налаштування HTTPS зазвичай включає встановлення Certbot на сервер, отримання сертифікатів та налаштування Nginx або Apache для використання цих сертифікатів (див. рис. 2.17).

Рисунок 2.17 – HTTPS налаштування

Розгортання веб-сайту на Digital Ocean є ефективним процесом, що включає кілька ключових етапів від створення сервера до налаштування HTTPS. Цей процес дозволяє забезпечити надійну, безпечну і продуктивну роботу веб-сайту, що є критично важливим для успішного функціонування віртуального марафону здорового способу життя. Digital Ocean надає зручні інструменти та ресурси, які значно спрощують цей процес, роблячи його доступним навіть для менш досвідчених розробників. [13]

2.7 Тестування та використання віртуального марафону

2.7.1 Тестування веб-сайту

Для віртуального марафону здорового способу життя тестування включало функціональне тестування, тестування інтерфейсу користувача та тестування продуктивності, з особливою увагою до виявлення і виправлення помилок.

Функціональне тестування було проведено для забезпечення того, що всі функції сайту відповідають вимогам. Це включало тестування сценаріїв користувача, таких як реєстрація, аутентифікація, створення та редагування подій, а також взаємодія з продуктами. Автоматизовані тести, написані за допомогою Selenium та pytest, дозволили виявити помилки у логіці роботи сайту. Було виправлено проблеми з обробкою помилок при реєстрації користувачів, а також неполадки з відображенням даних у деяких формах. [24]

Тестування інтерфейсу користувача (UI) включало перевірку зовнішнього вигляду і зручності використання сайту. Було проведено тестування коректності відображення сторінок, функціонування елементів інтерфейсу, таких як форми, кнопки та меню. Інструменти для UI тестування, такі як Cypress та Jest, були використані для автоматизації процесу, що дозволило виявити проблеми з розташуванням елементів на мобільних пристроях. Після виявлення цих помилок були внесені коригування для забезпечення адаптивності інтерфейсу на різних пристроях.[11]

Навантажувальне тестування було проведено для оцінки продуктивності сайту під час збільшення навантаження. Для цього використовувався інструмент WebPageTest, який імітував реальні умови використання з різними рівнями трафіку (див. рис. 2.18). [14]

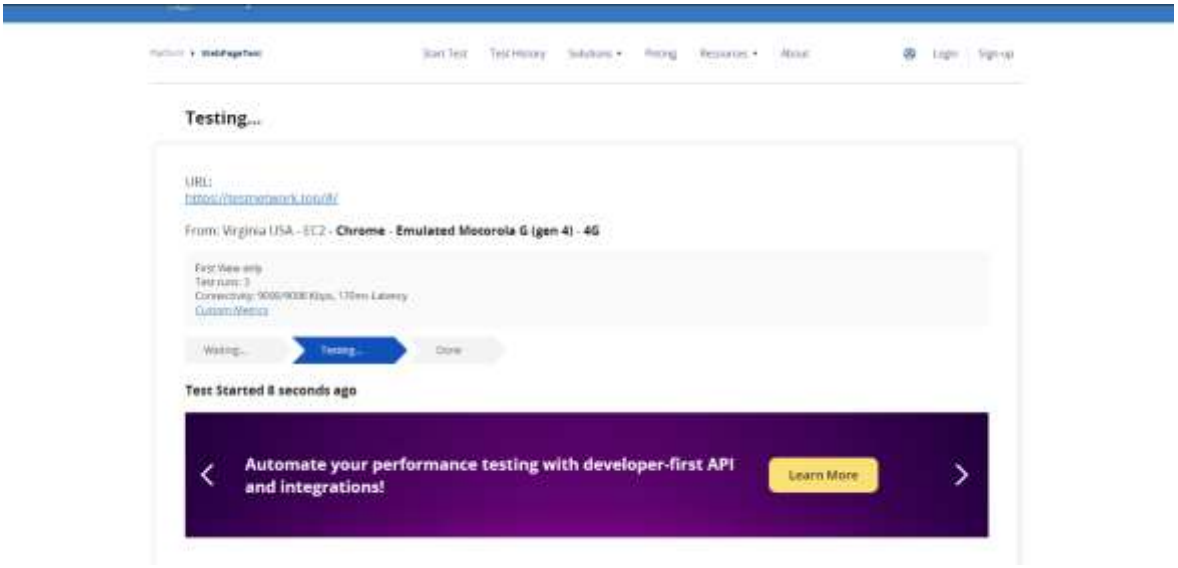

Рисунок 2.18 – Проведення тестування

Виявлені проблеми включали затримки завантаження сторінок і непередбачувану поведінку серверу при високому навантаженні. Було виправлено кілька вузьких місць у коді та оптимізовано базу даних, що значно покращило час відповіді сервера і загальну продуктивність сайту.

Навантажувальне тестування було невід'ємною частиною оцінки продуктивності, яке дозволяло визначити межі можливостей системи при збільшенні навантаження. Метою тестування було виявлення і усунення проблем, таких як затримки у відповіді, збої або нестабільність системи, після аналізу було зрозуміло в якому напрямку необхідно продовжувати працювати (див. рис. 2.19).

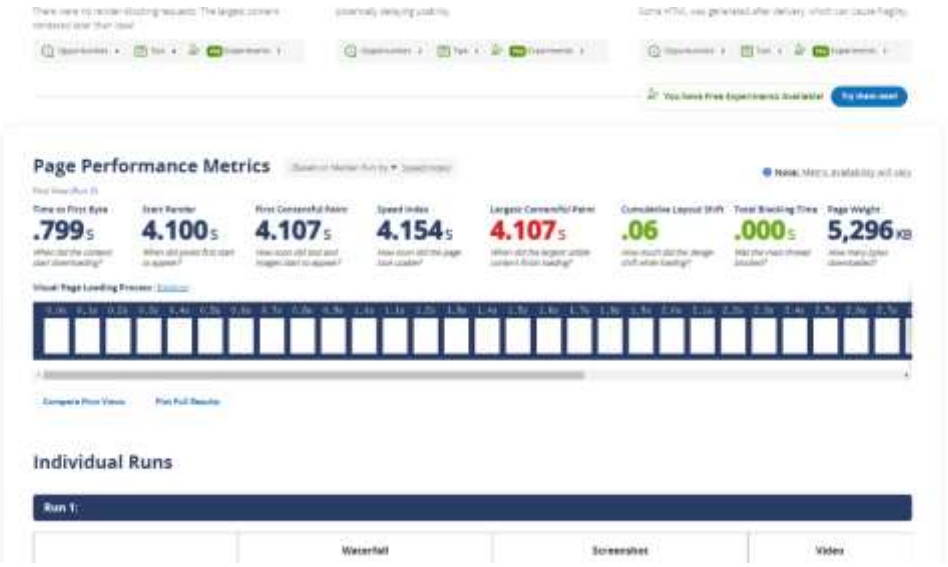

Рисунок 2.19 – Результат тестування навантаження

Інструмент аналізував час завантаження сторінок, відгук сервера та інші параметри продуктивності, що дозволило виявити слабкі місця в системі. Було прийнято рішення про оптимізацію налаштувань серверу та налаштування кешування, що забезпечило підвищення стабільності та швидкодії сайту при різних рівнях трафіку.

2.7.2 Використання віртуального марафону здорового способу життя

Веб-сайт віртуального марафону здорового способу життя надає користувачам інтерактивну платформу для участі у спортивних заходах та доступу до різноманітних тренувальних програм. Сайт розроблений з урахуванням потреб користувачів, що прагнуть вести здоровий спосіб життя, і забезпечує зручну навігацію через ключові функціональні сторінки, які роблять взаємодію з платформою простою та інтуїтивно зрозумілою.

Головна сторінка є візитною карткою сайту, яка зустрічає користувачів при вході. Вона оснащена яскравим дизайном і важливою інформацією про поточні події, новини та пропозиції. Основна мета цієї сторінки — зацікавити користувача та направити його до найбільш важливих розділів сайту. Тут можна знайти анонси майбутніх подій, короткий опис популярних тренувальних програм, а також посилання на новини, що пов’язані зі здоровим способом життя (див. рис. 2.20).

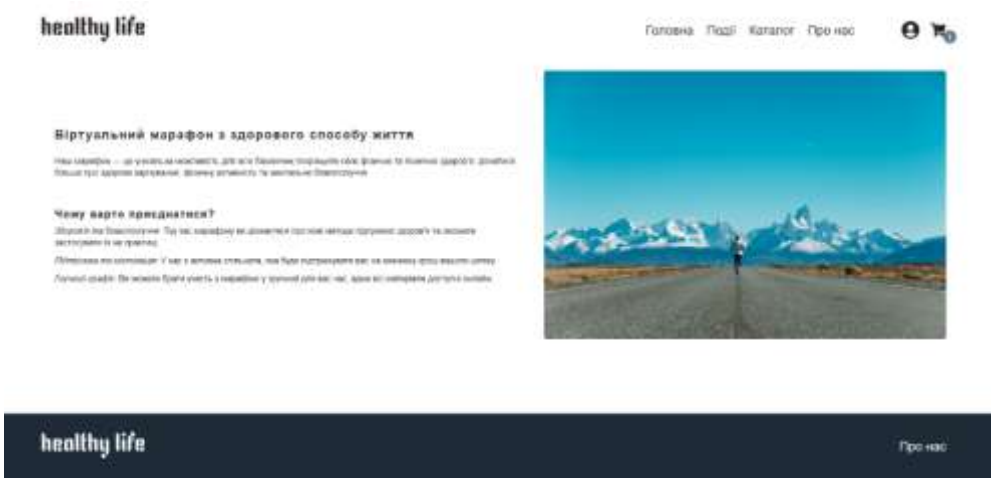

Рисунок 2.20 – Головна сторінка віртуального марафону

Сторінка подій є основним місцем для огляду та реєстрації на спортивні заходи. Користувачі можуть переглядати список доступних подій, які можуть включати марафони, змагання, тренування на свіжому повітрі та інші активності. Для кожної події надається докладна інформація, включаючи дату, час, місце проведення, а також опис заходу та вимоги до учасників (див. рис. 2.21).

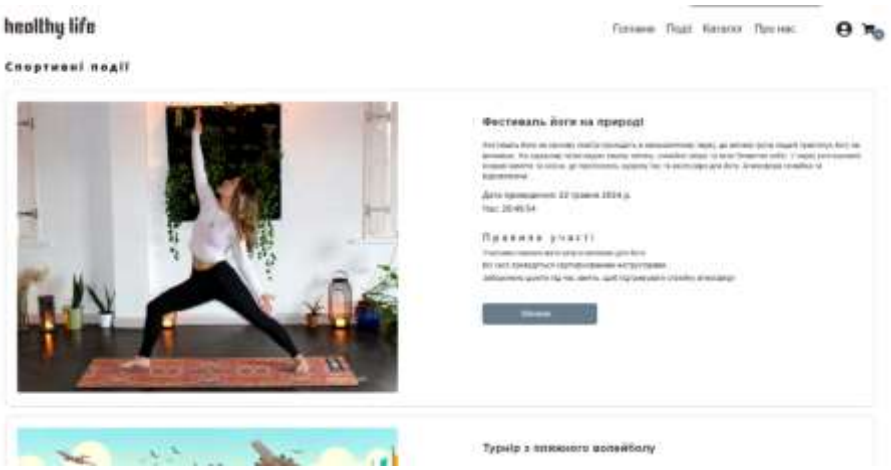

Рисунок 2.21 – Сторінка подій віртуального марафону

Каталог тренувальних програм надає користувачам доступ до широкого асортименту тренувань, розроблених для різних рівнів підготовки та цілей. Кожна програма представлена у вигляді окремої сторінки, де детально описані вправи та тривалість тренування. Користувачі можуть додати обрану програму до свого кошика, що полегшує планування тренувального процесу. (див. рис. 2.22).

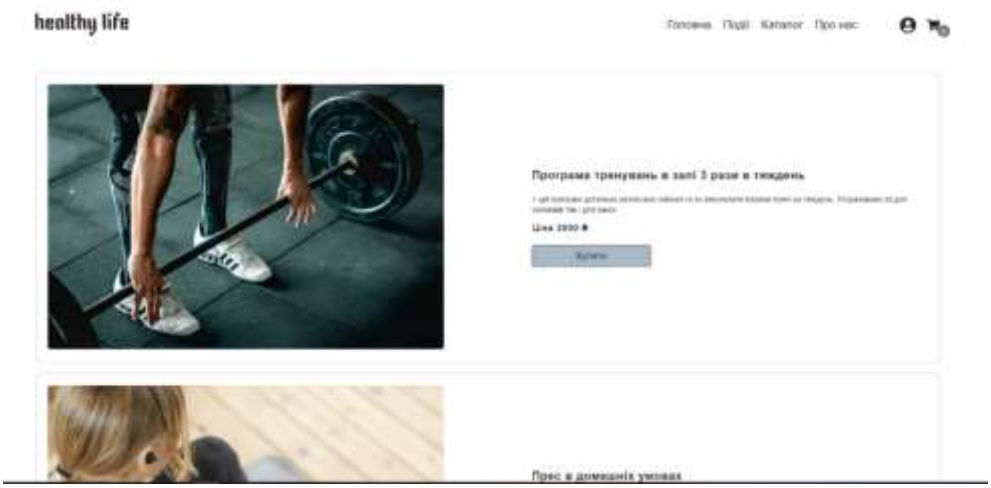

Рисунок 2.22 – Каталог тренування на сайті

Сторінка «Про нас» надає інформацію про місію, цінності та команду, що стоїть за створенням віртуального марафону. Цей розділ містить історію проекту, основні цілі та підходи до організації заходів і тренувальних програм. Користувачі можуть дізнатися більше про методологію, що використовується для розробки програм тренувань, а також про підтримку здорового способу життя та важливість фізичної активності (див. рис. 2.23).

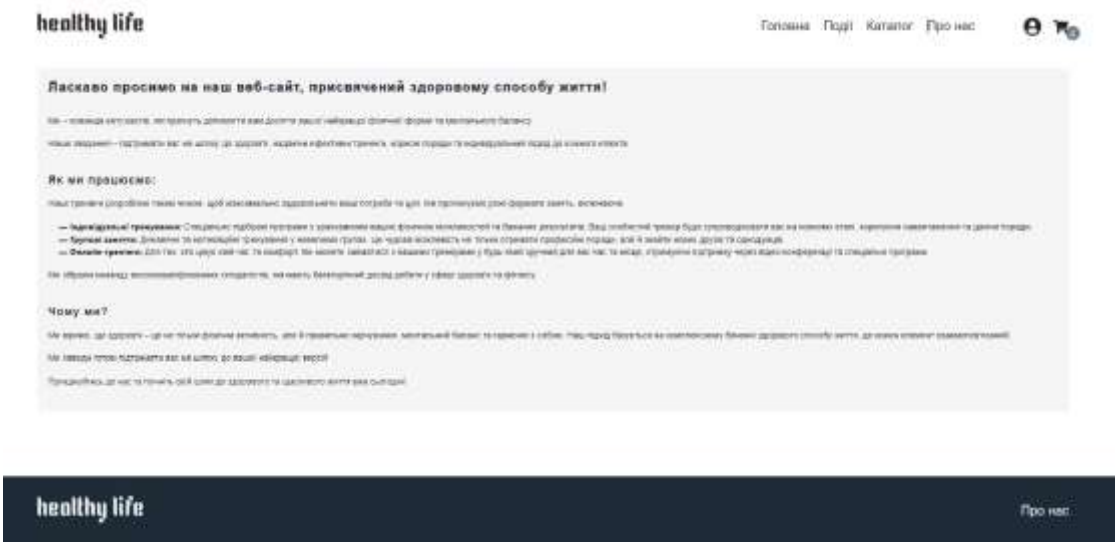

Рисунок 2.23 – Сторінка «Про нас»

Кошик є важливим компонентом сайту, який дозволяє користувачам збирати вибрані тренувальні програми та події для подальшого придбання або участі. Користувачі можуть легко переглядати свої вибрані елементи, редагувати їх кількість, видаляти непотрібні позиції та оформлювати замовлення. Іноді користувачі кладуть у них товари, але залишають сайт, не купивши нічого. Часто це пов’язано з тим, що оформлення кошика інтернет-магазину виконане неписьменно і люди мають труднощі, роблячи покупки. Кошик також відображає загальну вартість обраних програм і подій, що дозволяє користувачам мати чітке уявлення про фінансові витрати Кошик онлайн-магазину – це сторінка, на яку клієнт відправляє товари, щоб потім придбати їх. (див. рис. 2.24). [23]

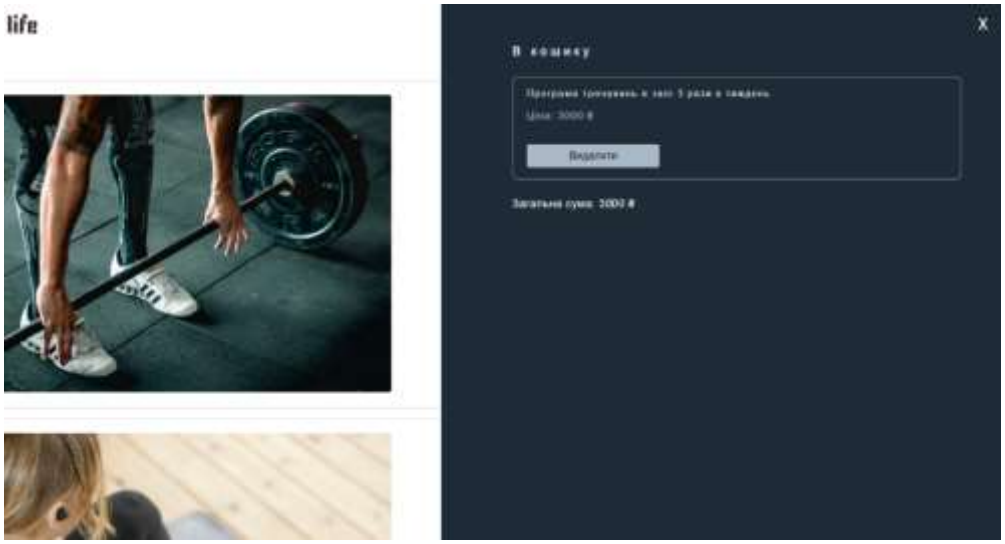

Рисунок 2.24 – Кошик

Для доступу до більшості функцій сайту необхідна реєстрація або вхід у свій кабінет. Сторінка реєстрації дозволяє користувачам створити новий обліковий запис, заповнивши форму з особистою інформацією та контактними даними. Після реєстрації або входу користувачі отримують доступ до свого персоналізованого кабінету, де можуть переглядати свою історію замовлень, керувати підписками на тренувальні програми, відстежувати прогрес у тренуваннях та змінювати особисті налаштування. Особистий кабінет в майбутньому буде дозволяти користувачам налаштовувати сповіщення про майбутні події та отримувати рекомендації щодо тренувань на основі їхніх вподобань та попередніх виборів (див. рис. 2.25 та 2.26).

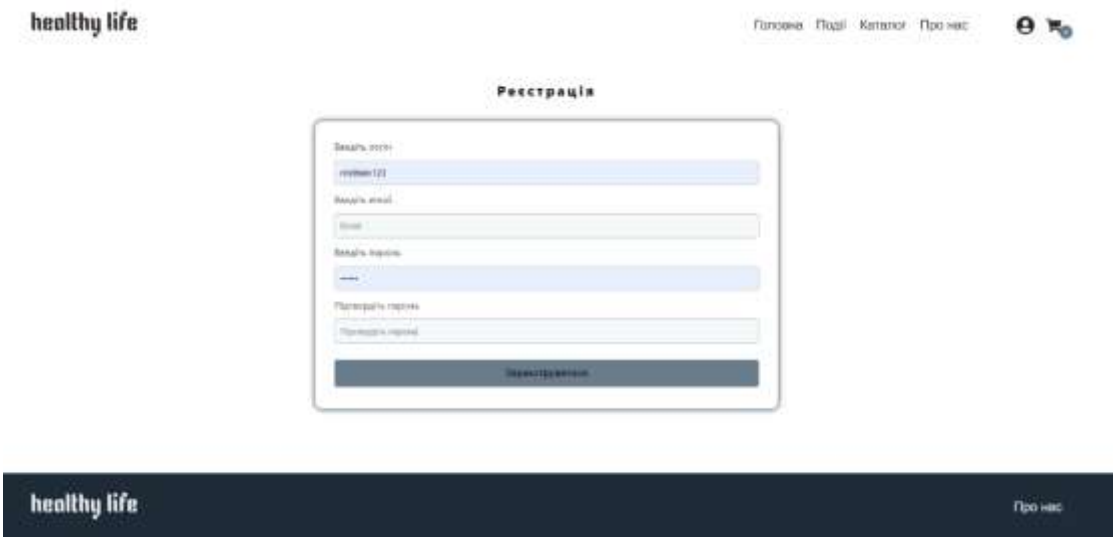

Рисунок 2.25 – Реєстрація на сайті

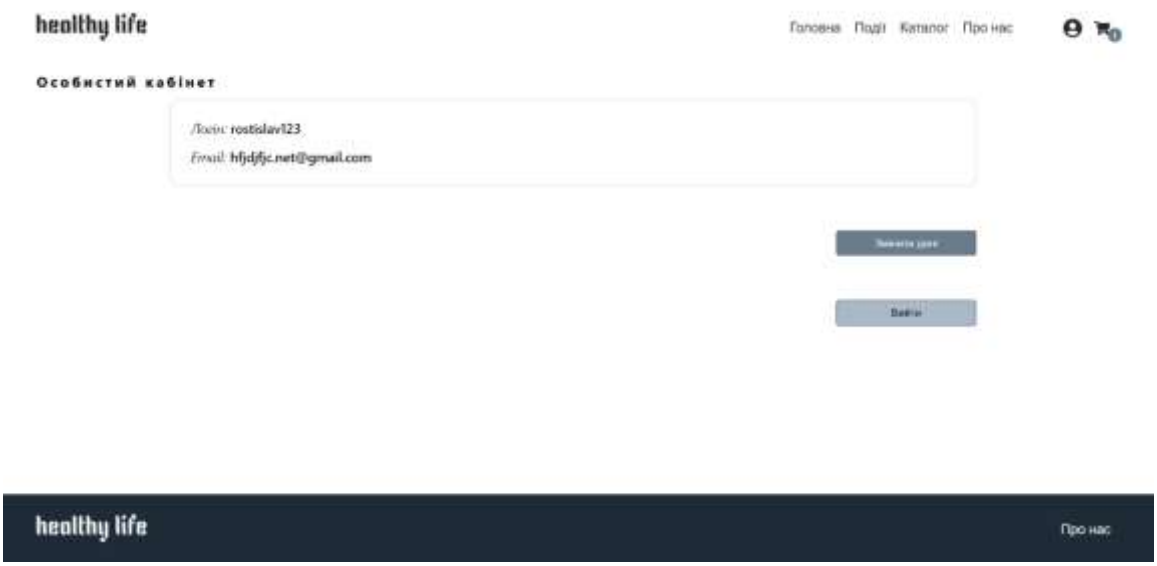

Рисунок 2.26 – Особистий кабінет

Веб-сайт віртуального марафону здорового способу життя створений з урахуванням потреб користувачів, надаючи зручний доступ до всіх необхідних функцій для участі в спортивних заходах і планування тренувального процесу. Завдяки інтуїтивно зрозумілому інтерфейсу, широкому асортименту програм і зручним інструментам управління користувачі можуть легко знаходити, обирати та долучатися до всіх аспектів здорового способу життя, що пропонує сайт.

2.8 Висновки до другого розділу

Проект віртуального марафону здорового способу життя є комплексною платформою, що об'єднує технологічні рішення для підтримки та стимулювання здорового способу життя серед користувачів. Розробка сайту була спрямована на створення зручного, інтерактивного та інформативного ресурсу, який допомагає користувачам знаходити, планувати та брати участь у спортивних подіях та тренуваннях. Проект об'єднав сучасні веб-технології, зручний інтерфейс користувача та надійні сервіси для забезпечення високої функціональності та ефективності роботи сайту.

Вибір технологій для проекту був обґрунтований необхідністю забезпечення надійності, масштабованості та продуктивності. Front-end частина була реалізована за допомогою Vue 3, що забезпечує високу реактивність та адаптивність інтерфейсу

користувача. Використання Vue 3 дозволило створити інтерактивний інтерфейс, який відповідає сучасним вимогам до веб-додатків. Для серверної частини був обраний Django, що гарантує безпеку, масштабованість та зручне управління даними. Django забезпечує надійну основу для обробки запитів, авторизації та аутентифікації, а також для інтеграції з базою даних.

База даних SQLite, яка використовувалася для зберігання даних проекту, забезпечує простоту налаштування та ефективність у малих і середніх проектах. Однак, вона була оптимізована для забезпечення стабільної роботи та високої продуктивності при обробці запитів. Під час тестування було виявлено кілька вузьких місць, які були виправлені для забезпечення максимальної ефективності обробки запитів та збереження даних.

Сайт був розроблений з акцентом на зручність користувача, що забезпечується інтуїтивно зрозумілим інтерфейсом і простими навігаційними рішеннями. Користувачі можуть легко знайти інформацію про події, тренувальні програми, а також управляти своїми замовленнями та підписками. Впроваджені функції дозволяють додавати тренувальні програми до кошика, реєструватися на події та переглядати свій прогрес у тренуваннях. Особлива увага була приділена адаптивності дизайну для різних типів пристроїв, що забезпечує зручний доступ до сайту як з комп'ютера, так і з мобільних пристроїв.

Проект пройшов комплексне тестування, включаючи функціональне тестування, тестування інтерфейсу користувача та тестування продуктивності. Всі виявлені помилки були оперативно виправлені, що забезпечило високу стабільність та надійність роботи сайту. Навантажувальне тестування за допомогою WebPageTest показало, що сайт здатний ефективно справлятися з великим обсягом запитів, що є критично важливим для забезпечення безперебійної роботи платформи під час пікових навантажень.

Розгортання проекту на платформі Digital Ocean забезпечило надійне хостинг середовище з високою доступністю та масштабованістю. Інструкції з розгортання були детально опрацьовані, що дозволило забезпечити швидке та ефективне налаштування серверного середовища. Підтримка та обслуговування сайту

здійснюються згідно з найкращими практиками, що гарантує його стабільність і безпеку.

На основі отриманих результатів та з урахуванням відгуків користувачів планується подальше вдосконалення функціоналу сайту. Передбачається розширення бібліотеки тренувальних програм, впровадження нових функцій для користувачів, таких як персоналізовані тренувальні плани та інтеграція з іншими сервісами для моніторингу здоров'я. Також планується розширення маркетингових можливостей для залучення нових користувачів та покращення взаємодії з існуючими.

3. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ХОРОНИ ПРАЦІ

3.1 Правила техніки безпеки при експлуатації обладнання

На підприємствах різного роду для обробки продуктів використовують електромеханічне і електронагрівальне обладнання. До експлуатації цього устаткування допускаються особи, навчені за програмою технічного мінімуму, що пройшли інструктаж на робочому місці та стажування. Вони повинні бути забезпечені інструкціями безпечної експлуатації обслуговуваного устаткування і ознайомлені під розписку з їх змістом.

Небезпека електромеханічного устаткування для обслуговуючого персоналу обумовлена наявністю в ньому електроприводу і робочих органів, які рухаються з великою швидкістю, а також і можливістю розлітання частинок оброблюваних сировини і продуктів.

Перед ввікненням машин в роботу необхідно перевіряти стан заземлення (занулення), цілісність ізоляції кабелю і елементів штепсельного роз'єму, надійність кріплення змінних механізмів, наявність і справність огорож, відсутність яких-небудь предметів у завантажувальних пристроях і робочих камерах.

Подачу продукту до робочого органу слід проводити, як правило, після випробування машини на холостому ходу. Для проштовхування продукту всередину бункера або робочої камери застосовують спеціальні пристосування (штовхачі, пестики, лопатки, шкрібники). Видалення продуктів, що заклинилися, або їх залишків здійснюється після повної зупинки двигуна і робочих органів машини. У разі виникнення стороннього шуму, підвищення температури поверхні електроприводу і кабелю живлення необхідно вимкнути машину. [15]

Після закінчення роботи машину вимикають кнопкою «Стоп» або перемикачем на її корпусі, вимикають з електричної мережі кнопкою «Стоп» електромагнітного пускача або витяганням вилки з розетки і очищають від залишків продуктів.

Для теплової обробки харчової сировини і продуктів, отримання гарячої води і приготування напоїв на підприємствах використовуютьелектронагрівальні апарати. Небезпека цих апаратів для обслуговуючого персоналу обумовлена наявністю

пристроїв, що знаходяться під напругою, і джерел високої температури, а деяких з них – ще й можливістю руйнування робочої камери надмірним тиском пари. У конструкції електронагрівальнихапаратів передбачені запобіжні захисні засоби для автоматичного вимкнення їх у разі відхилення параметрів від допустимих значень, а також сигнальні пристрої. Для видалення надлишків тепла, пари і газів деякі апарати мають мати місцеве вентиляційне відсмоктування. [15]

Перед ввімкненням будь-якого апарату в роботу необхідно перевірити наявність і стан запобіжних захисних засобів і сигнальних пристроїв, заземлення (занулення) корпусу, а також цілісність ізоляції кабелю і елементів штепсельного роз'єму. Для кожного виду електронагрівальних апаратів розроблені особливі вимоги безпеки під час експлуатації.

Експлуатація таких апаратів має бути припинена у разі відмови запобіжних захисних засобів і сигнальних пристроїв, неконтрольованого підвищення температури електронагрівників, а деяких з них – у разі пониження рівня води нижче допустимого значення, надмірного підвищення тиску пари.

Після закінчення роботи електронагрівальний апарат вимикають перемикачем, що є на його корпусі, і вимикають з електричної мережі. Потім видаляють з нього залишки води або продукту.

Технічне обслуговування і ремонт електромеханічного і електронагрівального устаткування проводять за договором механіки ремонтно-монтажних комбінатів, сервісних організацій.

Необхідно звернути увагу на наступних аспектах безпеки під час роботи з пристроями:

- До роботи на верстатах допускаються особи віком не молодше 18 років, які пройшли медичний огляд, навчання та інструктаж на робочому місці. Працівник повинен користуватися спецодягом і засобами індивідуального захисту, виконувати тільки ту роботу, за якою він проінструктований і яка доручена керівником робіт. На робочих місцях повинні бути відповідні інструкції з охорони праці під час роботи з інструментом, обладнанням і пристроями. Не дозволяється виконувати роботи на несправному обладнанні та використовувати обладнання та інструмент не за призначенням.

- Ремонтно-технологічне обладнання повинно бути забезпечено зручними в експлуатації запобіжними пристроями, що забезпечують добрий огляд і видимість виробу, що ремонтується (оброблюється) та захист очей. У випадку неможливості за технічними причинами використання запобіжного щитка власник повинен видати працівникам захисні окуляри.

- Для виконання постійних робіт пневматичним ударним інструментом повинно бути виділено спеціальне приміщення або окреме робоче місце, яке необхідно огородити переносними або стаціонарними звукопоглинаючими екранами. З метою запобігання вібраційній хворобі у працівників із механізованим (пневматичним) ручним інструментом необхідно застосовувати пневматичні молотки з пристроями для гасіння вібрації та видавати працівникам засоби індивідуального захисту рук від вібрації. Ручний пневматичний інструмент (молотки для клепання та рубання, свердлувальні та шліфувальні машинки тощо) повинен бути обладнаний ефективними глушителями шуму й викиду стисненого повітря.

- Пристрої, призначені для роботи під навантаженням (металеві підставки, домкрати тощо), слід щоденно оглядати перед початком роботи. Ручні важільно- рейкові домкрати повинні виключати самовільне опускання вантажу при знятті зусилля з важеля або рукоятки, забезпечуватися стопорами, що виключають вихід ґвинта або рейки при знаходженні штоку у верхньому крайньому положенні. Витікання рідини або повітря з робочих циліндрів домкратів або підйомників під час переміщення вантажів не допускється.

- Виготовлення, ремонт та заточування інструменту повинні проводитися централізовано спеціально навченим працівником. Використання нового або відремонтованого інструменту та пристроїв допускається тільки після випробування та приймання в експлуатацію.

- Для перенесення інструменту, якщо це потрібно за умовами роботи, кожному працівникові видається сумка або легкий переносний ящик. Для складання дрібних нарізаних заготовок повинна бути передбачена спеціальна тара, що забезпечує зручне транспортування і безпечне зачалювання при транспортуванні краном. Тара повинна бути прочною, розрахованою на необхідну вантажопідйомність, мати напис про максимально допустиме навантаження і періодично перевірятись та випробовуватися.

3.2 Розрахунок місцевої витяжної вентиляції для верстату чи обладнання

Оскільки система вентиляції охоплює спектр приміщень, які мають різне призначення, то розрахунок системи вентиляції, для кожного виду приміщень буде індивідуальним.

Розрахунок вентиляції виробничих приміщень, передбачає більшу увагу розрахунку по виділенню шкідливостей, а саме виділенню теплоти, вологи, шкідливих випарів. Вид розрахунку напряму залежить від технологічного процесу виконуваного під час виробництва продукції.

Розрахунок вентиляції згідно санітарних норм передбачається в більшості для громадських будівель. Такий розрахунок базується на умові забезпечення необхідною кількістю вентиляційного повітря, яке має податись за одну годину для однієї людини.

Нормативно встановлені такі значення повітрообміну згідно санітарних норм:

- 20 (м3/год*люд) – коли людина перебуває тимчасово у приміщенні;
- 40 (м3/год*люд) – при довготривалому перебуванні людини у приміщенні, яке вентилюється;
- 80 (м3/год*люд) – коли людина перебуває у стані фізичного навантаження, наприклад спортзал.

Розрізняють кілька методів розрахунку вентиляційних систем:

- за кратність повітрообміну
- за відношенню до санітарних норм
- за теплонадлишкам
- за вологонадлишкам
- за газовиділенням

висновок органів з охорони пам'яток архітектури про можливість проведення перебудови приміщення.

Щоб правильно виконати розрахунок витяжної вентиляції, слід визначити кількість шкідливостей, які виділяються в приміщенні, тобто асиміляції надлишку теплоти, вологи та розчинених у повітрі газів.

Розрахунок повітрообміну по кратності:

$$L = n * S * H \quad (3.1)$$

де L — необхідна продуктивність припливної вентиляції, м^3 /год.;

n — нормована кратність повітрообміну: для житлових приміщень n = 1, для офісів n=2,5

S — площа приміщення, м^2;

H — висота приміщення, м;

Розрахунок повітрообміну по кількості людей:

$$L = N * Lнорм \quad (3.2)$$

де L — необхідна продуктивність припливної вентиляції, м^3 /год.;

N — кількість людей;

Lнорм — норма витрати повітря на одну людину.

Для початку було розраховано значення повітрообмін по кількості людей:

$$L = 2 * 30 = 60 \text{ м}^3/\text{год} \quad (3.3)$$

Дальше було розраховано значення повітрообміну по кратності:

$$L = 1 * 10 * 3 = 30 \text{ м}^3/\text{год} \quad (3.4)$$

Отже мінімальні вимоги до повітрообміну дорівнюють: по кратності 30м3 /год, по кількості людей 60 м3/год .

3.3 Висновок до третього розділу

У розділі з охорони праці для проекту веб-сайту, що організовує віртуальний марафон здорового способу життя, було розглянуто два ключові питання: правила техніки безпеки при експлуатації обладнання та розрахунок місцевої витяжної вентиляції для верстату чи обладнання. Ці аспекти є важливими для забезпечення безпечної та здорової роботи всіх учасників і організаторів проекту.

при експлуатації обладнання є пріоритетом для забезпечення збереження здоров'я працівників та учасників віртуального марафону. У проекті були розроблені та впроваджені детальні правила техніки безпеки, які охоплюють всі аспекти роботи з обладнанням, що використовується для створення та підтримки веб-сайту, а також для проведення віртуальних заходів. Ці правила включають:

- Обов'язкові інструктажі та навчання з безпеки для всіх працівників, які працюють з обладнанням.
- Регулярні перевірки та технічне обслуговування обладнання для забезпечення його справності та безпечної експлуатації.
- Використання відповідних засобів індивідуального захисту (ЗІЗ) для працівників, що працюють з потенційно небезпечним обладнанням.
- Встановлення чітких процедур для дій у разі виникнення аварійних ситуацій.

Ці заходи спрямовані на мінімізацію ризиків травматизму та забезпечення безпечної роботи всіх учасників процесу.

Вентиляція є важливим аспектом для забезпечення комфортних та безпечних умов праці. В рамках проекту було проведено розрахунок місцевої витяжної вентиляції для обладнання, яке використовується у процесі створення контенту для веб-сайту та віртуального марафону.

Забезпечення охорони праці в проекті веб-сайту з віртуальним марафоном здорового способу життя базується на дотриманні вимог законодавства та впровадженні найкращих практик у цій сфері. Важливими елементами є:

- Систематичний підхід до ідентифікації та оцінки ризиків, пов'язаних з роботою на обладнанні та організацією віртуальних заходів.
- Регулярні інструктажі та навчання працівників з питань безпеки та охорони праці.
- Постійний моніторинг умов праці та впровадження заходів для покращення безпеки та комфорту.

Підтримка психологічного комфорту працівників є важливим аспектом для ефективної роботи команди. Створення сприятливої робочої атмосфери, що сприяє творчості та співпраці, є пріоритетом для нашої компанії. Це досягається через підтримку відкритого спілкування, організацію командних заходів та забезпечення умов для відпочинку та релаксації.

Компанія постійно працює над вдосконаленням системи охорони праці, впроваджуючи нові технології та методики для підвищення рівня безпеки. Використання сучасних засобів захисту, автоматизація процесів та регулярний моніторинг показників охорони праці дозволяють створити безпечне та ефективне робоче середовище.

У підсумку, розділ з охорони праці в проекті веб-сайту з віртуальним марафоном здорового способу життя відображає системний та комплексний підхід до створення безпечних умов праці. Всі заходи, що впроваджуються, спрямовані на підтримання фізичного та психологічного здоров'я працівників, підвищення їхньої продуктивності та задоволеності роботою, що є ключовими факторами для успішної реалізації проекту.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи освітнього рівня “бакалавр”, розробки віртуального марафону здорового способу життя засобами Django, Python, JavaScript, Vue3, HTML та CSS, було виконано всі раніше поставлені завдання при дослідженні задачі.

Результати дослідження теми було інтерпретовано у вигляді написання трьох розділів кваліфікаційної роботи, два з яких технічні що мають на меті реалізацію веб-сайту.

В першому розділі було проаналізовано предметну область, сформовано вимоги до розробки веб-сайту, проведено пошук актантів та можливі варіанти використання веб-сайту, детальніше описано ключові. Вибрано стек для реалізації.

В другому розділі описано архітектуру віртуального марафону здорового способу життя, детальніше про вибір технологій відповідно до стеку, файлову структуру веб-сайту, структурні особливості вибраної бази даних, розгортання на хостигових платформах, про тестування та використання розробленої системи.

В третьому розділі, було розглянуто правила техніки безпеки при експлуатації обладнання та розрахунок місцевої витяжної вентиляції для верстату чи обладнання.

В процесі роботи над кваліфікаційною роботою:

- освоєно новий теоретичний матеріал досліджуваної теми, заповнено пробіли у наявних знаннях;
- розглянуто сучасний інструментарій розробки веб-сайтів та вибрано підходящий;
- проведено порівняння між функціональними можливостями веб-сайту та функціональними можливостями конкурентів;
- на основі отриманої інформації веб-сайт було: розроблено та протестовано.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Розробка клієнт-серверного web-застосунку з використанням Python/Django та Angular. DSpace Repository :: Electronic Kyiv-Mohyla Academy Institutional Repository. URL: https://ekmair.ukma.edu.ua/items/4a2ea6f2-6e3c-40e2-88f9-1752da8c2673 (дата звернення: 28.06.2024).

2. Техніка безпеки під час експлуатації обладнання в ремонтних майстернях - Охорона праці і пожежна безпека. Охорона праці і пожежна безпека. URL: https://oppb.com.ua/news/tehnika-bezpeky-pid-chas-ekspluataciyi-obladnannya-v-remontnyh-maysternyah (дата звернення: 28.06.2024).

3. 3.12.4 documentation. 3.12.4 Documentation. URL: https://docs.python.org/3/ (дата звернення: 28.06.2024)

4. Contrib packages | Django documentation. Django Project. URL: https://docs.djangoproject.com/en/4.2/ref/contrib/ (дата звернення: 28.06.2024).

5. DigitalOcean | cloud infrastructure for developers. DigitalOcean | Cloud Infrastructure for Developers. URL: https://www.digitalocean.com/ (date of access: 28.06.2024).

6. Django | Создание моделей. METANIT.COM - Сайт о программировании. URL: https://metanit.com/python/django/5.1.php (дата звернення: 28.06.2024).

7. Django: developing web using python. IEEE Xplore. URL: https://ieeexplore.ieee.org/document/10183246 (дата звернення: 28.05.2024).

8. Getting started - vue.js. vue.js. URL: https://012.vuejs.org/guide/ (дата звернення: 28.05.2024).

9. Glass E., Camisso J. How to set up django with postgres, nginx, and gunicorn on ubuntu. DigitalOcean | Cloud Infrastructure for Developers. URL: https://www.digitalocean.com/community/tutorials/how-to-set-up-django-with-postgres-nginx-and-gunicorn-on-ubuntu-18-04 (date of access: 28.06.2024).

10. Microsoft. Documentation for visual studio code. Visual Studio Code - Code Editing. Redefined. URL: https://code.visualstudio.com/docs (дата звернення:

28.06.2024).

11. Models | Django documentation. Django Project. URL: https://docs.djangoproject.com/en/4.2/topics/db/models/ (дата звернення: 28.05.2024).

12. The Django admin site | Django documentation. Django Project. URL: https://docs.djangoproject.com/en/4.2/ref/contrib/admin/ (date of access: 28.06.2024).

13. VirtualMarathon. VirtualMarathon. URL: https://testnetwork.top/#/ (дата звернення: 28.06.2024).

14. Webpagetest. webpagetest. URL: https://www.webpagetest.org/ (date of access: 28.06.2024).

15. Website testing checklist and web application testing checklist | qawerk. QAwerk. URL: https://qawerk.com/blog/website-testing-checklist/ (date of access: 28.06.2024).

16. Fryz M., Mlynko B. Property Analysis of Conditional Linear Random Process as a Mathematical Model of Cyclostationary Signal // 2nd International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP 2022). Ternopil, Ukraine: CEUR Workshop Proceedings, 2022. Vol. 3309. P. 77–82.

17. Fryz M., Mlynko B. Determination of the characteristic function of discrete-time conditional linear random process and its application // Sci. J. TNTU. 2023. Vol. 109, № 1. P. 16–23.

18. Fryz M., Mlynko B. Property analysis of multivariate conditional linear random processes in the problems of mathematical modelling of signals // Technol. Audit Prod. Reserv. 2022. Vol. 3, № 2(65). P. 29–32.

19. Фриз М.Є., Млинко Б.Б. Умовні лінійні випадкові процеси з дискретним часом та їх властивості // Вісник Хмельницького національного університету. Серія: Технічні науки. 2022 (309), № 3. С. 7–12.

20. Fryz M., Mlynko B. Properties of Stationarity and Cyclostationarity of Conditional Linear Random Processes // 2020 IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering

(TCSET). Lviv-Slavske, Ukraine: IEEE, 2020. P. 166–170.

21. Fryz M., Scherbak L., Mlynko B., Mykhailovych T. Linear Random Process Model-Based EEG Classification Using Machine Learning Techniques // Proceedings of the 1st International Workshop on Computer Information Technologies in Industry 4.0 (CITI 2023). Ternopil, Ukraine: CEUR Workshop Proceedings, 2023. Vol. 3468. P. 126–132.

22. Бабак В. П., Марченко Б. Г., Фриз М. Є. Теорія ймовірностей, випадкові процеси та математична статистика. К.: Техніка, 2004. 288 с.

23. Как улучшить юзабилити интернет-магазина: 25 приемов с примерами. SEOQUICK. URL: https://seoquick.com.ua/online-store-usability (дата звернення: 28.06.2024).

24. Создание структуры интернет-магазина: облегчаем жизнь пользователю. Разработка интернет-магазинов под ключ – idbi. URL: https://idbi.ru/blogs/blog/sozdanie-struktury-internet-magazina (дата звернення: 28.06.2024).

25. Створення структури інтернет-магазину. URL: https://idbi.ru/blogs/blog/sozdanie-struktury-internet-magazina. (дата звернення: 28.06.2024).

26. Тестування продуктивності - QALight. QALight. URL: https://qalight.ua/baza-znaniy/testuvannya-produktivnosti/ (дата звернення: 28.06.2024).

27. Функції кошика в інтернет-магазині - Розробка та створення сайтів Сімферополь. Розробка та створення сайтів Сімферополь - Розробка сайтів, створення сайтів, розробка логотипів, дизайнер. URL: https://simferopil.rozrobka-sajtiv.in.ua/rozrobka-sajtiv-blog/funkcziyi-koshyka-v-internet-magazyni/ (дата звернення: 28.06.2024).

28. Функціональне тестування - QALight. QALight. URL: https://qalight.ua/baza-znaniy/funktsionalne-testuvannya/. (дата звернення: 28.06.2024).

29. Getting started | PyCharm. PyCharm Help. URL: https://www.jetbrains.com/help/pycharm/getting-started.html (дата звернення:

28.06.2024).

30. Github. URL: https://tutorial.djangogirls.org/django_admin/ (дата звернення: 28.06.2024).

31. Sport.ua. Фітнес і тренування вдома для початківців ⇒ вправи для новачків дому » ефективні вправи - sport.ua. СПОРТ.UA. URL: https://sport.ua/uk/zdorovyj-obraz-zhizni/548549-domashnie-trenirovki-dlya-nachinayushchih-uprazhneniya-osobennosti-i-plan (дата звернення: 28.06.2024).

32. Sweigart A. Recursive book of recursion: ace the coding interview with python and javascript. No Starch Press, Incorporated, 2022.

33. URL: http://molodyvcheny.in.ua/files/journal/2015/9/66.pdf (дата звернення: 28.06.2024).

34. URL: https://www.exlab.net/hostingdomain.html (дата звернення: 28.06.2024).

ДОДАТКИ

Додаток А

Лістинг 1 – Код файлу «settings.py»

```
"""
Django settings for virtual_maraton project.

Generated by 'django-admin startproject' using Django 5.0.6.

For more information on this file, see
https://docs.djangoproject.com/en/5.0/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/5.0/ref/settings/
"""

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# Quick-start development settings - unsuitable for production
#                                                          See
https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/
```

```
    # SECURITY WARNING: keep the secret key used in production secret!
    SECRET_KEY                    =                    'django-insecure-
0wk@#prm9igj9#&pm_a4*b8spxyj3sc8du8&9ka2*%&(ucc!if'


    # SECURITY WARNING: don't run with debug turned on in production!
    DEBUG = True


    ALLOWED_HOSTS = [
        '*'
    ]


    # Application definition


    INSTALLED_APPS = [
        'django.contrib.admin',
        'django.contrib.auth',
        'django.contrib.contenttypes',
        'django.contrib.sessions',
        'django.contrib.messages',
        'django.contrib.staticfiles',
        'drf_yasg',
        'events.apps.EventsConfig',
        'products.apps.ProductsConfig',
        'rest_framework',
        'rest_framework_simplejwt',
        'corsheaders',
    ]


    MIDDLEWARE = [
```

```
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
CORS_ORIGIN_ALLOW_ALL = True

ROOT_URLCONF = 'virtual_maraton.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates']
        ,
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

```
WSGI_APPLICATION = 'virtual_maraton.wsgi.application'


# Database
# https://docs.djangoproject.com/en/5.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


# Password validation
#   https://docs.djangoproject.com/en/5.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
                                                    'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
                                                    'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
                                                    'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
```

```
        {
                                                    'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
        },
    ]

    # Internationalization
    # https://docs.djangoproject.com/en/5.0/topics/i18n/

    LANGUAGE_CODE = 'en-us'

    TIME_ZONE = 'UTC'

    USE_I18N = True

    USE_TZ = True

    # Static files (CSS, JavaScript, Images)
    # https://docs.djangoproject.com/en/5.0/howto/static-files/

    STATIC_URL = 'static/'
    MEDIA_URL = 'media/'
    STATIC_ROOT = BASE_DIR / 'static'
    MEDIA_ROOT = BASE_DIR / 'media'

    # Default primary key field type
    #    https://docs.djangoproject.com/en/5.0/ref/settings/#default-auto-
field

    DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
       'rest_framework_simplejwt.authentication.JWTAuthentication',
    )
}
```

Додаток Б

Лістинг 2 – Код файлу «manage.py»

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
                os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'virtual_maraton.settings')
    try:
                        from    django.core.management    import
execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed
and "
            "available on your PYTHONPATH environment variable?
Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()
```

Додаток В

Лістинг 3 – Код файлу «urls.py»

```
"""
URL configuration for virtual_maraton project.

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/5.0/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path
from django.conf import settings
from django.conf.urls.static import static
from django.urls import include
from rest_framework import permissions
```

```
    from drf_yasg.views import get_schema_view
    from drf_yasg import openapi
    from events.views import EventsView
    from products.views import ProductsView
    from rest_framework_simplejwt.views import (
        TokenObtainPairView,
        TokenRefreshView,
    )
    from products.views import UserView, GetUser

    schema_view = get_schema_view(
       openapi.Info(
          title="Snippets API",
          default_version='v1',
          description="Test description",
          terms_of_service="https://www.google.com/policies/terms/
",
          contact=openapi.Contact(email="contact@snippets.local"),
          license=openapi.License(name="BSD License"),
       ),
       public=True,
       permission_classes=(permissions.AllowAny,),
    )

    urlpatterns = [
                path('swagger/',     schema_view.with_ui('swagger',
cache_timeout=0), name='schema-swagger-ui'),
                   path('redoc/',      schema_view.with_ui('redoc',
cache_timeout=0), name='schema-redoc'),
```

```
        path('api-auth/', include('rest_framework.urls')),
        path('users/', UserView.as_view()),
        path('api/user/', GetUser.as_view()),
        path('admin/', admin.site.urls),
        path('events/', EventsView.as_view()),
        path('products/', ProductsView.as_view()),

            path('api/token/',    TokenObtainPairView.as_view(),
name='token_obtain_pair'),
          path('api/token/refresh/',  TokenRefreshView.as_view(),
name='token_refresh'),

    ]                +                static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)
```

Додаток Г

Лістинг 4 – Код файлу «README.md»

```
# Form-Data [![NPM Module](https://img.shields.io/npm/v/form-data.svg)](https://www.npmjs.com/package/form-data) [![Join the chat at https://gitter.im/form-data/form-data](http://form-data.github.io/images/gitterbadge.svg)](https://gitter.im/form-data/form-data)


A library to create readable ```"multipart/form-data"``` streams. Can be used to submit forms and file uploads to other web applications.


The API of this library is inspired by the [XMLHttpRequest-2 FormData Interface][xhr2-fd].


[xhr2-fd]: http://dev.w3.org/2006/webapi/XMLHttpRequest-2/Overview.html#the-formdata-interface


[![Linux Build](https://img.shields.io/travis/form-data/form-data/v4.0.0.svg?label=linux:6.x-12.x)](https://travis-ci.org/form-data/form-data)

[![MacOS Build](https://img.shields.io/travis/form-data/form-data/v4.0.0.svg?label=macos:6.x-12.x)](https://travis-ci.org/form-data/form-data)

[![Windows Build](https://img.shields.io/travis/form-data/form-data/v4.0.0.svg?label=windows:6.x-12.x)](https://travis-ci.org/form-data/form-data)


[![Coverage Status](https://img.shields.io/coveralls/form-data/form-data/v4.0.0.svg?label=code+coverage)](https://coveralls.io/github/form-data/form-data?branch=master)
```

[![Dependency Status](https://img.shields.io/david/form-data/form-data.svg)](https://david-dm.org/form-data/form-data)

Install

```
npm install --save form-data
```

Usage

In this example we are constructing a form with 3 fields that contain a string,
a buffer and a file stream.

``` javascript
var FormData = require('form-data');
var fs = require('fs');

var form = new FormData();
form.append('my_field', 'my value');
form.append('my_buffer', new Buffer(10));
form.append('my_file', fs.createReadStream('/foo/bar.jpg'));
```

Also you can use http-response stream:

``` javascript
var FormData = require('form-data');
```

```
var http = require('http');

var form = new FormData();

http.request('http://nodejs.org/images/logo.png',
function(response) {
  form.append('my_field', 'my value');
  form.append('my_buffer', new Buffer(10));
  form.append('my_logo', response);
});
```

Or @mikeal's [request](https://github.com/request/request) stream:

```
``` javascript
var FormData = require('form-data');
var request = require('request');

var form = new FormData();

form.append('my_field', 'my value');
form.append('my_buffer', new Buffer(10));
form.append('my_logo',
request('http://nodejs.org/images/logo.png'));
```
```

In order to submit this form to a web application, call ```submit(url, [callback])``` method:

```` javascript
form.submit('http://example.org/', function(err, res) {
  // res – response object (http.IncomingMessage)  //
  res.resume();
});

```

For more advanced request manipulations ```submit()``` method returns ```http.ClientRequest``` object, or you can choose from one of the alternative submission methods.

### Custom options

You can provide custom options, such as `maxDataSize`:

``` javascript
var FormData = require('form-data');

var form = new FormData({ maxDataSize: 20971520 });
form.append('my_field', 'my value');
form.append('my_buffer', /* something big */);
```

List of available options could be found in [combined-stream](https://github.com/felixge/node-combined-stream/blob/master/lib/combined_stream.js#L7-L15)

### Alternative submission methods

You can use node's http client interface:

``` javascript
var http = require('http');

var request = http.request({
  method: 'post',
  host: 'example.org',
  path: '/upload',
  headers: form.getHeaders()
});

form.pipe(request);

request.on('response', function(res) {
  console.log(res.statusCode);
});
```

Or if you would prefer the `'Content-Length'` header to be set for you:

``` javascript
form.submit('example.org/upload', function(err, res) {
  console.log(res.statusCode);
});
```

To use custom headers and pre-known length in parts:

``` javascript
var CRLF = '\r\n';
var form = new FormData();

var options = {
  header: CRLF + '--' + form.getBoundary() + CRLF + 'X-Custom-Header: 123' + CRLF + CRLF,
  knownLength: 1
};

form.append('my_buffer', buffer, options);

form.submit('http://example.com/', function(err, res) {
  if (err) throw err;
  console.log('Done');
});
```

Form-Data can recognize and fetch all the required information from common types of streams (```fs.readStream```, ```http.response``` and ```mikeal's request```), for some other types of streams you'd need to provide "file"-related information manually:

``` javascript
someModule.stream(function(err, stdout, stderr) {
  if (err) throw err;
```

```
  var form = new FormData();

  form.append('file', stdout, {
    filename: 'unicycle.jpg', // ... or:
    filepath: 'photos/toys/unicycle.jpg',
    contentType: 'image/jpeg',
    knownLength: 19806
  });

  form.submit('http://example.com/', function(err, res) {
    if (err) throw err;
    console.log('Done');
  });
});
```

The `filepath` property overrides `filename` and may contain a relative path. This is typically used when uploading [multiple files from a directory](https://wicg.github.io/entries-api/#dom-htmlinputelement-webkitdirectory).

For edge cases, like POST request to URL with query string or to pass HTTP auth credentials, object can be passed to `form.submit()` as first parameter:

```javascript
form.submit({
  host: 'example.com',
  path: '/probably.php?extra=params',
```

```
  auth: 'username:password'
}, function(err, res) {
  console.log(res.statusCode);
});
```

In case you need to also send custom HTTP headers with the POST request, you can use the `headers` key in first parameter of `form.submit()`:

```javascript
form.submit({
  host: 'example.com',
  path: '/surelynot.php',
  headers: {'x-test-header': 'test-header-value'}
}, function(err, res) {
  console.log(res.statusCode);
});
```

### Methods

- [_Void_ append( **String** _field_, **Mixed** _value_ [, **Mixed** _options_] )](https://github.com/form-data/form-data#void-append-string-field-mixed-value--mixed-options-).
- [_Headers_ getHeaders( [**Headers** _userHeaders_] )](https://github.com/form-data/form-data#array-getheaders-array-userheaders-)
- [_String_ getBoundary()](https://github.com/form-data/form-data#string-getboundary)

- [_Void_ setBoundary()](https://github.com/form-data/form-data#void-setboundary)

- [_Buffer_ getBuffer()](https://github.com/form-data/form-data#buffer-getbuffer)

- [_Integer_ getLengthSync()](https://github.com/form-data/form-data#integer-getlengthsync)

- [_Integer_ getLength( **function** _callback_ )](https://github.com/form-data/form-data#integer-getlength-function-callback-)

- [_Boolean_ hasKnownLength()](https://github.com/form-data/form-data#boolean-hasknownlength)

- [_Request_ submit( _params_, **function** _callback_ )](https://github.com/form-data/form-data#request-submit-params-function-callback-)

- [_String_ toString()](https://github.com/form-data/form-data#string-tostring)

#### _Void_ append( **String** _field_, **Mixed** _value_ [, **Mixed** _options_] )

Append data to the form. You can submit about any format (string, integer, boolean, buffer, etc.). However, Arrays are not supported and need to be turned into strings by the user.

```javascript
var form = new FormData();
form.append( 'my_string', 'my value' );
form.append( 'my_integer', 1 );
form.append( 'my_boolean', true );
form.append( 'my_buffer', new Buffer(10) );
form.append( 'my_array_as_json', JSON.stringify( ['bird','cute'] ) )
```

You may provide a string for options, or an object.

```javascript
// Set filename by providing a string for options
form.append( 'my_file', fs.createReadStream('/foo/bar.jpg'), 'bar.jpg' );

// provide an object.
form.append( 'my_file', fs.createReadStream('/foo/bar.jpg'), {filename: 'bar.jpg', contentType: 'image/jpeg', knownLength: 19806} );
```

#### _Headers_ getHeaders( [**Headers** _userHeaders_] )

This method adds the correct `content-type` header to the provided array of `userHeaders`.

#### _String_ getBoundary()

Return the boundary of the formData. By default, the boundary consists of 26 `-` followed by 24 numbers

for example:

```javascript
--------------------------515890814546601021194782
```

#### _Void_ setBoundary(String _boundary_)

Set the boundary string, overriding the default behavior described above.

_Note: The boundary must be unique and may not appear in the data._

#### _Buffer_ getBuffer()

Return the full formdata request package, as a Buffer. You can insert this Buffer in e.g. Axios to send multipart data.

```javascript
var form = new FormData();
form.append( 'my_buffer', Buffer.from([0x4a,0x42,0x20,0x52,0x6f,0x63,0x6b,0x73]) );
form.append( 'my_file', fs.readFileSync('/foo/bar.jpg') );

axios.post( 'https://example.com/path/to/api',
            form.getBuffer(),
            form.getHeaders()
          )
```

**Note:** Because the output is of type Buffer, you can only append types that are accepted by Buffer: *string, Buffer, ArrayBuffer, Array, or Array-like Object*. A ReadStream for example will result in an error.

#### _Integer_ getLengthSync()

Same as `getLength` but synchronous.

_Note: getLengthSync __doesn't__ calculate streams length._

#### _Integer_ getLength( **function** _callback_ )

Returns the `Content-Length` async. The callback is used to handle errors and continue once the length has been calculated

```javascript
this.getLength(function(err, length) {
  if (err) {
    this._error(err);
    return;
  }

  // add content length
  request.setHeader('Content-Length', length);

  ...
}.bind(this));
```

#### _Boolean_ hasKnownLength()
Checks if the length of added values is known.

#### _Request_ submit( _params_, **function** _callback_ )
Submit the form to a web application.

```javascript
var form = new FormData();
form.append( 'my_string', 'Hello World' );

form.submit( 'http://example.com/', function(err, res) {
  // res – response object (http.IncomingMessage)  //
  res.resume();
} );
```

#### _String_ toString()

Returns the form data as a string. Don't use this if you are sending files or buffers, use `getBuffer()` instead.

### Integration with other libraries

#### Request

Form submission using [request](https://github.com/request/request):

```javascript
var formData = {
  my_field: 'my_value',
  my_file: fs.createReadStream(__dirname + '/unicycle.jpg'),
};

request.post({url:'http://service.com/upload', formData: formData}, function(err, httpResponse, body) {
  if (err) {
    return console.error('upload failed:', err);
  }
  console.log('Upload successful! Server responded with:', body);
});
```

For more details see [request readme](https://github.com/request/request#multipartform-data-multipart-form-uploads).

#### node-fetch

You can also submit a form using [node-fetch](https://github.com/bitinn/node-fetch):

```javascript
var form = new FormData();

form.append('a', 1);

fetch('http://example.com', { method: 'POST', body: form })
    .then(function(res) {
        return res.json();
    }).then(function(json) {
        console.log(json);
    });
```

#### axios

In Node.js you can post a file using [axios](https://github.com/axios/axios):

```javascript
const form = new FormData();
const stream = fs.createReadStream(PATH_TO_FILE);

form.append('image', stream);
```

```
// In Node.js environment you need to set boundary in the header field 'Content-Type' by calling method `getHeaders`
const formHeaders = form.getHeaders();

axios.post('http://example.com', form, {
  headers: {
    ...formHeaders,
  },
})
.then(response => response)
.catch(error => error)
```

## Notes

- ```getLengthSync()``` method DOESN'T calculate length for streams, use ```knownLength``` options as workaround.
- ```getLength(cb)``` will send an error as first parameter of callback if stream length cannot be calculated (e.g. send in custom streams w/o using ```knownLength```).
- ```submit``` will not add `content-length` if form length is unknown or not calculable.
- Starting version `2.x` FormData has dropped support for `node@0.10.x`.
- Starting version `3.x` FormData has dropped support for `node@4.x`.

## License

Form-Data is released under the [MIT](License) license.

**Додаток Д**

## Лістинг 5 – Код файлу «App.vue»

```
<script setup>

import HeaderComponent from "@/components/HeaderComponent.vue";
import FooterComponent from "@/components/FooterComponent.vue";

</script>

<template>
  <div class="app_wrapper">
    <HeaderComponent />

    <router-view class="content"/>

    <FooterComponent />
  </div>

</template>
<style scoped>
.app_wrapper {
  display: flex;
  flex-direction: column;
  min-height: 100vh;
}

.content {
  flex: 1 1 auto;
}
</style>
```

**Додаток Е**

**Лістинг 6 – Код файлу «main.js»**

```
import './assets/main.css'


import { createApp } from 'vue'
import App from './App.vue'
import {router} from './router'
import {store} from './store/store.js'


const app = createApp(App)



app.use(router)
app.use(store)


app.mount('#app')
```

**Додаток Є**

## Лістинг 7 – Код файлу «router.js»

```
import HomePage from "@/pages/HomePage.vue";
import * as VueRouter from "vue-router";
import EventsPage from "@/pages/EventsPage.vue";
import Catalog from "@/pages/Catalog.vue";
import AboutUsPage from "@/pages/AboutUsPage.vue";
import PersonalAccount from "@/pages/PersonalAccount.vue";
import UserRegistration from "@/pages/UserRegistration.vue";

const routes = [
    {path: "/", component: HomePage},
    {path: "/events", component: EventsPage},
    {path: "/catalog", component: Catalog},
    {path:"/about", component: AboutUsPage},
    {path:"/registration", component: UserRegistration, name: "registration"},
      {path:"/personal", component: PersonalAccount, name: "personal"},
]

export const router = VueRouter.createRouter({
    history: VueRouter.createWebHashHistory(),
    routes: routes,
})
```