МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

СЕМЕН ОЛЕКСАНДР ДМИТРОВИЧ

Допускається до захисту:
в.о. завідувача кафедри
інформаційних технологій
канд. техн. наук, доцент
_____ О. В. Зелінська
«____»_____ 2025 р.

ВЕБПЛАТФОРМА ДЛЯ ПРОВЕДЕННЯ СПОРТИВНИХ ЗМАГАНЬ

Спеціальність 122 Комп’ютерні науки

Кваліфікаційна (бакалаврська) робота

Керівник:
О. В. Зелінська, доцент кафедри
інформаційних технологій,
канд. техн. наук, доцент
_____

Оцінка: _____ / _____ / _____
(бали/за шкалою ЄКТС/за національною шкалою)
Голова ЕК: _____

Вінниця – 2025

АНОТАЦІЯ

Семен О.Д. Вебплатформа для проведення спортивних змагань. Спеціальність 122 «Комп’ютерні науки», освітня програма «Комп’ютерні науки». Донецький національний університет імені Василя Стуса, Вінниця 2025.

У кваліфікаційній (бакалаврській) роботі розроблено вебплатформу для проведення спортивних змагань. Реалізовано функціонал для створення подій, реєстрації учасників, відображення результатів змагань, підтримки військових. Під час виконання роботи були використані такі інструменти: Python, Flask, HTML, CSS, Bootstrap.

Ключові слова: вебплатформа, веброзробка, спортивні змагання, Python, Flask, Bootstrap.

67 ст., 33 рис., 42 джер.

ABSTRACT

Semen O.D. Web platform for sports competitions. Speciality 122 ‘Computer Science’, educational program ‘Computer Science’. Vasyl' Stus Donetsk National University, Vinnytsia 2024.

In the qualification (bachelor's) thesis, a web platform for sports competitions was developed. The functionality for creating events, registering participants, displaying competition results, and supporting the military has been implemented. Python, Flask, HTML, CSS, Bootstrap were used in the development.

Keywords: web platform, web development, sports competitions, Python, Flask, Bootstrap.

ЗМІСТ

ВСТУП

У сучасному світі особливої ваги набуває застосування вебтехнологій у сфері спорту, зокрема для організації та проведення спортивних змагань. Інструменти, які автоматизують процеси реєстрації учасників, планування змагань, підбиття результатів та взаємодії між усіма сторонами змагань, суттєво підвищують ефективність і прозорість цих процесів, зменшуючи вплив людського фактору. Тому розвиток засобів для автоматизації всіх цих процесів, є дуже актуальним.

В Україні за останні роки збільшилася кількість військовослужбовців, ветеранів, а також поранених, які проходять реабілітацію. Суспільство виявилося не готовим до такого, тому є потреба збільшувати кількість ініціатив, які сприятимуть підтримці військових. Залучення військових до спортивної активності – один із дієвих методів. Участь у спортивних заходах сприятиме соціальній адаптації, психологічній реабілітації та підтримці фізичної форми [1].

Тому створення вебплатформи, яка забезпечує не лише стандартні функції організації спортивних подій, але й орієнтована на підтримку військових, є актуальним та значущим завданням.

Мета цього дослідження полягає у створенні вебплатформи для проведення спортивних змагань, яка дозволить організовувати події, ставати їх учасником та створюватиме можливості участі для військових. Для досягнення мети, необхідно:

- проаналізувати існуючі аналоги вебплатформ для проведення спортивних змагань.
- побудувати архітектуру вебплатформи, обрати середовище розробки та інструменти для реалізації.
- розробити структуру та модель бази даних.
- реалізувати вебплатформу для проведення спортивних змагань.

Об’єктом дослідження є вебплатформа для проведення спортивних змагань. Предметом дослідження є науково-теоретичні основи, методичні

підходи та практичні аспекти створення вебплатформи для проведення спортивних змагань.

Практичне значення одержаних результатів полягає у створенні вебплатформи, яка може використовуватися для організації спортивних заходів, зокрема для військових та ветеранів.

Теоретичне значення полягає в дослідженні та узагальненні підходів до розробки вебзастосунків з використанням сучасних інструментів. У роботі розглянуто застосування мови програмування Python, а саме фреймворку Flask, який забезпечує гнучку розробку серверної частини платформи. Для створення клієнтської частини – HTML, CSS, а також фреймворк Bootstrap, який дозволяє використовувати готові шаблони для створення адаптивного, зручного та сучасного інтерфейсу користувача. Для збереження даних – SQLite, що є оптимальним рішенням для проєктів малого масштабу.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз вебплатформ для проведення спортивних змагань

При пошуку вебплатформ для проведення спортивних змагань, було виявлено проблему відсутності існуючих аналогів, які дозволять організувати змагання, з широким спектром вибору видів спорту, а також реєструватися учасникам на них. Більшість платформ передбачають або реєстрацію як учасника, або організацію змагань. Організація спортивних змагань на більшості платформ є недоступною для звичайного користувача, оскільки тільки затверджені організатори можуть створювати події, що унеможливлює проведення невеликих за масштабом змагань або громадських ініціатив. Тому в цьому розділі буде розглянуто платформи, які частково реалізують ці функції.

Run Ukraine – це провідна організація в Україні, яка спеціалізується на проведенні масових бігових змагань [2]. Основні функції платформи Run Ukraine:

- Професійна реєстрація учасників через спеціальну CRM-систему.
- Онлайн-оплата участі.
- Формування стартових протоколів та розподіл за віковими категоріями.
- Автоматичний підрахунок результатів через систему таймінгу.

Run Ukraine відзначається високою якістю організації, власною системою управління подіями, а також якістю реалізації та масштабом платформи. Серед недоліків можна виділити: Закритість платформи – звичайний користувач не може самостійно створити змагання та орієнтована виключно на бігові події.

Загалом, Run Ukraine – це приклад якісного сервісу, в якому зручно реалізована реєстрація учасника та оплата змагань. Проте його не можна розглядати як гнучку вебплатформу, оскільки він не надає можливості для організації спортивних подій невеликого масштабу, і має вузьке спрямування.

Challonge – вебплатформа, яка дозволяє створювати турніри у форматах одинарного вибування, подвійного вибування, кругової системи [3].

Переваги сервісу: Простий та зрозумілий інтерфейс; Широкий спектр підтримуваних видів спорту, включно з кіберспортом; Безкоштовний доступ до базових функцій.

Недоліки сервісу Challonge: Більше спеціалізується на кіберспортивних змаганнях; Вузька спрямованість – турнірна логіка, без екосистеми спортивної платформи.

Даний сервіс можна розглядати як гарний приклад реалізації функції створення та проведення змагань. Він простий, зрозумілий, та дозволяє швидко організувати змагання.

Отже, існуючі аналоги не пропонують одночасно і створення спортивних подій для усіх бажаючих, і реєстрацію учасників на них. Вони мають більш вузький функціонал, спрямований або на створення подій, або на участь в них. Це свідчить про те, що ніша вільна, і тема роботи актуальна.

1.2 Вимоги до сайтів для проведення спортивних змагань

Сайти, призначені для проведення спортивних змагань, повинні забезпечувати комплексну підтримку всіх етапів організації події: від планування до підбиття підсумків. Відповідно до аналізу існуючих рішень та особливостей спортивної сфери, можна виділити низку функціональних, технічних та нефункціональних вимог до таких вебплатформ.

Функціональні вимоги до сайту для проведення спортивних змагань:

Реєстрація користувачів. Платформа має забезпечувати можливість створення облікових записів для двох категорій користувачів: учасників та організаторів. Після реєстрації та успішної верифікації даних, користувач отримуватиме можливість користуватись особистим кабінетом.

Створення та управління подіями. Організатор повинен мати інструменти для зручного створення подій, з можливістю зазначення заголовку, опису, дати

проведення, а також завантаження афіші події. Крім того, потрібно передбачити можливість редагувати опубліковані дані, а також видаляти їх.

Онлайн реєстрація учасників. Для учасників має бути передбачена реєстраційна форма, де необхідно вказати особисті дані та дані для зв'язку.

Автоматичне формування турнірної сітки або розкладу.

Фіксація та публікація результатів. Організатори повинні мати доступ до інтерфейсу, через який вони вносять результати учасників змагань. Після затвердження результатів, вони мають відображатися у загальнодоступному розділі сайту.

Адміністративна панель для організаторів. Адміністративна панель має давати змогу переглядати актуальні змагання, переглядати списки учасників, містити інші функції для зручного управління змаганнями.

Зворотній зв'язок. Передбачено контактні дані для зв'язку користувачів з адміністраторами, для вирішення питань та пропозицій. Платформа може надсилати користувачам автоматичні повідомлення про події (скасування або перенесення події, оприлюднення результатів, тощо)

Технічні вимоги до сайту для проведення спортивних змагань:

Адаптивність інтерфейсу. Веб-платформа повинна коректно відображатися на різних типах пристроїв, зокрема на смартфонах та планшетах, забезпечуючи зручний користувацький досвід незалежно від розміру екрану. Інтерфейс має автоматично адаптуватися під роздільну здатність пристрою, зберігаючи при цьому повну функціональність усіх розділів системи.

Захист персональних даних. Усі персональні дані, що вводяться користувачами під час реєстрації або у процесі користування платформою, повинні бути надійно захищені. Передбачається використання сучасних методів шифрування. Обробка, зберігання та передача персональних даних мають здійснюватися відповідно до вимог чинного законодавства України або положень Загального регламенту захисту даних Європейського Союзу (GDPR), залежно від юрисдикції використання платформи.

Масштабованість. Архітектура веб-платформи має бути побудована з урахуванням можливості масштабування для підтримки великої кількості одночасних користувачів. Потрібно враховувати потенційне збільшення контенту, який буде розміщуватися на вебплатформі.

Регулярне резервне копіювання бази даних. З метою забезпечення збереження критично важливої інформації, система повинна підтримувати регулярне резервне копіювання бази даних. Резервні копії мають створюватися з певною періодичністю та зберігатися у безпечному місці з можливістю швидкого відновлення у разі виникнення збоїв або втрати даних.

Нефункціональні вимоги до сайту для проведення спортивних змагань:

Зручний інтерфейс користувача. Інтерфейс платформи повинен бути інтуїтивно зрозумілим і зручним у користуванні для всіх категорій користувачів. Має бути реалізована чітка і логічна структура сторінок, зрозуміла навігація, а також принципи UI/UX-дизайну, що підвищують зручність взаємодії користувача з системою.

Платформа повинна забезпечувати швидке завантаження контенту навіть за умов повільного інтернет-з’єднання. Цього можна досягти за допомогою стиснення зображень без втрати якості, застосування технологій кешування сторінок і скриптів, а також використання сучасних підходів до асинхронного завантаження контенту.

Безвідмовна робота у важливі моменти. Сайт має гарантувати стабільну роботу навіть у періоди пікового навантаження, наприклад, під час масових реєстрацій або в момент публікації результатів. Повинні бути передбачені механізми захисту від збоїв, аварійного відновлення, а також моніторинг системи в реальному часі.

Вебплатформа для проведення спортивних змагань повинна відповідати широкому спектру критеріїв якості. Вона має бути стабільною у роботі, інтуїтивно зрозумілою для користувачів, оптимізованою з технічної точки зору, а також функціонально насиченою. Особлива увага повинна приділятися забезпеченню високого рівня користувацького досвіду та дотриманню вимог

інформаційної безпеки. Система повинна ефективно реагувати на запити, адаптуватися до навантажень і забезпечувати безперервний доступ до критично важливих функцій у будь-який момент.

1.3 Постановка завдання

На основі проведеного аналізу існуючих рішень для організації спортивних змагань, а також із урахуванням визначених функціональних та технічних, було сформульовано завдання розробити вебплатформу, яка дозволятиме ефективно організовувати, адмініструвати та висвітлювати спортивні події. Розроблювана система повинна задовольняти потреби як звичайних учасників, так і організаторів змагань. Окрема увага приділяється підтримці соціально значущої мети – створенню умов для участі та реабілітації військовослужбовців через залучення до спортивної активності.

Мета створення вебплатформи – забезпечення зручного та функціонального інструменту для:

- Швидкої організації різноманітних спортивних подій.
- Реєстрації учасників в онлайн форматі.
- Керування перебігом змагань, обробки та публікації результатів.
- Поширення інформації серед зацікавленої аудиторії.

Основні функціональні вимоги до розробленої платформи:

Система ролей. Реалізація чіткої системи ролей: користувач при реєстрації обирає одну з доступних ролей – учасник або організатор. Функціональні можливості вебплатформи відрізняються залежно від ролі: організаторам надається розширений доступ до управлінських функцій (створення, редагування, перегляд учасників тощо), тоді як учасникам доступна взаємодія з подіями у межах своєї участі.

Перегляд подій та змагань. Незалежно від наявності облікового запису, кожен відвідувач має можливість переглядати всі опубліковані на платформі спортивні події. Для кожної події передбачається детальна інформація: дата,

місце проведення, опис, афіша тощо. Окремо передбачено категорію змагань для військовослужбовців, що дозволяє виділяти події з акцентом на соціальну реабілітацію та підтримку захисників.

Реєстрація на змагання. Участь у змаганнях можлива як для авторизованих, так і неавторизованих користувачів. Водночас, авторизовані учасники отримують додаткові переваги – зокрема, автоматичне заповнення даних при реєстрації, можливість переглядати статус участі, а також скасовувати свою реєстрацію в особистому кабінеті.

Кабінет організатора. Організатори отримують доступ до розширеної адміністративної панелі, де реалізується повний цикл керування подіями: створення нових змагань, редагування або видалення існуючих, перегляд списку зареєстрованих учасників, додавання та оновлення результатів змагань, а також можливість видалення некоректних або помилкових даних.

Кабінет учасника. Учасники мають змогу переглядати перелік змагань, на які вони зареєструвалися, та за потреби скасовувати участь. Це дозволяє користувачам більш гнучко планувати свою активність та відслідковувати історію взаємодії з платформою.

Структура сторінок вебплатформи. Платформа має включати декілька логічно структурованих сторінок:

- Головна сторінка – загальна інформація про платформу, останні події та новини.
- Сторінка змагань – каталог актуальних подій із можливістю фільтрації та пошуку.
- Сторінка для військових – спеціальний розділ з інформацією для військовослужбовців, з роз’ясненнями щодо участі у змаганнях.
- Інформаційна сторінка про платформу – сторінка з інформацією про цілі платформи.

Основні технічні вимоги до розробленої платформи:

Адаптивний інтерфейс. Графічний інтерфейс користувача має бути оптимізованим для коректного відображення на різних типах пристроїв –

комп’ютерах, планшетах і смартфонах. Це забезпечить зручність користування незалежно від технічних засобів доступу.

Система автентифікації. Реалізовано захищену систему входу з використанням шифрування та автентифікації. Доступ до функціоналу платформи розмежовується відповідно до ролі користувача.

Обробка форм та даних користувачів. Усі ключові дії такі як, реєстрація на змагання, створення подій, додавання результатів тощо, реалізовано через зручні веб-форми з валідацією даних. Передбачено перевірку обов’язкових полів, правильності форматів, наприклад, для дати та email, а також відображення повідомлень про помилки.

Захист від базових вразливостей. Має бути здійснено обробку помилок, додано перевірки правильності введених даних. Взаємодія з базою даних має бути реалізована таким чином, щоб унеможливити загрозу пошкодження даних.

Основні нефункціональні вимоги до розробленої платформи:

Зручність користування. Інтерфейс має бути логічно структурованим і зрозумілим для користувача. Навігація повинна бути інтуїтивною: користувачі легко знаходять потрібні розділи.

Продуктивність. Вебсторінки повинні завантажуватися швидко навіть при великій кількості одночасних користувачів. Для цього застосовуються сучасні підходи до оптимізації коду, зображень, а також кешування сторінок.

Мінімалістичний дизайн. Зовнішній вигляд інтерфейсу повинен залишатися візуально чистим та не перевантаженим елементами. Основна увага має бути зосереджена на функціоналі та контенті, що підвищує концентрацію користувача на основних діях без зайвих відволікань.

Таким чином, сформульоване технічне завдання охоплює як функціональні, так і нефункціональні вимоги до створення сучасної вебплатформи для проведення спортивних змагань. Особливу увагу приділено підтримці двох основних ролей користувачів – учасника та організатора, із чітким розмежуванням доступу до функцій відповідно до ролі. Важливими

складовими проєкту є реалізація зручних механізмів створення подій та реєстрації учасників, редагування змагань, внесення результатів.

Платформа повинна забезпечувати зручний та інтуїтивно зрозумілий інтерфейс, адаптивний дизайн, високу продуктивність. Водночас, соціальна складова розробки підкреслюється створенням окремої категорії змагань для військовослужбовців, що спрямована на їхню підтримку та реабілітацію через спортивну активність.

Усі вищезазначені вимоги стали основою для подальших етапів проєктування архітектури платформи, реалізації її функціоналу та проведення тестування для перевірки відповідності заданим критеріям якості.

РОЗДІЛ 2
ІНСТРУМЕНТИ ТА ТЕХНОЛОГІЇ ДЛЯ РЕАЛІЗАЦІЇ ВЕБПЛАТФОРМИ

2.1 Архітектура вебплатформи

Для реалізації поставленого завдання обрано трирівневу клієнт-серверну архітектуру, яка дозволяє чітко розділити систему на логічно незалежні компоненти: рівень представлення, рівень бізнес-логіки та рівень даних [4]. Така архітектурна модель забезпечує високий рівень масштабованості, модульності, зручності супроводу, а також дає змогу легко адаптувати або розширювати функціональність у майбутньому [5].

Рівень представлення відповідає за взаємодію кінцевого користувача з вебплатформою [6]. Його реалізовано у вигляді вебінтерфейсу, який доступний через будь-який сучасний веббраузер. Саме тут користувачі взаємодіють із системою, здійснюючи основні дії, серед яких:

- Перегляд переліку спортивних подій.
- Ознайомлення з деталями змагань.
- Реєстрація та авторизація.
- Подання заявки на участь у змаганнях.
- Створення та редагування подій, внесення результатів (для організаторів).

Інтерфейс реалізовано з використанням сучасних вебтехнологій – HTML, CSS, JavaScript – у поєднанні з відповідними фреймворками, які забезпечують динамічну побудову сторінок, адаптивний дизайн і зручну навігацію. Кожна сторінка має логічну структуру та забезпечує інтуїтивно зрозумілу взаємодію.

Основні функції рівня представлення включають:

Виведення даних. Відображення актуальної інформації про змагання, результати, повідомлення про дії користувача (наприклад, підтвердження реєстрації чи помилка введення).

Обробка користувацького вводу. Робота з формами, кнопками та іншими елементами інтерфейсу. Перевірка правильності введення даних.

Комунікація з сервером. Надсилання HTTP-запитів (GET, POST, PUT, DELETE) до серверної частини для обробки даних або виконання певних дій (реєстрація, авторизація, створення події тощо).

Відображення відповідей сервера. Отримані у форматі JSON або HTML відповіді обробляються у браузері та перетворюються на зрозуміле для користувача представлення.

Рівень логіки відповідає за опрацювання всіх внутрішніх процесів вебплатформи [7]. Саме на цьому рівні реалізується основна функціональність застосунку: обробка вхідних запитів від клієнтів, перевірка введених даних, керування правами доступу, взаємодія з базою даних, формування відповідей, а також виконання додаткових службових операцій.

Серверна частина платформи може бути реалізована за допомогою різних мов програмування та сучасних фреймворків. Серед найпоширеніших технологій можна виділити [8- 9]:

- Python: Flask, Django;
- JavaScript/TypeScript: Node.js, Express.js, NestJS;
- Java: Spring Boot;
- PHP: Laravel;
- C#: .NET Core;
- Go, Ruby, Rust – використовуються у випадках специфічних технічних вимог.

Основні функції, реалізовані на цьому рівні, включають:

Обробка вхідних запитів. Отримання HTTP-запитів від клієнта, їх маршрутизація до відповідних обробників, відповідно до структури REST-архітектури або іншої обраної архітектурної моделі.

Валідація даних. Перевірка правильності введених користувачем даних (наприклад, під час реєстрації або створення подій), а також перевірка автентичності користувача при авторизації.

Керування правами доступу. Розмежування прав відповідно до ролі користувача (учасник, організатор, адміністратор), обмеження або надання доступу до певних функцій платформи.

Взаємодія з базою даних. Зчитування, створення, оновлення та видалення даних, необхідних для роботи платформи.

Формування відповідей. Побудова відповідей для клієнтської частини у вигляді HTML-сторінок для подальшого відображення у браузері.

Цей рівень відіграє ключову роль у забезпеченні цілісної та стабільної роботи всієї системи, оскільки на ньому зосереджено виконання бізнес-процесів та логіки, необхідної для забезпечення потреб користувачів і досягнення функціональних цілей платформи.

Рівень даних відповідає за зберігання, обробку та управління інформацією, що використовується на всіх етапах роботи вебплатформи [10]. Саме на цьому рівні реалізується надійне збереження структурованих і неструктурованих даних, забезпечується їхня доступність, цілісність і актуальність для інших компонентів системи.

У якості основного сховища даних найчастіше використовуються реляційні системи управління базами даних, зокрема PostgreSQL, MySQL, SQLite [11]. Їхня популярність пояснюється високою надійністю, підтримкою складних транзакцій, широкими можливостями запитів та забезпеченням цілісності даних. У разі потреби в обробці великих обсягів неструктурованої інформації або забезпеченні максимальної швидкодії можуть застосовуватись NoSQL-рішення, зокрема документоорієнтовані бази даних – MongoDB, Redis тощо [12].

Доступ до бази даних може здійснюватися безпосередньо використання SQL-запитів, коли розробник формує запити до бази вручну або застосування ORM (Object-Relational Mapping) – об'єктно-реляційного відображення, яке дозволяє працювати з даними у вигляді об’єктів мови програмування [13]. Це суттєво спрощує розробку, підвищує читабельність коду та зменшує ризик помилок.

У структурі бази даних зберігаються такі користувачі: облікові записи користувачів, їхні ролі, зашифровані паролі, контактні та персональні дані; змагання: назви подій, їхній опис, місце і дата проведення, вид спорту, тип події; реєстрації на змагання: зв’язок між користувачем і подією, статус реєстрації; результати: оцінки виступів або місця, отримані учасниками, які вносяться організатором після завершення події.

Рівень даних забезпечує не лише фізичне збереження інформації, але й відповідає за підтримку зв’язків між сутностями, гарантії консистентності та унікальності записів, ефективне індексування і пошук даних, резервне копіювання для запобігання втраті критичної інформації.

Таким чином, трирівнева архітектура вебплатформи дозволяє ефективно розділити обов’язки між компонентами системи:

- Клієнтський рівень відповідає за інтерфейс взаємодії з користувачем.
- Серверна рівень реалізує бізнес-функціональність платформи.
- Рівень даних гарантує збереження, доступність та цілісність інформації.

Такий підхід дозволяє створити масштабовану, надійну та зручну у супроводі систему, яка задовольняє сучасні вимоги до вебзастосунків та має потенціал для подальшого розвитку.

2.2 Обґрунтування та вибір технологій для реалізації вебплатформи

Реалізація сучасної вебплатформи потребує ретельного вибору технологій, які забезпечать належний рівень функціональності, швидкодії, масштабованості, зручності розробки та подальшого супроводу. У процесі проєктування системи було обрано стек технологій, що відповідає вимогам до платформи: підтримка динамічної взаємодії з користувачем, простота інтеграції компонентів, а також відповідність сучасним стандартам веброзробки.

HTML – це стандартна мова розмітки, яка використовується для створення та структурування контенту вебсторінок [14]. HTML є основою будь-якого

вебресурсу, незалежно від його складності. Саме HTML визначає логіку відображення елементів інтерфейсу, структурує текстовий вміст, формує форми взаємодії з користувачем.

Особливу роль у сучасному HTML відіграють семантичні теги (<header>, <nav>, <main>, <section>, <footer>), які забезпечують логічну організацію сторінки, спрощують сприйняття структури як користувачем, так і програмними засобами – зокрема, пошуковими системами та програмами для людей з інвалідністю [15]. Це важливо з огляду на соціальну орієнтованість платформи – зокрема, надання зручного доступу до інформації військовослужбовцям, які можуть користуватися різними пристроями та браузерами.

HTML у цій вебплатформі виконує роль основного шару представлення. Він тісно інтегрується з такими технологіями, як:

- CSS – для стилізації та адаптивного оформлення інтерфейсу.
- JavaScript – для реалізації динамічної взаємодії з користувачем.

Наприклад, під час перегляду сторінки змагань користувач може обрати категорію «Для військових». У відповідь JavaScript надсилає запит на сервер, отримує список відповідних подій, після чого HTML-структура сторінки оновлюється без повного перезавантаження. Це покращує користувацький досвід, підвищує швидкість та зменшує навантаження на сервер.

HTML є критично важливим компонентом у побудові інтерфейсу вебплатформи. Завдяки своїй універсальності, сумісності з іншими технологіями та підтримці сучасних стандартів, він дозволяє створити зрозумілий, доступний та функціональний вебінтерфейс, орієнтований на широке коло користувачів. У контексті розробки системи для організації спортивних змагань, включаючи події для військових, HTML виступає як фундамент, на якому ґрунтується зручність, ефективність та доступність взаємодії з платформою.

CSS – це мова стилів, яка використовується для візуального оформлення вебсторінок [16]. У межах вебплатформи для проведення спортивних змагань CSS відіграє ключову роль у формуванні зовнішнього вигляду інтерфейсу, забезпечуючи зручність, естетичну привабливість і адаптивність ресурсу.

За допомогою CSS реалізується: кольорова схема та загальний стиль інтерфейсу, шрифти та їх параметри, положення елементів на сторінці, відступи, розміри та межі, адаптивність інтерфейсу до різних розмірів екранів, покращення доступності для різних категорій користувачів.

Для прискорення розробки інтерфейсу, зменшення кількості ручного кодування стилів і забезпечення консистентності оформлення, у платформі використовується Bootstrap – популярний фреймворк із відкритим кодом, який побудовано на основі CSS і JavaScript [17].

Основні переваги використання Bootstrap [18]:

Адаптивний дизайн. Bootstrap базується на гнучкій сітковій системі (grid system), яка дозволяє елементам інтерфейсу автоматично підлаштовуватись під розмір екрана користувача – від смартфонів до широкоформатних моніторів. Це критично важливо для доступності платформи з різних пристроїв.

Готові компоненти інтерфейсу. Bootstrap містить велику кількість заздалегідь оформлених елементів: кнопок, форм, вкладок, модальних вікон, повідомлень тощо. Їхнє використання значно прискорює створення функціональних сторінок.

Консистентний стиль. Усі елементи оформлюються за єдиними стандартами, що забезпечує візуальну узгодженість та професійний вигляд усіх сторінок платформи.

Застосування Bootstrap у розробці вебплатформи:

Структурування сторінок. За допомогою сіткової системи організовано розміщення елементів контенту на сторінках платформи, включаючи головну сторінку, сторінки змагань, профілі користувачів тощо. Гнучка розмітка дозволяє легко розташувати контент, при цьому зберігаючи зручність для користувачів на будь-яких пристроях.

Реалізація навігації. Верхня частина сайту буде побудована на компонентах Bootstrap, що дозволить створити зручну навігацію, яка буде коректно працювати як на широких екранах, так і на мобільних пристроях.

Форми реєстрації та входу. Стандартизовані форми забезпечують зручність заповнення, підказки, валідацію та зворотний зв’язок. Це сприяє зручній і швидкій взаємодії користувача із платформою.

Візуальні сповіщення та модальні вікна. За допомогою вбудованих компонентів можна реалізувати сповіщення про успішну реєстрацію, помилки введення, підтвердження дій. Модальні вікна використовуються для підтвердження важливих операцій, наприклад, скасування реєстрації на змагання або видалення події.

Підвищення користувацького досвіду. Bootstrap включає підтримку JavaScript-компонентів, які додають інтерфейсу плавність і реактивність, наприклад анімації кнопок. Також Компоненти Bootstrap розроблені з урахуванням стандартів доступності, що дозволяє користувачам з обмеженими можливостями комфортно працювати з платформою.

Використання CSS у поєднанні з фреймворком Bootstrap дозволяє створити адаптивний, зручний та візуально привабливий інтерфейс платформи. Це суттєво прискорює розробку, підвищує якість користувацького досвіду й забезпечує відповідність вебплатформи сучасним вимогам до вебресурсів, орієнтованих на широку аудиторію.

Python – це високорівнева інтерпретована мова програмування, яка широко використовується для створення вебдодатків завдяки своїй лаконічності, читабельності коду, широкому вибору бібліотек та активній спільноті розробників [19]. У межах розробки вебплатформи для проведення спортивних змагань Python виступає основою серверної частини системи, яка відповідає за обробку запитів, реалізацію бізнес-логіки та взаємодію з базою даних.

Основні переваги використання Python у веброзробці:

Простота та зрозумілий синтаксис. Python має інтуїтивно зрозумілу структуру, що значно пришвидшує написання коду, особливо при реалізації складної логіки. Завдяки цьому можливе швидке прототипування функціоналу платформи та гнучке внесення змін у майбутньому [20].

Широкий вибір фреймворків. Python підтримує такі потужні вебфреймворки, як Flask і Django. Flask – фреймворк, який дозволяє створити вебдодаток із мінімальними накладними витратами, зберігаючи гнучкість архітектури [21]. Django, натомість, забезпечує високий рівень абстракції та безліч вбудованих функцій [22]. У даному проєкті використовується Flask, оскільки він ідеально підходить для створення кастомізованих та масштабованих рішень з індивідуальною логікою.

Велика кількість бібліотек. Python має велику кількість бібліотек, наприклад, для роботи з HTTP-запитами – Flask, Requests, базами даних – SQLAlchemy, автентифікацією – Flask-Login, Flask-JWT, безпекою – Werkzeug, bcrypt, тестуванням – pytest, unittest. Це дозволяє гнучко формувати серверну логіку платформи.

Застосування Python у розробці вебплатформи для проведення спортивних змагань:

Обробка HTTP-запитів. Сервер, реалізований за допомогою Flask, приймає та обробляє запити від клієнта: перегляд сторінок, реєстрація та вхід, подача заявки на змагання, перегляд результатів тощо. На цьому рівні також реалізується маршрутизація URL-адрес до відповідних обробників функцій.

Реалізація бізнес-логіки. Python відповідатиме за основні функції платформи: керування користувачами і їх ролями, модерацію та фільтрацію подій, реєстрацію та скасування участі у змаганнях, збереження результатів, перевірку прав доступу.

Взаємодія з базою даних. Використання ORM-бібліотеки SQLAlchemy дозволяє взаємодіяти з базою даних на рівні об’єктів [23]. Це спрощує збереження, оновлення та отримання даних, а також зменшує ризик помилок при ручному написанні SQL-запитів. Python також підтримує виконання складних запитів за допомогою виразної мови запитів ORM.

Модульність та масштабованість. Уся серверна логіка організована у вигляді окремих модулів, що полегшує підтримку, рефакторинг і розширення

функціональності. Наприклад, у майбутньому легко можна додати нові типи змагань, інтеграцію з API для карт або платіжних систем.

Зручність розробки та підтримки. Python дозволяє швидко створювати робочі прототипи та тестові версії вебплатформи, які можна перевіряти й адаптувати відповідно до вимог користувачів. Завдяки активній спільноті та великій кількості документації, Python забезпечує ефективну розробку та оперативне вирішення технічних проблем.

Використання Python як основної мови програмування серверної частини вебплатформи для проведення спортивних змагань забезпечує надійність, гнучкість та масштабованість розв'язку. Python дозволяє ефективно реалізувати бізнес-логіку системи, інтегруватися з базою даних, керувати користувачами, забезпечити безпеку даних і підтримувати подальший розвиток платформи. Поєднання простоти, потужного інструментарію та активної екосистеми робить Python оптимальним вибором для створення сучасних вебдодатків соціального спрямування.

Flask – це легкий і потужний вебфреймворк, який надає розробнику мінімалістичну, але розширювану архітектуру для побудови вебдодатків будь-якої складності. Його популярність пояснюється гнучкістю, простотою використання, високим рівнем контролю над компонентами застосунку та широким вибором розширень.

У рамках реалізації вебплатформи для проведення спортивних змагань Flask використовується як основа серверної частини. Завдяки своїй модульності й простій структурі цей фреймворк чудово підходить для реалізації бізнес-логіки та організації взаємодії з базою даних.

Основні переваги Flask [24]:

Мінімалістична архітектура. Flask забезпечує лише базовий набір інструментів, наприклад маршрутизація, запуск сервера, обробка запитів, які дозволяють розробнику самому вибирати структуру застосунку та інтегрувати тільки ті компоненти, які дійсно необхідні. Такий підхід сприяє створенню легкої, ефективної та зрозумілої серверної логіки.

Гнучкість і масштабованість. Flask дозволяє без труднощів додавати нові маршрути, функції та логіку – як у процесі первинної розробки, так і при подальшому розширенні платформи. Зміни в архітектурі не потребують значних переробок, що є критично важливим для гнучкої розробки кваліфікаційного проєкту.

Активна спільнота і велика кількість розширень. Існує велика кількість готових рішень, які легко інтегруються з Flask. Для роботи з базами даних – Flask-SQLAlchemy, автентифікації – Flask-Login, форм – Flask-WTF, логування, кешування, захисту від CSRF-атак тощо. Це значно спрощує реалізацію функцій без необхідності створювати їх з нуля. Активна спільнота допомагає знайти відповіді на питання, які виникають в ході роботи з Flask.

Роль Flask у вебплатформі для спортивних змагань:

Обробка HTTP-запитів. Flask отримує запити, надіслані з клієнтського інтерфейсу, та визначає відповідну реакцію – виклик функції обробника, обробку даних, генерацію HTML-сторінки або JSON-відповіді.

Маршрутизація. За допомогою декораторів Flask дозволяє встановити зв’язок між URL-шляхом і конкретною функцією, яка буде викликана при зверненні до цього шляху. Такий механізм є основою організації навігації в системі.

Взаємодія з базою даних. Flask легко інтегрується з ORM-бібліотекою SQLAlchemy, яка забезпечує зручну роботу з об’єктно-реляційною моделлю. Усі основні сутності платформи реалізуються у вигляді Python-класів, які автоматично відображаються на відповідні таблиці в базі даних.

Обробка форм. Flask спільно з розширенням Flask-WTF дозволяє ефективно створювати та валідувати форми для таких дій, як реєстрація користувача, створення події, редагування профілю тощо. Валідація здійснюється як на стороні сервера, так і на клієнтському рівні за допомогою HTML5 і JavaScript.

Безпека доступу. Flask підтримує впровадження механізмів аутентифікації та авторизації, що регулює права доступу користувачів до різних функцій платформи залежно від їх ролі.

Шаблонізація інтерфейсу. Flask використовує шаблонізатор Jinja2, який дає змогу вбудовувати змінні та логіку безпосередньо в HTML-сторінки. Завдяки цьому інтерфейс формується динамічно відповідно до даних із бази або результатів дій користувача.

Обробка помилок і логування. Flask забезпечує централізовану обробку винятків. Наприклад, при спробі доступу до неіснуючого змагання користувач отримає повідомлення про помилку 404. Крім того, сервер логуватиме всі виняткові ситуації, що спрощує діагностику проблем під час розробки та в експлуатації.

Фреймворк Flask є оптимальним вибором для реалізації серверної частини вебплатформи для проведення спортивних змагань. Його гнучка, модульна архітектура, підтримка розширень, потужна система маршрутизації та шаблонізації дають змогу створити повноцінний вебдодаток із чистою та підтримуваною структурою коду. Flask забезпечує контроль над кожним аспектом логіки платформи, даючи розробнику змогу реалізувати як базову функціональність, так і складніші механізми. Таким чином, Flask забезпечує надійний технологічний фундамент для вебсистеми, орієнтованої на масштабування, ефективність і зручність для користувачів.

SQLite – це легка, вбудована реляційна система керування базами даних, яка не потребує окремого серверного програмного забезпечення [25]. Вона відома своєю простотою інтеграції, мінімальними вимогами до ресурсів та високою швидкістю доступу до даних у межах локального середовища. Завдяки своїм характеристикам SQLite широко застосовується в мобільних застосунках, десктопних програмах, а також у невеликих або середніх вебпроєктах, особливо на етапах розробки та тестування.

На відміну від класичних серверних СУБД, таких як MySQL або PostgreSQL, SQLite не вимагає запуску окремого сервера. Всі дані зберігаються

у звичайному файлі з розширенням .db, який читається і записується напряму через файл-систему. Такий підхід забезпечує простоту в налаштуванні та переносимості, що робить SQLite ідеальним вибором для дипломного проєкту.

Основні переваги SQLite [26]:

Простота використання. Для початку роботи не потрібно встановлювати серверну інфраструктуру, створювати користувачів або налаштовувати мережеві підключення – достатньо вказати шлях до файлу бази даних і підключити його до застосунку.

Швидкодія. SQLite працює напряму з локальним файлом, що забезпечує дуже швидкий доступ до даних у невеликих системах, де обсяг даних і кількість одночасних з'єднань є обмеженими.

Автономність і портативність. Усі дані, включно зі схемою таблиць і самими записами, зберігаються в одному файлі. Це значно спрощує процес резервного копіювання, перенесення на інший комп'ютер або хостинг, а також дозволяє зберігати кілька версій бази даних для різних середовищ.

Надійність. SQLite відповідає стандарту ACID, що забезпечує збереження цілісності даних навіть у випадках збою застосунку або системи.

У рамках розробки вебплатформи для проведення спортивних змагань SQLite використовується як основна база даних, яка виконує такі функції:

- Зберігання структурованих даних. У базі зберігається інформація про користувачів, спортивні події, категорії змагань, заявки на участь, результати, а також ролі й права доступу.
- Інтеграція з ORM. SQLite чудово працює у зв'язці з ORM-бібліотекою SQLAlchemy. Це дозволяє працювати з базою даних через Python-код замість написання складних SQL-запитів, що пришвидшує розробку та зменшує кількість помилок.
- Швидкий старт і розгортання. Через відсутність потреби в налаштуванні серверної СУБД платформа може бути запущена миттєво. Достатньо створити файл бази даних і розмістити його в кореневій директорії застосунку.

- Відповідність обсягу проєкту. Оскільки вебплатформа орієнтована на обмежене коло користувачів, навантаження на базу даних буде відносно невеликим. SQLite легко впорається з обробкою запитів у такому середовищі.
- Перспективи розширення. SQLite використовується на початковому етапі, але при зростанні кількості користувачів чи у разі переходу на продуктивне середовище її можна замінити на більш потужну СУБД, наприклад PostgreSQL, завдяки використанню ORM-рівня абстракції, що не потребує суттєвих змін у коді застосунку.

SQLite є оптимальним вибором для реалізації бази даних вебплатформи на даному етапі. Її простота, автономність, швидкість і повна інтеграція з інструментами Flask забезпечують зручне середовище для зберігання та обробки даних користувачів і змагань. У контексті даного проєкту вона дозволяє зосередитися на реалізації бізнес-логіки платформи, не витрачаючи ресурси на адміністрування складної серверної СУБД. Такий підхід повністю відповідає вимогам проєкту як із точки зору функціональності, так і ефективності розробки.

Visual Studio Code – це безкоштовне, кросплатформне інтегроване середовище розробки, створене компанією Microsoft. Воно є одним із найпопулярніших інструментів серед розробників завдяки своїй легкості, гнучкості, широкій підтримці мов програмування та багатому набору функціональних розширень [27]. У контексті реалізації вебплатформи для проведення спортивних змагань, VS Code виступає центральним інструментом, що забезпечує повний цикл розробки: від створення HTML-шаблонів до програмування серверної логіки на Python та налаштування бази даних.

Основні переваги використання Visual Studio Code [28]:

Підтримка розширень. Доступна велика кількість розширень, які легко встановлюються. Для проєкту використовуються такі розширення:

- Python – забезпечує підтримку синтаксису, автодоповнення, запуску коду та інтеграцію з середовищем Python.
- Flask Snippets – полегшує написання типових конструкцій коду Flask.

- Live Server – забезпечує миттєвий перегляд змін в інтерфейсі користувача при редагуванні HTML/CSS.
- Bootstrap 5 Snippets – прискорює верстку за допомогою готових шаблонів компонентів Bootstrap.

Інтегрований термінал. Вбудований термінал дає змогу запускати Flask-сервер, створювати або мігрувати базу даних, встановлювати залежності з pip, працювати з віртуальним середовищем, виконувати команди git, не покидаючи середовища розробки. Це значно пришвидшує розробку та зменшує кількість помилок, пов'язаних із перемиканням між різними інструментами.

Управління проєктною структурою. VS Code містить панель, у якій відображається дерево файлів, що дозволяє швидко орієнтуватися у структурі проєкту. Можна легко створювати та впорядковувати папки для шаблонів, стилів, серверної логіки, бази даних, зображень тощо. Це важливо для підтримки чистоти та логіки у великих проєктах.

Підсвічування синтаксису та автодоповнення. VS Code забезпечує високий рівень підтримки синтаксису Python, HTML, CSS, JavaScript, що полегшує написання коду та зменшує кількість помилок.

Підтримка Git. Інтеграція з Git дозволяє керувати версіями проєкту, комітити зміни, створювати гілки та зливати їх – усе безпосередньо з редактора. Це корисно для збереження проміжних версій дипломного проєкту.

При розробці вебплатформи VS Code використовується як єдине середовище розробки для всіх компонентів застосунку:

Розробка інтерфейсу користувача. Через підтримку Live Server та Bootstrap Snippets швидко створюється і перевіряється візуальна частина платформи.

Програмування серверної логіки. Модулі на Flask, взаємодія з базою даних, реалізація маршрутів і обробка HTTP-запитів виконуються у Python-скриптах із повною підтримкою форматування та відлагодження.

Робота з базою даних SQLite. Через відповідні розширення можна переглядати вміст бази даних, перевіряти, чи коректно зберігається інформація про події, користувачів, заявки та результати.

Тестування і налагодження. Завдяки можливості запуску серверу безпосередньо з редактора та використанню інтегрованого дебагера, можна швидко знаходити і виправляти помилки в коді.

Visual Studio Code – ефективний інструмент для реалізації повного циклу розробки вебплатформи: від дизайну інтерфейсу до реалізації складної серверної логіки на Python із використанням Flask та SQLite. Його багатофункціональність, гнучкість і підтримка численних розширень дозволяє сконцентруватися на реалізації функціоналу, а не на технічному налаштуванні середовища. Завдяки своїй функціональності та зручності, VS Code забезпечує комфортне середовище для швидкої та якісної розробки дипломного проєкту.

2.3 Проєктування бази даних для сайту

Для забезпечення стабільної та зручної роботи вебплатформи для проведення спортивних змагань потрібно спроектувати реляційну базу даних, структура якої охоплює всі необхідні аспекти функціонування системи – від управління користувачами до збереження реєстрацій і результатів подій. В ході проектування мають бути враховані такі вимоги:

Підтримка багаторівневої взаємодії між користувачами. У системі передбачено різні ролі користувачів, що визначають рівень доступу до функціоналу платформи.

Збереження інформації про кожне змагання. Необхідно зберігати назву, опис, дату, зображення події та організатора.

Реєстрація на події. Мають зберігатися данні про участь користувача у змаганнях, не залежно від того чи авторизований він.

Збереження результатів змагань. Має бути можливість фіксації результатів за різними типами показників.

Розпочати варто з таблиці для зберігання даних користувачів, її назва User. Таблиця міститиме інформацію про користувачів, незалежно від їхньої ролі, (рис. 2.1).

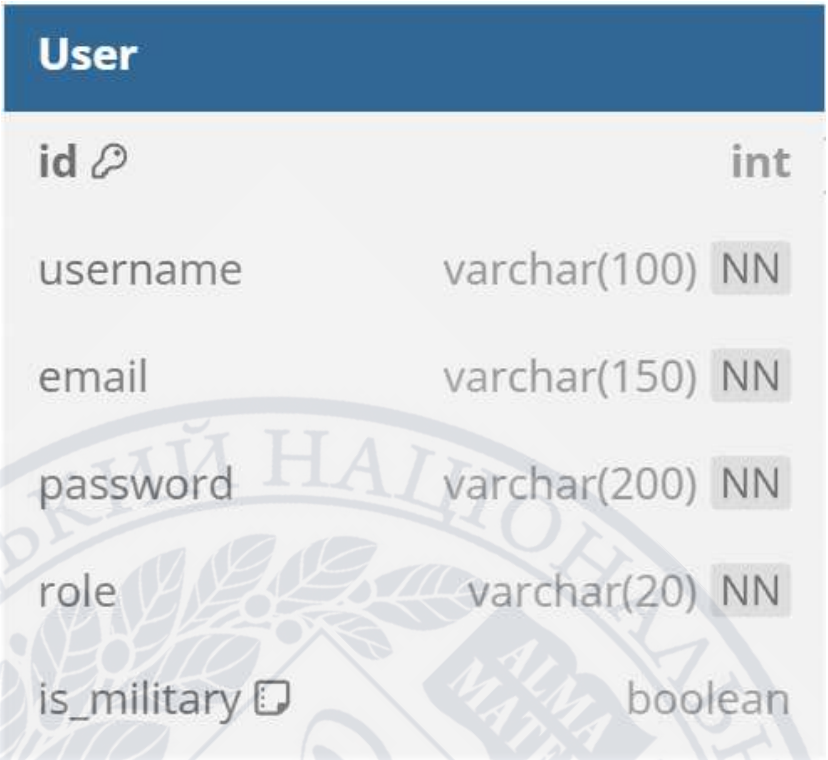

Рисунок 2.1 – Таблиця для збереження даних користувачів

Структура таблиці User:

- id – це первинний ключ, який використовується для ідентифікації користувача та містить в собі числове значення.
- username – ім’я користувача, яке обмежується довжиною в 100 символів.
- email – електронна пошта користувача, яка зберігається у форматі «email@post.com».
- password – пароль користувача, який зберігається у хешованому вигляді, з метою підвищення безпеки даних користувача.
- role – містить роль користувача на сайті (організатор або учасник).
- is_military – зберігає булеве значення, яке позначає приналежність користувача до військових.

Ця табличка забезпечує мінімальні необхідні дані користувача, для того щоб реєструватися на участь у змаганнях, а також зручна для побудови зв’язків з іншими таблицями.

Наступна таблиця Post зберігатиме інформацію про всі змагання та події, які публікуються організаторами, (рис. 2.2).

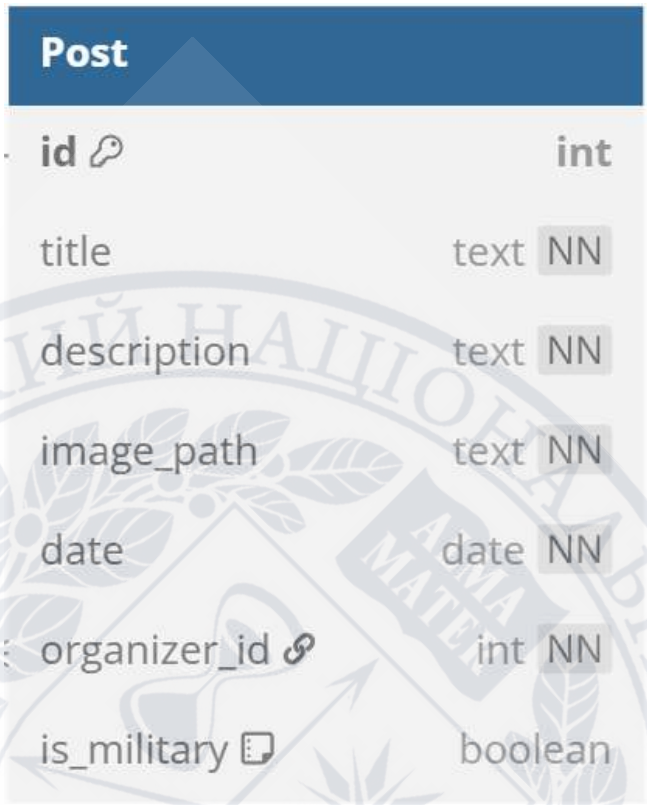

Рисунок 2.2 – Таблиця для збереження даних про змагання

Структура таблиці Post:

- id – це первинний ключ, унікальний ідентифікатор події, містить в собі числове значення.
- title – містить в собі заголовок події, найголовнішу коротку інформацію, яка відображатиметься на картках подій.
- description – зберігає опис події, детальнішу інформацію від організатора для учасників про змагання.
- image_path – містить в собі шлях до зображень, які використовуються як банер події.
- date – зберігає дату події, з відповідним типом даних.
- organizer_id – містить в собі первинний ключ користувача, який організовує подію, має зв’язок з таблицею User.

- is_military – зберігає булеве значення, яке позначає події призначені для військових.

Наступна таблиця Registration міститиме дані про всіх учасників, які зареєструвалися на певну подію, (рис. 2.3).

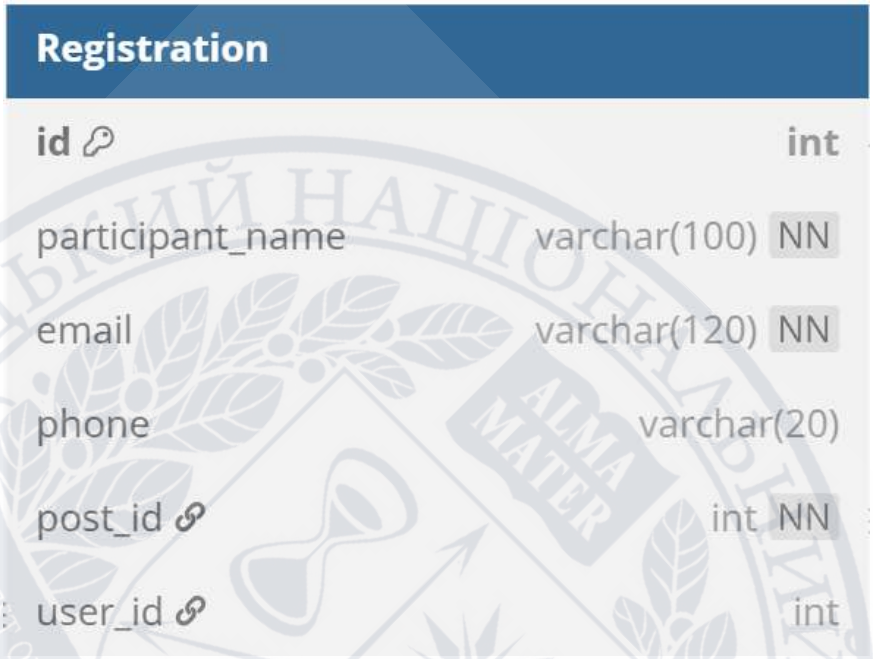

Рисунок 2.3 – Таблиця для збереження даних про реєстрацію на змагання

Структура таблиці Registration:

- id – ідентифікатор запису.
- participant_name – зберігатиме в собі ім’я учасника. Створено як окреме поле, а не зв’язок з таблицею User, для того щоб забезпечити можливість реєстрації неавторизованого користувача.
- email – електронна пошта учасника. Аналогічно створена окремим полем, для того щоб неавторизований користувач міг долучитися до події.
- phone – мобільний номер телефону, для зв’язку із зареєстрованими учасниками.
- post_id – зовнішній ключ із таблицею Post, вказує до якої події відноситься запис реєстрації учасника.

- user_id – зовнішній ключ із таблицей User, на випадок якщо користувач авторизований. В такому випадку користувачу не потрібно буде вводити свої ім’я та пошту, вони підтягнуться автоматчно при реєстрації.

Наступна таблиця Result зберігатиме в собі дані про результати подій, її потрібно створити таким чином, щоб підтримувалися результати з різних видів спорту, наприклад очки, час, кількість голів, місце учасника, (рис. 2.4).

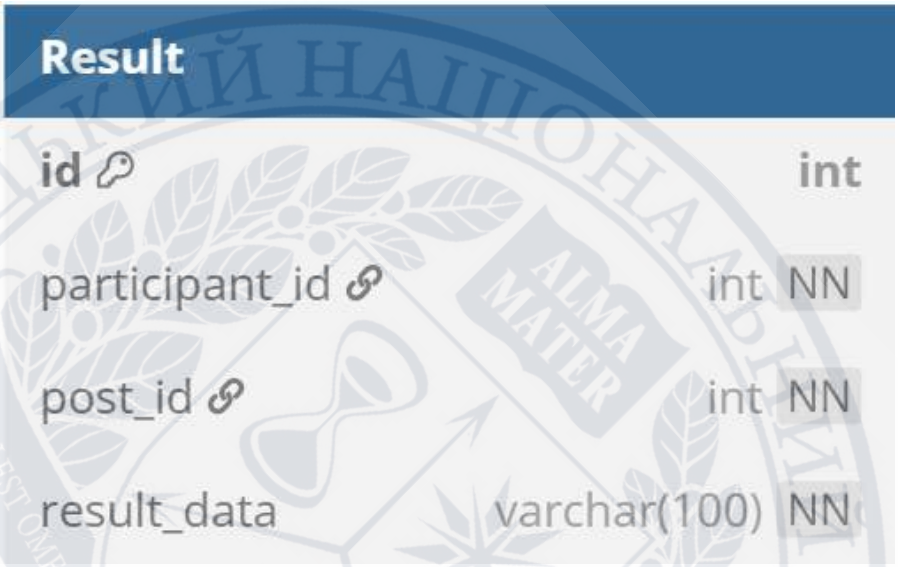

Рисунок 2.4 – Таблиця для збереження результатів змагань

Структура таблиці Result:

- id – первинний ключ, ідентифікатор результату.
- participant_id – зовнішній ключ із таблиці Registration, позначатиме учасника змагань.
- post_id – зовнішній ключ із таблицею Post, позначатиме змагання до яких відносяться результати.
- result_data – поле для збереження результатів. Реалізовано з типом даних varchar для того, щоб організатор міг записати будь які результати.

Загальні властивості бази даних:

- Нормалізація. Структура бази даних відповідає третій нормальній формі, оскільки усі поля в таблицях містять атомарні значення, усі неключові

атрибути повністю залежать від первинного ключа, жоден неключовий атрибут не залежить транзитивно від первинного ключа.

- Зв’язки між таблицями. Всі ключові таблиці пов’язані між собою через зовнішні ключі, що дозволяє забезпечити цілісність даних.
- Масштабованість. У разі розширення функціоналу платформи, структура бази даних легко доповнюється новими таблицями або зв’язками

Спроєктована реляційна база даних охоплює всі функціональні вимоги платформи для проведення спортивних змагань. Вона дозволяє реалізувати завдання поставленні до вебплатформи: створювати та публікувати змагання, реєструвати користувачів як учасників або організаторів, долучатися неавторизованим та авторизованим користувачам до змагань, зберігати результати подій. Гнучка структура забезпечує можливість розширення платформи в майбутньому без істотних змін в архітектурі.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБПЛАТФОРМИ

3.1 Складові системи

Розробка вебплатформи для проведення спортивних подій з використанням фреймворку Flask передбачає просту, але водночас гнучку архітектуру. В основі Flask-проєкту лежить чітко організована структура директорій і файлів, що дозволяє логічно розділяти частини програми на шаблони, стилі, статичні ресурси, серверну логіку та базу даних. Такий підхід забезпечує зручність у розробці, супроводі та змінах, масштабуванні додатку. Розглянемо фізичну структуру проєкту, представлену у вигляді дерева, (рис. 3.1).

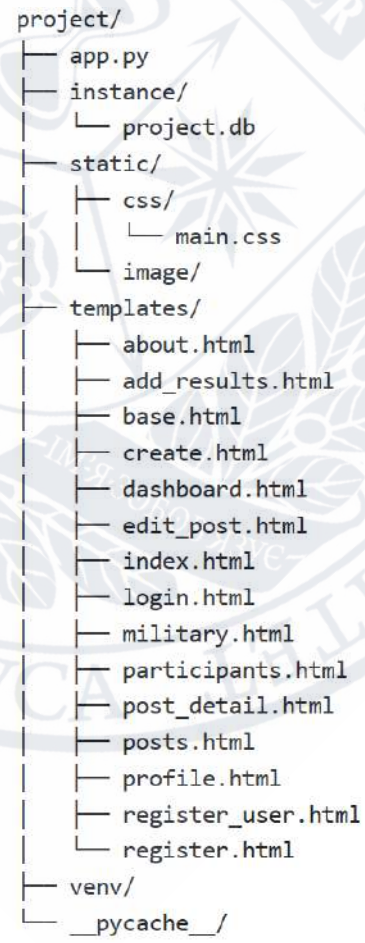

Рисунок 3.1 – Фізична структура проєкту у вигляді дерева

Файл app.py є центральним елементом усього Flask-застосунку. Саме з нього розпочинається виконання вебплатформи. Цей файл виконує роль точки входу в систему та об’єднує усі логічні складові проєкту: конфігурацію, маршрутизацію, підключення до бази даних, ініціалізацію моделей, а також запуск сервера.

У типовому Flask-додатку саме app.py містить головну змінну app, яка є екземпляром класу Flask. Цей екземпляр використовується для реєстрації маршрутів, обробки запитів, підключення до бази даних та налаштування всієї логіки роботи вебсервера.

Flask-додаток містить типові компоненти. Для початку роботи, потрібно створити екземпляр Flask-додатку. Для цього використовується такий код, (рис. 3.2)

```
from flask import Flask
app = Flask(__name__)
```

Рисунок 3.2 – Приклад коду створення об’єкту застосунку

Цей рядок створює вебзастосунок, що базується на Flask. Він приймає в якості аргументу ім’я поточного модуля, що дозволяє Flask правильно визначити шляхи до статичних файлів, шаблонів та інших ресурсів.

Одразу після створення застосунку відбувається його конфігурація. У Flask конфігураційні параметри можуть задаватися безпосередньо у коді або через окремий файл. У цьому проєкті часто використовується база даних SQLite, тому конфігурація виглядає так, (рис. 3.3).

```
app.secret_key = 'your_secret_key_here'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///project.db'
```

Рисунок 3.3 – Конфігурація застосунку

Це вказує Flask, що потрібно використовувати базу даних SQLite, а також відключає надлишковий функціонал відстеження змін в об'єктах, що підвищує

продуктивність. Після задання конфігурації здійснюється ініціалізація системи керування базою даних SQLAlchemy, (рис.3.4).

```
db = SQLAlchemy(app)
```

Рисунок 3.4 – Ініціалізація системи керування БД SQLAlchemy

Після підключення всього необхідного для роботи та створення початкової структури файлу, наступним кроком відбувається ініціалізація моделі, які описують структуру таблиць баз даних. Це відбувається у вигляді класів, (рис. 3.5).

```
class Post(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.Text, nullable=False)
    description = db.Column(db.Text, nullable=False)
    image_path = db.Column(db.Text, nullable=False)
    date = db.Column(db.Date, nullable=False)
    organizer_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
    is_military = db.Column(db.Boolean, default=False)

    organizer = db.relationship('User', backref='events')
    results = db.relationship('Result', back_populates='post', cascade='all, delete-orphan')
```

Рисунок 3.5 – Приклад запису моделі та її зв’язків

Цей код дозволяє системі знати, які таблиці необхідно створити у базі при початковій ініціалізації, а в подальшому для міграцій.

У цьому ж файлі відбувається опис маршрутів – функцій, які відповідають на HTTP-запити користувача, (рис. 3.6).

```
@app.route("/posts")
def posts():
    military_only = request.args.get('military_only') == 'yes'
    if military_only:
        all_posts = Post.query.filter_by(is_military=True).all()
    else:
        all_posts = Post.query.all()
    return render_template('posts.html', posts=all_posts)
```

Рисунок 3.6 – Приклад коду для опису маршрутів

На зображені можна побачити приклад коду, в якому створений маршрут до фалу posts.html, в якому розміщуються всі змагання, які додавалися організаторами. Після чого написана функція, яка обробляє дію користувача – вибір категорії змагань для військових. Якщо користувач обирає цю категорію, виводяться тільки змагання з цією позначкою, а якщо ж користувач не обрав нічого – покажуться всі змагання.

На завершення, в кінці файлу міститься умовний блок, який дозволяє запускати застосунок безпосередньо, (рис.3.7).

```
if __name__ == '__main__':
    app.run(debug=True)
```

Рисунок 3.7 – Код запуску локального сервера розробки

Цей блок забезпечує запуск локального сервера розробки, а параметр «debug=True» дозволяє автоматично перезавантажувати застосунок при зміні коду і виводити корисну інформацію про помилки під час розробки.

Файл app.py виступає коренем усієї системи, поєднуючи всі її модулі в єдине ціле. Він відповідає за налаштування середовища, логіку взаємодії з базою даних, ініціалізацію моделей, маршрутизацію та обробку запитів. Завдяки його структурі, система легко масштабована, підтримувана та зручна у супроводі.

У папці instance розміщується файл «project.db» – це файл бази даних, який зберігає всі структуровані дані вебплатформи для проведення спортивних змагань. Він побудований на основі системи управління базами даних SQLite, що є вбудованою, легкою та зручною для використання в невеликих вебзастосунках.

Цей файл містить усю інформацію, необхідну для роботи системи, включаючи:

- Дані користувачів. Логіни, email-адреси, хешовані паролі, ролі користувачів, а також причетність до збройних сил України.
- Інформацію про події. Назви, описи, маршрути до зображень, дати проведення, організатори, позначка «для військових».

- Дані реєстрації на події інформацію про учасників, електронну пошту, контактні номери.
- Результати змагань. Узагальнене поле для зберігання результатів у різному форматі, прив’язане до конкретних учасників і подій.

Файл «project.db» є ядром системи з точки зору збереження стану – усі дії, які виконує користувач через інтерфейс, зрештою, фіксуються саме в цій базі даних.

Розміщення файлу БД в директорії instance є стандартною практикою у Flask-проєктах для зберігання конфіденційних даних або змінних, що залежать від середовища. Flask автоматично ігнорує цю папку при імпорті, що запобігає випадковому оприлюдненню вмісту, наприклад, при завантаженні проєкту на віддалений сервер чи в систему контролю версій [29].

Це дозволяє відокремити логіку застосунку від даних, які змінюються під час роботи системи.

Файл «instance/project.db» є центральним сховищем даних вебплатформи. Його структура базується на реляційній моделі, підтримує зв’язки між таблицями та відповідає принципам нормалізації. Завдяки використанню SQLite база проста в налаштуванні, не вимагає окремого сервера та легко інтегрується з Flask через SQLAlchemy.

Каталог «static/» призначений для зберігання статичних ресурсів, які не змінюються динамічно сервером у процесі виконання запитів. Ці файли передаються браузеру у незмінному вигляді та використовуються для оформлення та доповнення візуальної частини вебінтерфейсу [30].

Підкаталог «css/» містить таблиці стилів, які відповідають за зовнішній вигляд вебсторінок. У цьому проєкті використовується файл «main.css», який створений для незначних правок в оформленні верхньої частини сайту з навігацією, та нижньої, з різними посиланням. Більшість інтерфейсу стилізовано за допомогою Bootstrap.

Підкаталог «image/» призначений для зберігання графічних матеріалів, які використовуються в межах вебінтерфейсу. Це можуть бути:

- Банери змагань, які завантажують організатори при створенні подій.
- Іконки, логотипи, декоративні елементи інтерфейсу.
- Фонові зображення для оформлення окремих сторінок.

Зображення підключаються через Jinja2. Розміщення зображень в «static/image/» дає змогу легко керувати контентом, зберігати логічну структуру та забезпечувати швидке завантаження медіа на клієнтському боці [31].

Переваги використання «static/» у Flask:

Автоматична обробка. Flask автоматично підключає каталог «static/» для обробки запитів до статичних ресурсів без необхідності створювати окремі маршрути.

Кешування. Браузери кешують файли з цієї директорії, що зменшує навантаження на сервер і прискорює завантаження сторінок.

Структурованість. Легко масштабувати проєкт, додаючи інші ресурси, як-от JavaScript-файли (static/js/), шрифти (static/fonts/) тощо.

Каталог «static/» є невіддільною частиною структури вебплатформи, що відповідає за зберігання ресурсів, які формують візуальне сприйняття системи. Вміст цієї директорії безпосередньо не змінюється сервером, але активно використовується клієнтською частиною для побудови повноцінного, привабливого і зручного інтерфейсу користувача.

Каталог «templates/» призначений для зберігання HTML-шаблонів, які генеруються динамічно за допомогою шаблонізатора Jinja2 – вбудованого механізму Flask, що дозволяє поєднувати HTML-код із Python-логікою [32]. Саме ці шаблони відображаються у браузері користувача як кінцеві вебсторінки. Вони підключаються у відповідних маршрутах через функцію render_template() та заповнюються динамічними даними.

Структуроване використання шаблонів дозволяє досягти повторного використання коду, покращення читабельності, спрощення підтримки та розширення проєкту.

Файл base.html є каркасом для інших сторінок. Він містить загальну структуру, яка повторюється на всіх сторінках:

- HTML-розмітку
- Підключення CSS стилів
- Заголовок, header сайту
- Footer сайту
- Блок контенту, що змінюється.

Інші шаблони наслідують base.html через Jinja2-інструкцію {% extends "base.html" %}, вставляючи лише унікальний контент у відповідні блоки. Решта файлів уже наповнюються контентом та функціями для користувачів.

Файли для виводу подій та їх перегляду:

index.html – головна сторінка, яка коротко викладає зміст сайту. Вона містить перелік найближчих змагань, дані про які передаються з БД через маршрут і відображаються у вигляді карток.

posts.html – сторінка зі списком усіх подій у вигляді карток, які містять короткий опис, дату, та кнопку для детальнішого перегляду інформації про змагання. Містить фільтр «Для військових», який залишає тільки події з відповідною позначкою.

post_detail.html – детальний перегляд інформації про змагання. Виводить заголовок, опис, дату, організатора, зображення та можливість реєстрації на подію. Якщо організатор події вніс результати, то замість кнопки реєстрації буде таблиця результатів.

Шаблони для керування подіями:

create.html – форма створення нової події (тільки для організаторів). Містить поля для заголовку, опису, дати, вибору зображення, позначку «для військових».

edit_post.html – форма редагування вже створеної події. Всі поля автоматично заповнені даними події, що редагується.

add_results.html – форма для додавання або редагування результатів змагань для кожного учасника, пов’язаного з конкретними змаганнями.

Шаблони для автентифікації та реєстрації:

login.html – форма входу для користувачів. Містить поля для логіну та пароля.

register.html – форма реєстрації нового користувача з обов'язковими полями: ім'ям, email, паролем, роллю.

register_user.html – форма реєстрації учасників на змагання. Змінюється залежно від того, чи авторизований користувач, якщо ні, то дані доведеться вводити самостійно.

Шаблони для особистого кабінету та користувача:

profile.html – профіль користувача з інформацією про його участь у подіях.

dashboard.html – панель керування для організатора. Містить список подій, створених користувачем, з можливістю керування ними.

participants.html – сторінка зі списком зареєстрованих учасників певної події. Доступ до неї мають тільки організатори з особистого кабінету. В ній відображаються дані учасників змагань, такі як ім'я, номер телефону, email.

Інформаційні сторінки:

about.html – загальна інформація про вебплатформу, її цілі, функціональність, контактні дані.

military.html – сторінка із інформацією для військових, висвітлює доступні можливості на платформі.

Завдяки механізму шаблонізації Jinja2, ці сторінки динамічні – реагують на дії користувача. Jinja2 підтримує змінні, умовні конструкції, цикли, наслідування шаблонів, включення підшаблонів. Це дозволяє динамічно формувати HTML-код на основі даних з БД .

Каталог «templates/» є ключовим для реалізації інтерфейсу користувача у вебплатформі. За допомогою шаблонів відображаються події, обробляється введення даних, реалізується автентифікація та навігація по сайту. Завдяки механізму наслідування базового шаблону і використанню Jinja2-перемінних забезпечується зручність розробки, розширюваність системи та висока ефективність побудови вебінтерфейсу [33].

«venv/» та «__pycache__/» це технічні директорії. «venv/» – віртуальне середовище Python, яке ізолює залежності проєкту. Це забезпечує стабільність, оскільки всі бібліотеки встановлюються локально й не залежать від глобального середовища Python на машині розробника [34]. «__pycache__/» – технічна директорія, яка створюється автоматично інтерпретатором Python для кешування байт-коду модулів. Її вміст не редагується вручну [35].

Структура Flask-проєкту дозволяє ефективно розділити відповідальності між різними компонентами вебплатформи. Чітке розміщення файлів у відповідних каталогах підвищує читаність коду, полегшує тестування, обслуговування та подальший розвиток системи.

3.2 Інтерфейс користувача

У цьому підрозділі розглядаються основні елементи інтерфейсу: головна сторінка, сторінки подій, реєстрації, профілю користувача та панелі організатора. В побудові інтерфейсу метою було досягнення зручності, інтуїтивності, логічності структури, оскільки це найбільше впливає на покращення користувацького досвіду.

Перше, що побачить користувач, це головна сторінка. Вона містить навігаційну панель зверху, кнопки для реєстрації або авторизації на сайті, перелік найближчих змагань до яких можна долучитися, кнопку-посилання для військових, та футер сайту, який містить основну контактну інформацію, (рис.3.8).

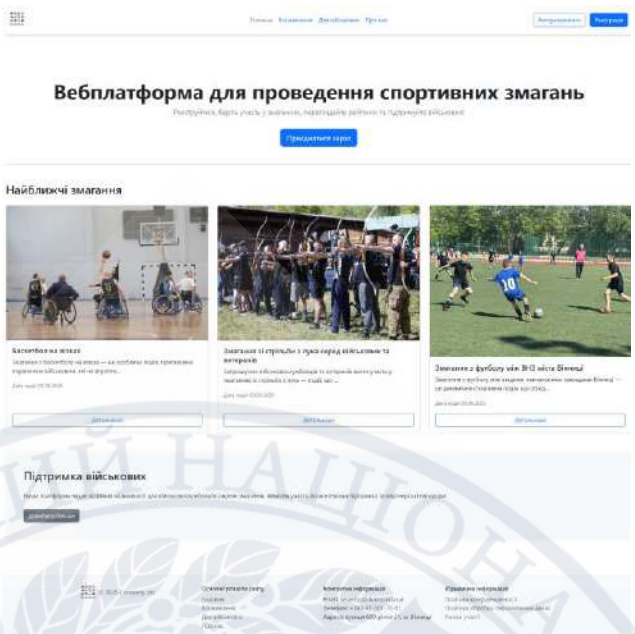

Рисунок 3.8 – Головна сторінка вебплатформи

Наступним кроком, користувачу потрібно зареєструватися на сайті. Для цього можна натиснути кнопку «приєднатися зараз» або кнопку «реєстрація». Після чого користувача перенаправить до реєстраційної форми, (рис. 3.9).

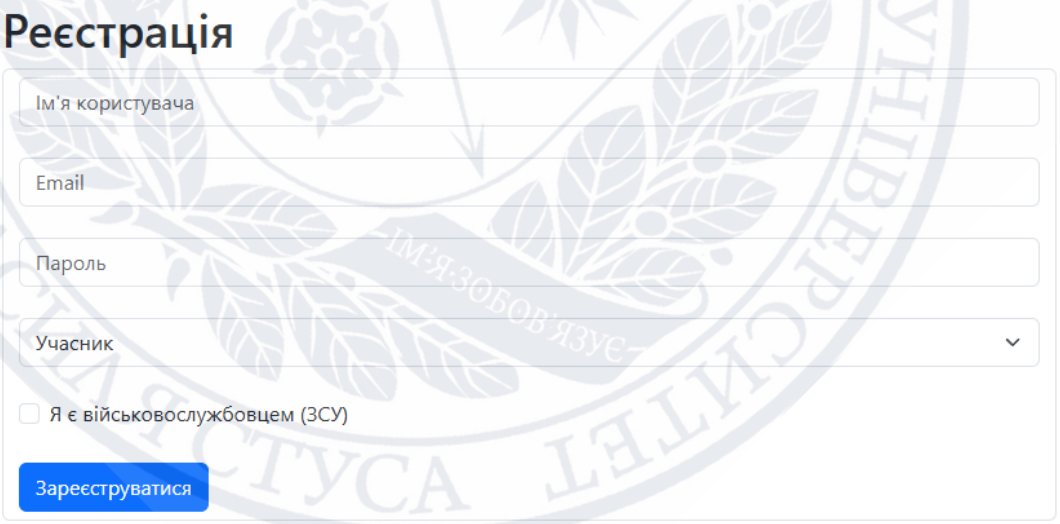

Рисунок 3.9 – реєстраційна форма для користувача

На цьому етапі користувач обирає роль учасника або організатора. Вибір ролі впливає на доступ до функцій в подальшому. Також, можна вказати причетність до ЗСУ. Після внесення даних та підтвердження реєстрації потрібно авторизуватися за допомогою імені та пароля, (рис. 3.10).

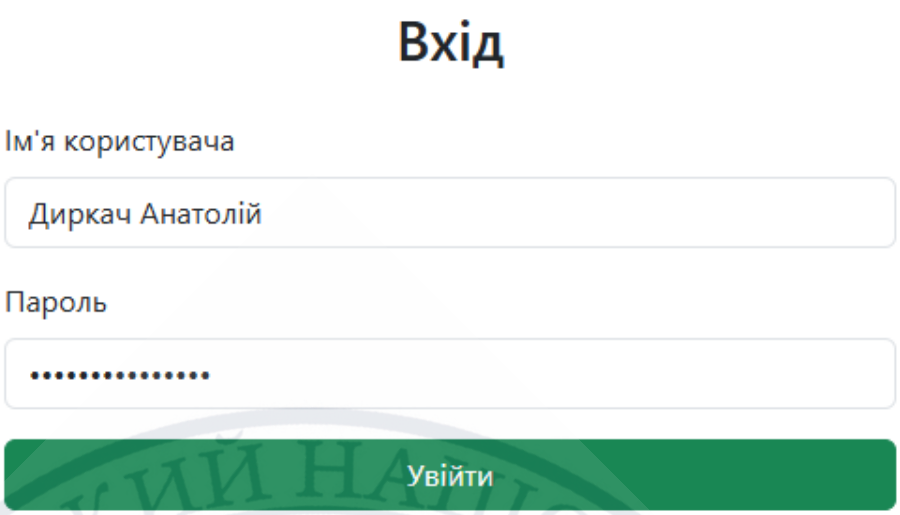

Рисунок 3.10 – Приклад форми авторизації

Наступна сторінка – «Всі змагання», тут користувач може обрати подію, яка його цікавить, та переглянути про неї більш детальну інформацію: назву, коротенький опис, дату. Після цього, можна натиснути кнопку «Детальніше» та перейти на сторінку із подією, (рис. 3.11).

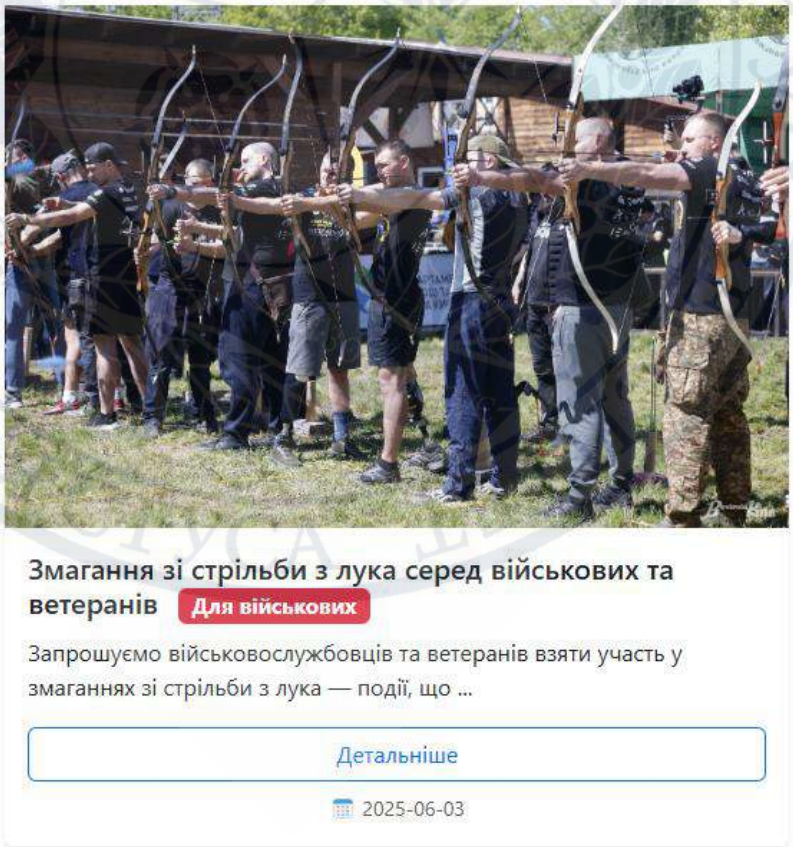

Рисунок 3.11 – Вигляд картки із подією

Якщо заголовок зацікавив користувача, він може переглянути детальну інформацію про змагання.

Рисунок 3.12 – Сторінка із змаганнями

Після натискання кнопки зареєструватися, відкривається сторінка із реєстраційною формою, де потрібно написати особисті дані. Форма буде відрізнятися залежно від того чи авторизувався користувач, (рис. 3.13 – 3.14).

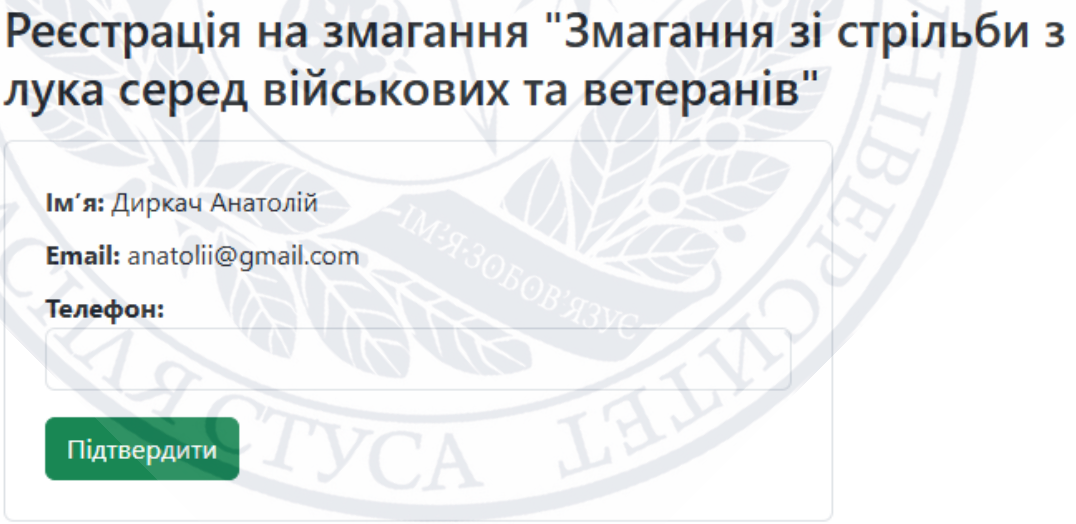

Рисунок 3.13 – Реєстраційна форма на змагання авторизованого користувача

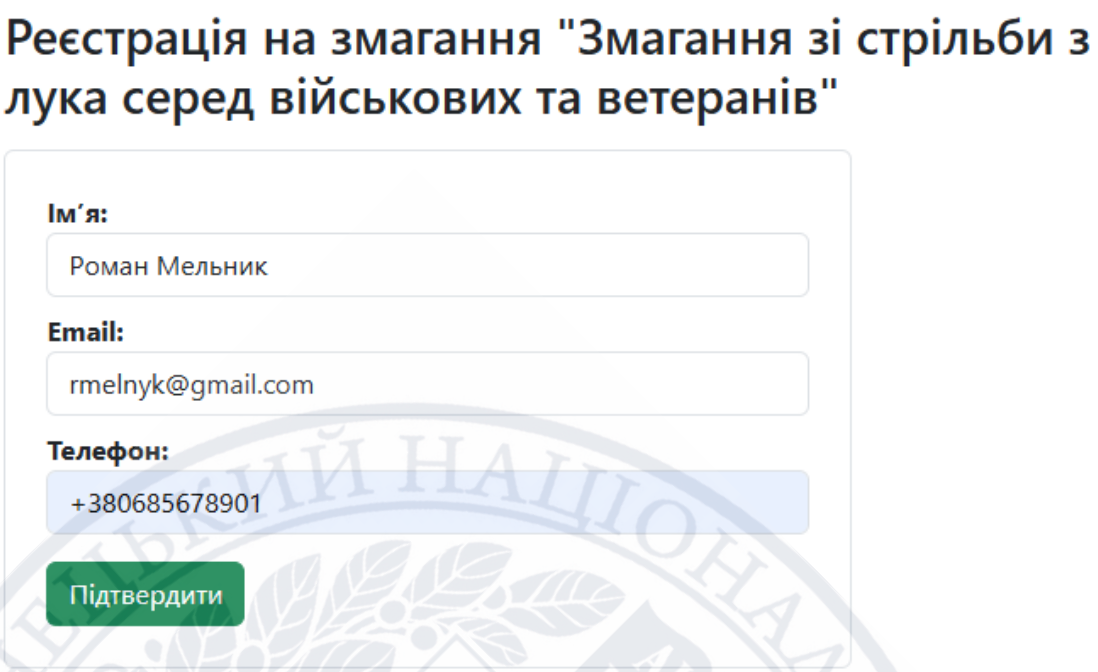

Рисунок 3.14 – Реєстраційна форма не авторизованого користувача

Після підтвердження даних, учасник побачить список учасників, які вже зареєструвалися, (рис. 3.15).

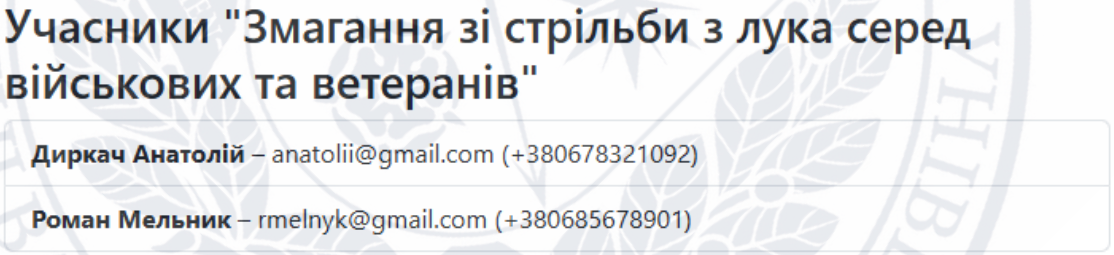

Рисунок 3.15 – Список учасників змагання

Перевага авторизованого користувача полягає в тому, що деякі дані про нього вносяться автоматично, а також він має особистий кабінет, де може переглядати змагання на які він зареєструвався, список конкурентів, скасувати реєстрацію на змагання, (рис. 3.16).

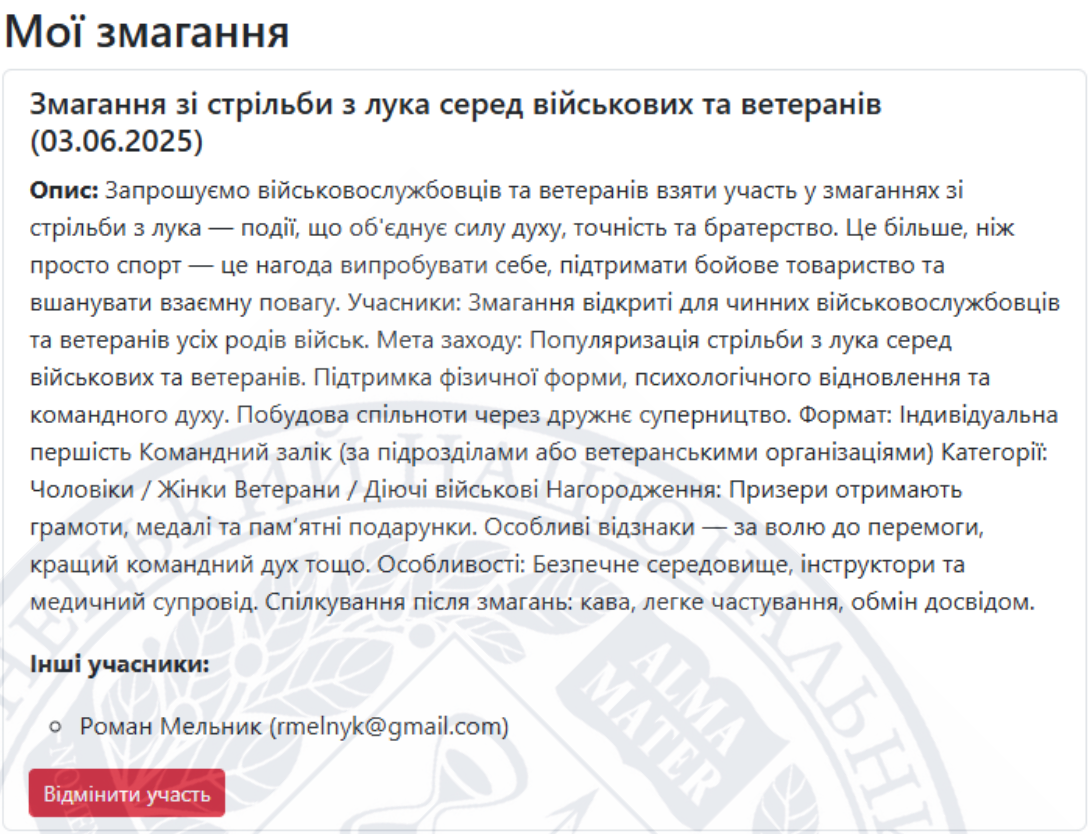

Рисунок 3.16 – Приклад вигляду картки реєстрації на змагання

Наступним етапом, буде розгляд можливостей користувача з роллю організатора. Процес реєстрації залишається незмінним, потрібно тільки обрати роль «Організатор» у відповідному випадаючому списку, (рис.3.17).

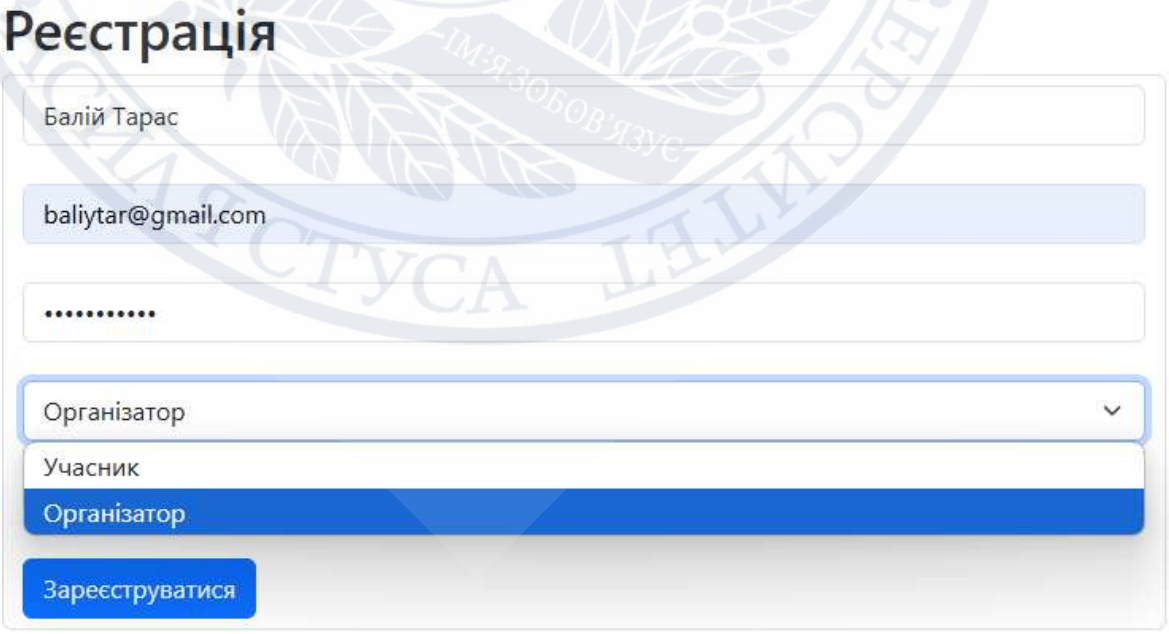

Рисунок 3.17 – Реєстрація від лиця організатора

Перевагою організатора є можливість створювати змагання. Для цього в нього є доступ до спеціального функціоналу, а також інший особистий кабінет. На зображенні нижче буде форма для створення змагання, яка відкривається після натискання на кнопку «Створити подію» на навігаційній панелі, (рис. 3.18).

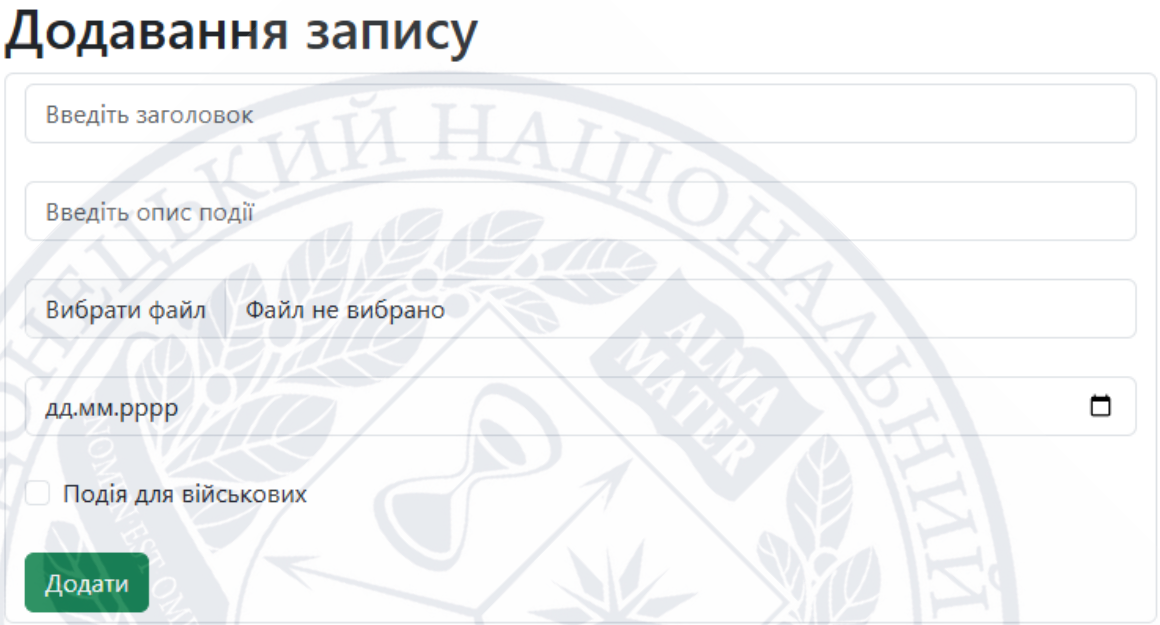

Рисунок 3.18 – Форма створення змагання

Данна форма збирає всі данні, які бачить користувач на картці змагання. Заголовок та опис події можна ввести самостійно або ж вставити зарання скопійований. Зображення можна обрати із збережених на комп’ютері. Дату можна ввести самостійно, або ж натиснути на значок календарика, та обрати в календарі. Прапорець «Подія для військових» відрізнить подію від решти, вона буде позначена червоним надписом «Для військових», (рис. 3.19).

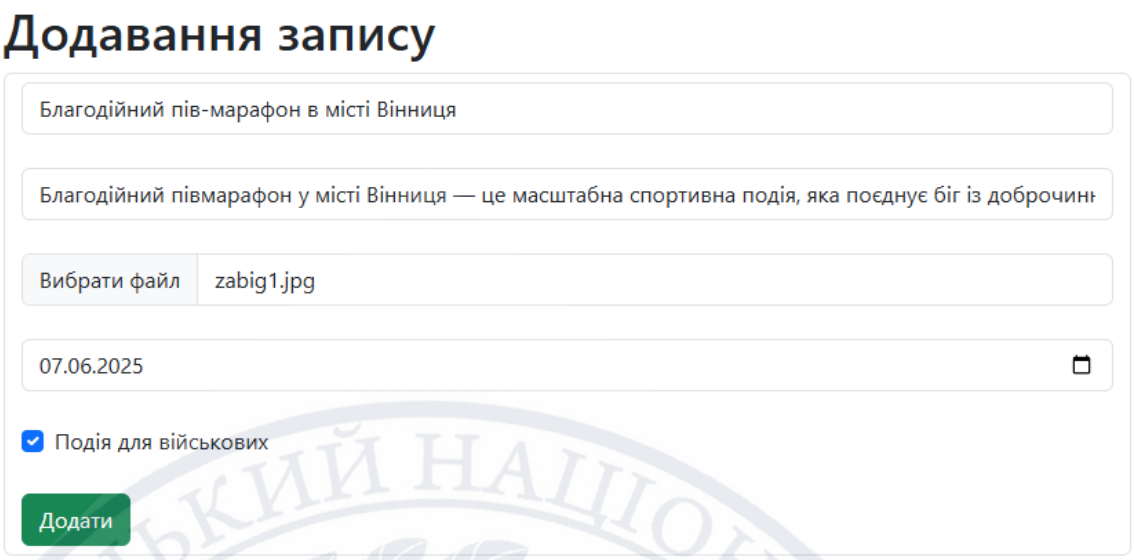

Рисунок 3.19 – Заповнена форма змагання

Після натискання кнопки додати, подію можна побачити на сторінці із всіма змаганнями, (рис. 3.20).

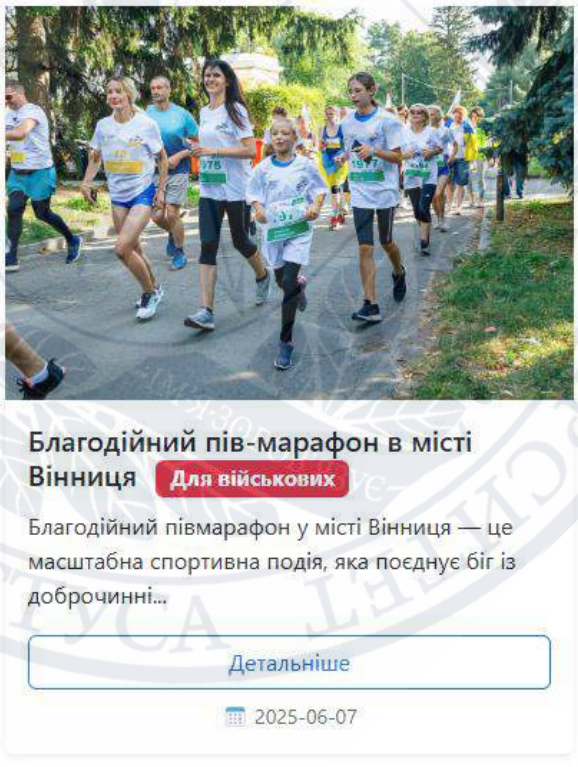

Рисунок 3.20 – Карта новоствореного змагання

Якщо організатор перейде в особистий кабінет, він побачить перелік створених ним подій. До кожної події збоку розміщенні кнопки, які дозволяють з нею взаємодіяти, (рис. 3.21).

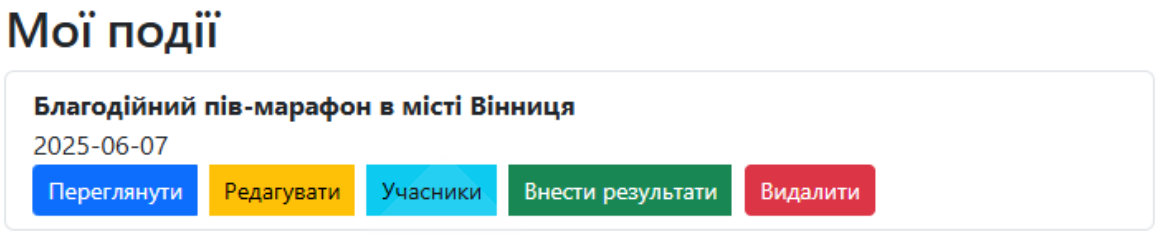

Рисунок 3.21 – Приклад вигляду змагань з кабінету організатора

Кнопка переглянути дозволяє відкрити сторінку змагання з детальним описом. Кнопка редагувати створена для внесення змін у деталі події. Крім назви та опису, можна обрати інше фото, дату, та змінити статус «Для військових». Приклад форми для редагування на зображені, (рис 3.22).

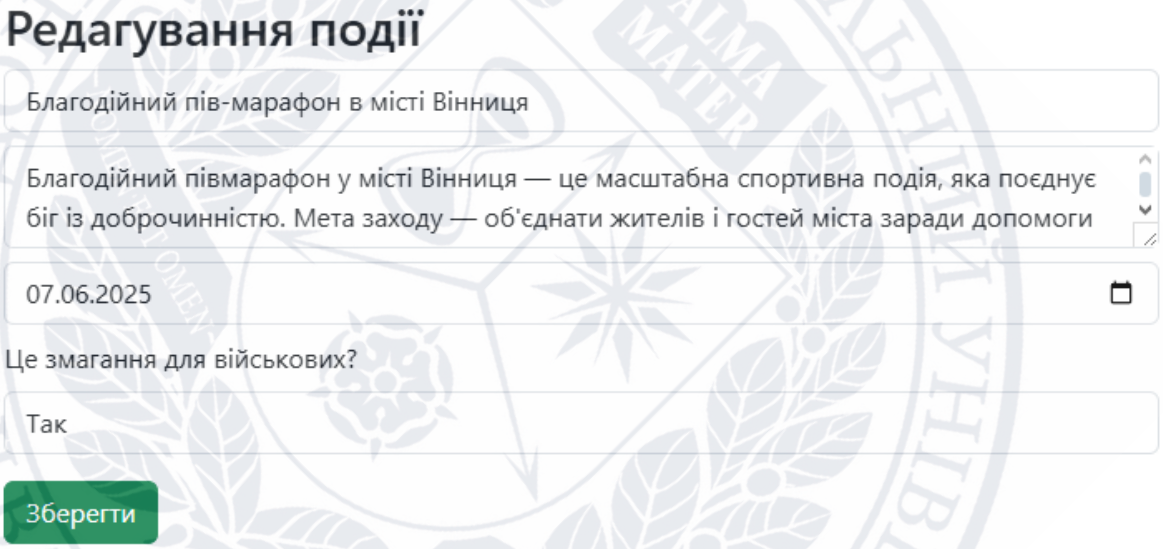

Рисунок 3.22 – Форма для редагування події

Кнопка учасники дозволяє переглядати список зареєстрованих учасників, який містить їхні імена, номери телефонів та електронну пошту, (рис.3.23).

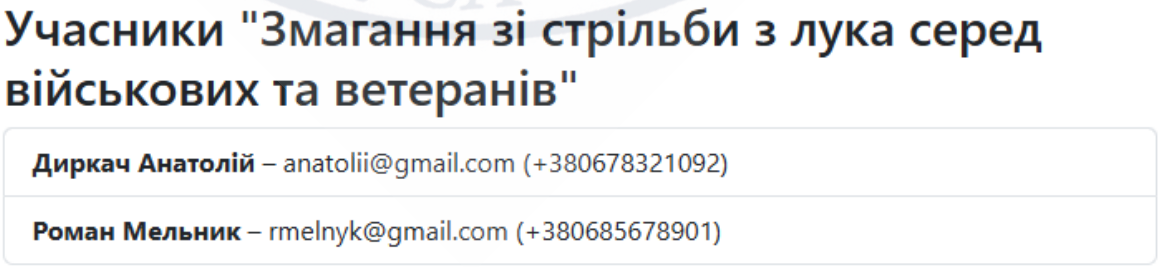

Рисунок 3.23 – Список учасників змагання

Кнопка «Внести результати» відкриває сторінку із списком учасників та вільним полем для введення. В це поле потрібно ввести результати, вони зберігатимуться у текстовому форматі, тому можна записувати як і час, так і «перше місце», (рис. 3.24).

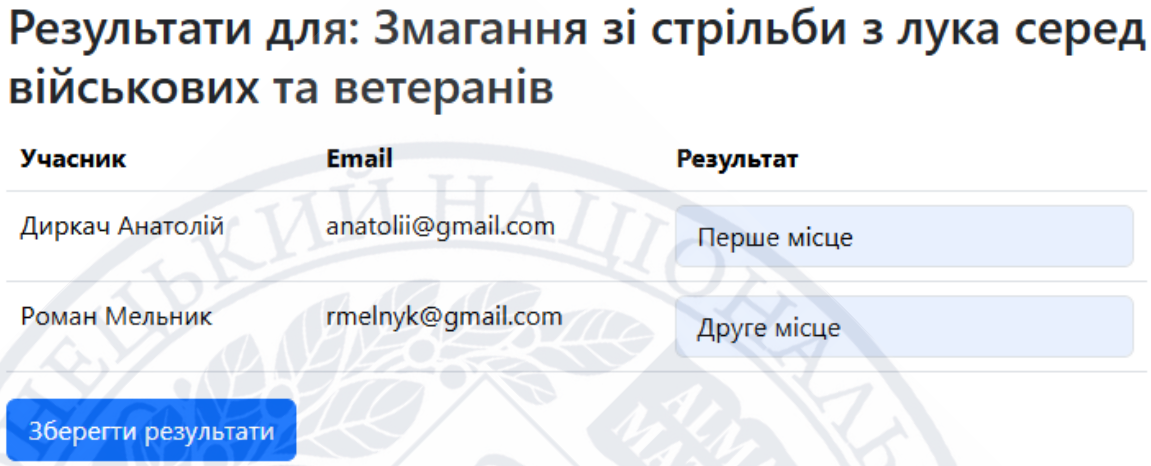

Рисунок 3.24 – Форма для внесення результатів змагань

Після збереження результатів, на сторінці із деталями події зникне кнопка реєстрації. Замість неї будуть відображатися результати події, (рис. 3.25).

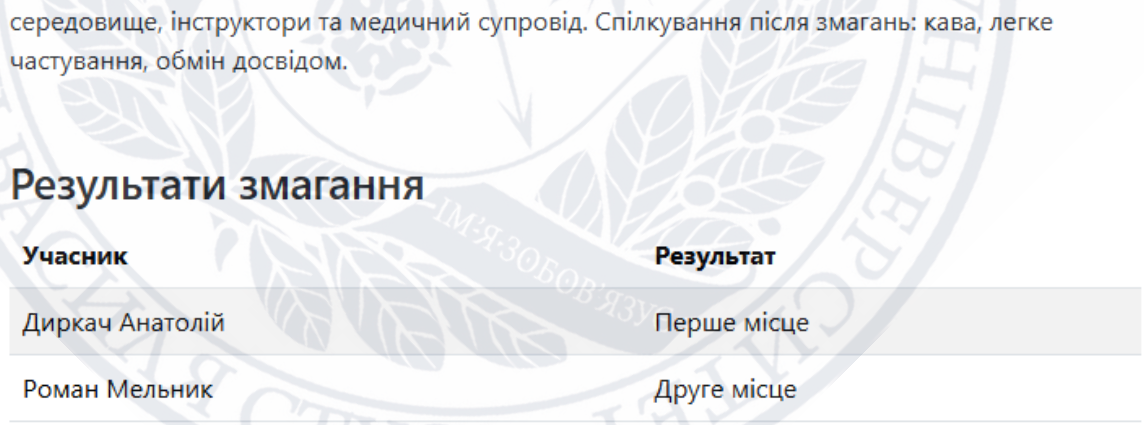

Рисунок 3.25 – Результати змагання на сторінці події

Кнопка «Видалити» видаляє змагання. Якщо її натиснути, буде спливаюче вікно з уточненням дії. Після підтвердження, подія зникає та видаляється з БД, (рис. 3.26).

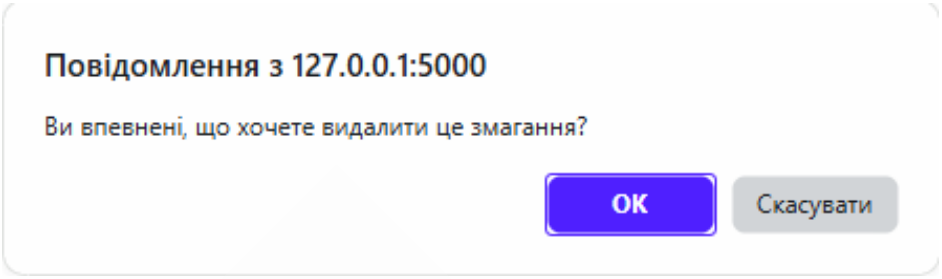

Рисунок 3.26 – Спливаюче вікно при видаленні змагань

Якщо користувач військовий і вперше відвідує сайт, на головній сторінці внизу він помітить заголовок «Підтримка військових», (рис. 3.27).

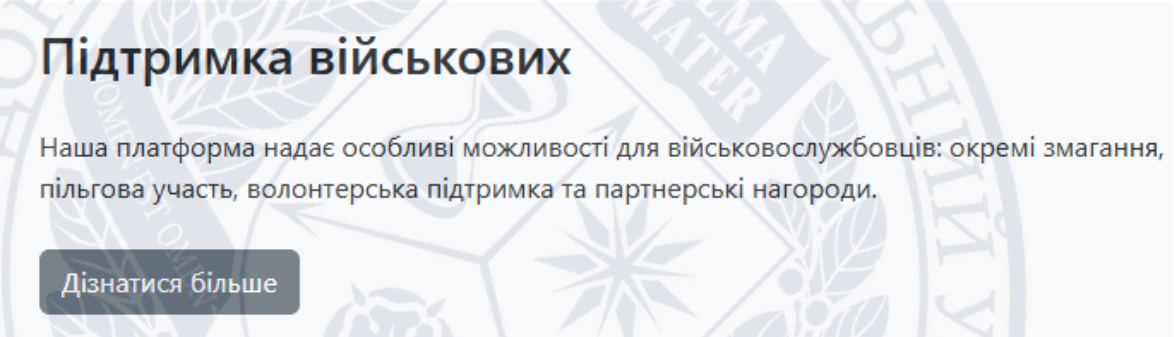

Рисунок 3.27 – Заголовок про підтримку військових

При переході на цю сторінку, користувач побачить опис спеціальних можливостей для військових:

- спеціальні змагання з участю військових.
- допомога у реабілітації та підтримка.
- адаптивні види спорту для поранених військових.

Після опису можливостей, внизу сторінки розміщена інформація про те як долучитися, (рис. 3.28).

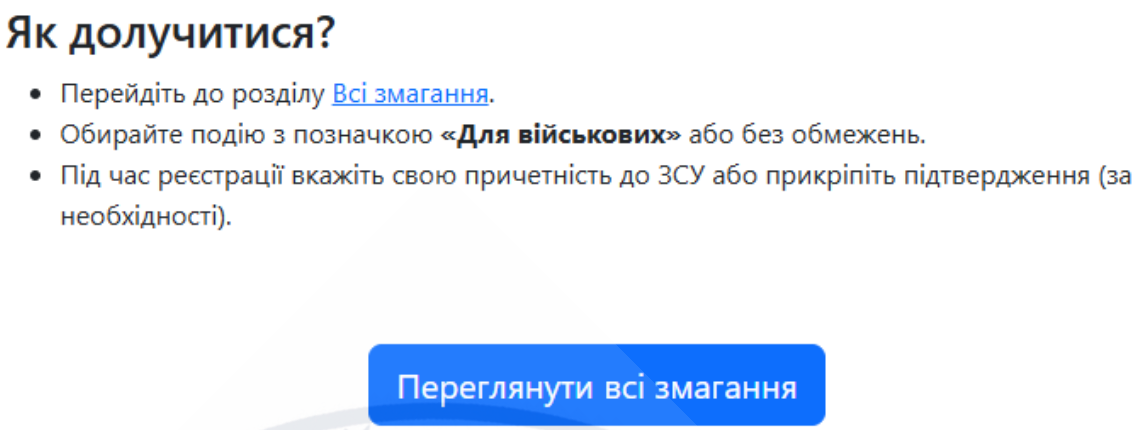

Рисунок 3.28 – Інструкція як долучитися до змагань військовим

На кожній сторінці вебплатформи розмішується навігаційна панель та футер сайту. Навігаційна панель містить посилання на всі сторінки вебплатформи, логотип, та кнопки які стосуються авторизації, реєстрації, або ж особистого кабінету користувача. Футер містить посилання для навігації сторінкою, контактну інформацію, та умови використання веб-сторінки, (рис. 3.29).

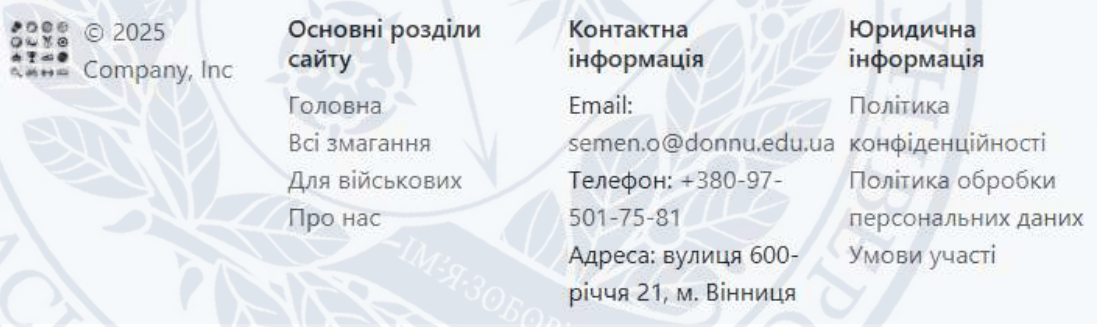

Рисунок 3.29 – Футер вебплатформи

У підсумку, було досягнуто поставленої мети – створення зручного, інтуїтивно зрозумілого, логічно структурованого інтерфейсу користувача. Завдяки своїй простоті та стриманості в кольорах, інтерфейс не перевантажує користувача. Увага зосереджується на найголовнішому – змаганнях та участі в них. По при це, завдяки використанню заокруглених форм та кольорових кнопок, інтерфейс має сучасний та гармонійний вигляд.

3.3 Перспективи розширення сайту

Розроблена вебплатформа є функціонально завершеним рішенням для організації та проведення спортивних змагань. Водночас її архітектура побудована таким чином, щоб у майбутньому було можливо гнучко розширювати функціонал, масштабувати базу даних та вдосконалювати користувацький досвід. Існує велика кількість можливих варіантів розширення платформи:

1. Перехід на іншу СУБД. Для продуктивного середовища, у разі зростання кількості користувачів і подій, доцільно реалізувати підтримку таких СУБД, як PostgreSQL або MySQL, які краще справляються з паралельними запитами, підтримують масштабування та резервне копіювання. Завдяки використанню ORM SQLAlchemy, цей перехід можливий без суттєвих змін у коді.

2. Розширення кабінету користувача. На даний момент профіль користувача не має багато функціоналу, але його можна доповнити можливістю налаштовувати профіль, наприклад, змінювати аватар, контактну інформацій. Реалізувати історію участі в змаганнях та відображення власних результатів, які будуть підтягуватись з БД. Розробити систему оповіщення про нові змагання які цікавлять користувача, а також про зміни у тих, на які він зареєстрований. Створити систему рейтингів або досягнень, для більшої мотивації та залученості користувачів.

3. Підтримка коментування та відгуків. Для формування активної міцної спільноти, можна створити середовище для спілкування між учасниками та організаторами. Це стане поштовхом для розвитку як платформи, так і спортивної спільноти загалом. Організатори отримуватимуть реальні відгуки, а учасники зможуть пропонувати свої ініціативи. Це дозволить створити активне ком’юніті навколо змагань.

4. Додавання адміністративної панелі. На цьому етапі є тільки дві ролі для користувачів платформи, але для такої вебплатформи доцільно створити третю – адміністратора, завданням якого буде контроль контенту та користувачів

на платформі. Адміністративна панель може мати функціонал для блокування користувачів або зміни їх ролей, модерації контенту – щоб запобігти публікації забороненого контенту тощо.

5. Інтеграція з електронною поштою. Автоматична розсилка листів користувачам після реєстрації, підтвердження участі у події, нагадування про дату змагання або повідомлення про публікацію результатів дозволить підвищити інформованість та залученість.

6. Підтримка геологації та мап. Додавання підтримки відображення місця проведення змагань на інтерактивній карті покращить користувацький досвід. Крім того, можна реалізувати фільтрування подій за відстанню до учасника, за допомогою радіусу.

7. Додавання оплати участі у змаганнях. Одним із перспективних напрямків розвитку платформи є реалізація функціоналу онлайн-оплати участі у змаганнях. Це дозволить організаторам монетизувати свої події, а учасникам – здійснювати реєстрацію у кілька кліків без необхідності додаткового зв’язку. Це можна реалізувати за рахунок розширення таблиць та інтеграції API популярних платіжних шлюзів, таких як LiqPay або WayForPay через REST-запити.

8. Розширення варіативності змагань. На даний момент вебплатформа не підтримує командної участі у змаганнях, але в подальшому це можна реалізувати. Крім того, додати можливості перегляду учасників команд, тренерського складу, успішності команди. Командні змагання можуть бути реалізовані у вигляді турніру на вибування або ж ліги.

9. Розширення форми реєстрації. Із розширенням варіативності змагань, буде необхідність розширювати реєстраційні форми: додати поля для завантаження документів (підтвердження причетності до збройних сил України чи медичні довідки), поле для вказання досвіду в обраній дисципліні, вибір формату участі, додаткові послуги у вигляді замовлення мерчу, харчування або проживання під час змагань.

Розроблена вебплатформа має значний потенціал для подальшого розвитку та вдосконалення. Запропоновані напрямки розширення охоплюють як

функціональні покращення для користувачів, так і адміністративні можливості, що дозволить забезпечити ефективну підтримку платформи на всіх етапах її життєвого циклу. Реалізація таких змін зробить систему більш універсальною, конкурентною та придатною до використання у широкому спектрі спортивних подій, придатною для комерційної реалізації.

ВИСНОВКИ

У ході виконання дипломної роботи було розроблено функціональну вебплатформу для організації та проведення спортивних змагань, яка відповідає сучасним вимогам до цифрових сервісів у сфері спорту. Основною метою проєкту було створення зручного, надійного та розширюваного вебзастосунку, який дозволяє як організаторам змагань, так і їхнім учасникам ефективно взаємодіяти між собою, керувати подіями, реєструватися та переглядати результати.

Розробка була здійснена на основі мікрофреймворку Flask, що забезпечило гнучкість у побудові структури застосунку та можливість розширення функціоналу в майбутньому. В процесі реалізації особливу увагу було приділено архітектурі системи, яка включає логічне розділення відповідальностей між модулями, підтримку шаблонізації за допомогою Jinja2, зберігання шаблонів у структурованому вигляді та організацію статичних ресурсів у відповідних каталогах. Такий підхід забезпечує легкість у підтримці та масштабуванні проєкту.

У системі реалізовано всі базові можливості, які необхідні для повноцінної роботи платформи: створення подій, їх редагування, відображення списку змагань, перегляд детальної інформації про кожну подію, можливість реєстрації як авторизованим, так і неавторизованим користувачам. Крім того, реалізовано модуль збереження результатів змагань, що дозволяє організаторам вносити підсумки подій, а учасникам – переглядати свої досягнення.

Особливістю проєкту стало впровадження підтримки категорії «військових» користувачів, а також реалізація подій із міткою «для військових». Це дозволяє гнучко фільтрувати події та формувати цільовий контент для різних категорій користувачів, що є особливо актуальним у сучасних умовах.

На рівні бази даних було розроблено реляційну модель, що відповідає принципам нормалізації. Завдяки використанню SQLAlchemy ORM, реалізовано ефективну взаємодію між об’єктами програми та базою даних. Проєкт було

протестовано на предмет коректної роботи основного функціоналу, що підтвердило його готовність до подальшого розгортання.

Також у дипломній роботі було проаналізовано можливі шляхи розширення функціональності платформи. Серед перспектив – реалізація онлайн-оплати участі у змаганнях, інтеграція з електронною поштою, підтримка командної участі, створення адміністративної панелі для модерації та керування користувачами, вдосконалення профілю користувача, зокрема збереження історії участі та відображення персональних досягнень. Ці кроки дозволять значно підвищити рівень взаємодії між платформою та користувачами, розширити аудиторію та зробити сервіс привабливішим для нових організаторів.

Крім того, важливо зазначити, що розроблена система побудована з урахуванням гнучкості: легко адаптується до різних баз даних, а використання шаблонного підходу дозволяє просто вносити зміни в інтерфейс. У структурі проєкту чітко виділено всі ключові компоненти – від логіки маршрутизації до шаблонів та ресурсів – що відповідає загальноприйнятим підходам до побудови сучасних Flask-застосунків.

Таким чином, розроблена вебплатформа є не лише завершеним продуктом із базовою функціональністю, але й потужною основою для подальшого розвитку. Вона може бути розгорнута у навчальних закладах, спортивних клубах, громадських ініціативах або в рамках державних програм для підтримки масового спорту, зокрема серед військовослужбовців. Універсальність, простота у користуванні, гнучкість у конфігурації та підтримка перспективного розширення роблять цей застосунок актуальним і затребуваним рішенням у сфері цифровізації спортивних процесів.

СПИСОК ВИКОРИСТАНИХ ПОСИЛАНЬ

1. Міністерство молоді та спорту України - Матвій Бідний: «Спорт – це ефективний інструмент реабілітації військових та потужний драйвер повоєнного відновлення». *Головна | Міністерство молоді та спорту України*. URL: https://surl.li/fmuqdq (Дата звернення: 14.04.2025).

2. RunUkraine | We are Road Runners. *RunUkraine*. URL: https://runukraine.org (Дата звернення: 14.04.2025).

3. Simplified Tournament Management. *Challonge.* URL: https://challonge.com/uk (Дата звернення: 15.04.2025).

4. Гончаренко Д. В., Мокін В. Б. Аналіз підходів щодо вибору архітектури інформаційних систем на основі інтернету речей за реальних умов. м. Вінниця, 2022 р. URL: https://library.megu.edu.ua:9443/jspui/handle/123456789/4213 (Дата звернення: 15.04.2025).

5. Фетісов Я. Побудова багаторівневого веб-застосування на хмарній платформі. 2024 р. URL: https://ekmair.ukma.edu.ua/handle/123456789/32073 (Дата звернення: 17.04.2025).

6. Круковська Я. В. Побудова багаторівневого веб-застосування на dockerплатформі : Курсова робота. Київ, 2021. 34 с. URL: https://surl.lu/jgamvy (Дата звернення: 17.04.2025).

7. Розробка веб-застосунку з багатошаровою архітектурою на основі платформи ASP.NET MVC CORE 2.0. URL: https://surl.lu/rkqvor 153-155 с. (Дата звернення: 19.04.2025).

8. Що таке Back-end розробка. *Wezom.* URL: https://wezom.com.ua/ua/blog/chto-takoe-back-end-razrabotka (Дата звернення: 19.04.2025).

9. Що таке backend-розробка і чим вона відрізняється від frontend. *MCToday.* URL: https://mc.today/uk/shho-take-backend-rozrobka/ (Дата звернення: 20.04.2025).

10. What is three-tier architecture. *IBM.* URL: https://www.ibm.com/think/topics/three-tier-architecture (Дата звернення: 20.04.2025).

11. 5 найкращих систем управління базами даних у 2022 році. *IStep Academy.* 30 черв. 2022 р. URL: https://cloud.itstep.org/blog/5-best-database-management-systems-in-2022 (Дата звернення: 24.04.2025).

12. NoSQL databases: Types, use cases, and 8 databases to try in 2025. *NetApp.* URL: https://www.instaclustr.com/education/nosql-database/nosql-databases-types-use-cases-and-8-databases-to-try/ (Дата звернення: 24.04.2025).

13. Оптимізація роботи з базами даних за допомогою ORM (Object-Relational Mapping). *ITRaiting.* URL: https://surl.li/pyivcp (Дата звернення: 24.04.2025).

14. Що таке HTML. *CSS.IN.UA.* URL: https://css.in.ua/article/shcho-take-css_3 (Дата звернення: 24.04.2025).

15. Гущин Д. HTML5 семантичні елементи. *КовелPost.* URL: https://kovelpost.com/blogs/283 (Дата звернення: 25.04.2025) .

16. Довідник по CSS властивостямю. *CSS.IN.UA.* URL: https://css.in.ua/css/properties (Дата звернення: 25.04.2025).

17. Bootstrap. URL: https://getbootstrap.com (Дата звернення: 27.04.2025).

18. Литвинюк В.С., Січко Т.В. Особливості створення веб-додатків та веб-сайтів за допомогою технології bootstrap 4. (Дата звернення: 27.04.2025).

19. Python. URL: https://www.python.org (Дата звернення: 10.05.2025).

20. Довідник з мови Python. *Python.* URL: https://docs.python.org/uk/3.13/reference/index.html (Дата звернення: 10.05.2025).

21. Welcome to Flask - Flask Documentation. *PalletsProject.* URL: https://flask.palletsprojects.com/en/stable/ (Дата звернення: 10.05.2025).

22. Близнюк А. Плюси і мінуси Django. *UkrNames.* URL: https://blog.ukrnames.com/veb-master/plyusi-i-minusi-django (Дата звернення: 10.05.2025).

23. SQLAlchemy – The Database Toolkit for Python. *SQLAlchemy*. URL: https://www.sqlalchemy.org (Дата звернення: 10.05.2025).

24. Основи Flask для веброзробки програмного забезпечення. *PNN*URL: https://pnn.com.ua/ua/blog/detail/lightweight-and-flexible-flask-fundamentals-for-software-web-development (Дата звернення: 12.05.2025).

25. What is SQLite. *SQLite*. URL: https://www.sqlite.org (Дата звернення: 12.05.2025).

26. Що таке SQLite. *FreeHost*. URL: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-sqlite/ (Дата звернення: 12.05.2025)

27. Visual Studio: IDE and Code Editor for Software Developers. *Microsoft*. URL: https://visualstudio.microsoft.com (Дата звернення: 12.05.2025).

28. Що таке Visual Studio Code. *TopKeis*. URL: https://top-keis.com.ua/shho-take-visual-studio-code/ (Дата звернення: 12.05.2025).

29. Flask Application Structure – Flask Documentation. *Flask*. URL: https://flask.palletsprojects.com/en/latest/patterns/packages/#larger-applications (Дата звернення: 16.05.2025).

30. Serving Static Files – Flask Docs. *Flask*. URL: https://flask.palletsprojects.com/en/latest/tutorial/static/ (Дата звернення: 16.05.2025).

31. Template Designer Documentation. *Jinja*. URL: https://jinja.palletsprojects.com/en/latest/templates/ (Дата звернення: 16.05.2025).

32. Flask Documentation. Templates. *Flask*. URL: https://flask.palletsprojects.com/en/latest/templating/ (Дата звернення: 16.05.2025).

33. Ascany P. Using Jinja2 Templates in Flask Applications. *RealPython*. URL: https://realpython.com/primer-on-jinja-templating/ (Дата звернення: 16.05.2025).

34. Python Packaging Authority. Installing packages using pip and virtual environments. *Python.org*. URL: https://packaging.python.org/en/latest/guides/installing-using-pip-and-virtual-environments/ (Дата звернення: 17.05.2025).

35. Python Documentation. The pycache directory and .pyc files. *Python.org*. URL: https://docs.python.org/3/library/compileall.html#pycache-directories (Дата звернення: 17.05.2025).

36. Goli V. R. Cross-Platform Mobile Development: Comparing React Native and Flutter, and Accessibility in React Native. *International Journal of Innovative Research in Computer and Communication Engineering*. 2023. Vol. 11, no. 03. URL: https://doi.org/10.15680/ijircce.2023.1103002.

37. Kurapati, L. (2024). Micro Frontend Architecture in Web Applications. International Journal for Multidisciplinary Research.

38. Necula, S. (2024). Exploring The Model-View-Controller (MVC) Architecture: A Broad Analysis of Market and Technological Applications. URL: https://doi.org/10.20944/preprints202404.1860.v1.

39. Smith, A., & Gonzalez, M. (2023). Modernizing Web Applications with MVP and MVC Patterns. ACM Computing Surveys.

40. García, R.F. (2023). MVP: Model–View–Presenter. In: iOS Architecture Patterns. Apress, Berkeley, CA. URL: https://doi.org/10.1007/978-1-4842-9069-9_3

41. Sudip Chakraborty, & P. S. Aithal. (2023). MVVM Demonstration Using C# WPF. International journal of applied engineering and management letters (IJAEML), 7(1), 1–14. URL: https://doi.org/10.5281/zenodo.7538711

42. Code-Behind and XAML in WPF. URL: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/advanced/code-behind-and-xaml-in-wpf

ДОДАТКИ

ДОДАТОК А

Лістинг коду створення таблиць бази даних

```
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///project.db'
db = SQLAlchemy(app)


class Post(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.Text, nullable=False)
    description = db.Column(db.Text, nullable=False)
    image_path = db.Column(db.Text, nullable=False)
    date = db.Column(db.Date, nullable=False)
    organizer_id = db.Column(db.Integer, db.ForeignKey('user.id'),
nullable=False)
    is_military = db.Column(db.Boolean, default=False)


    organizer = db.relationship('User', backref='events')
    results = db.relationship('Result', back_populates='post',
cascade='all, delete-orphan')


class Registration(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    participant_name = db.Column(db.String(100), nullable=False)
    email = db.Column(db.String(120), nullable=False)
    phone = db.Column(db.String(20), nullable=True)
    post_id = db.Column(db.Integer, db.ForeignKey('post.id'),
nullable=False)
    post = db.relationship('Post',
backref=db.backref('registrations', lazy=True))
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'),
nullable=True)
    user = db.relationship('User', backref='registrations')
    result = db.relationship('Result',
back_populates='participant', uselist=False, cascade='all, delete-
orphan')
```

```
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(100), unique=True,
nullable=False)
    email = db.Column(db.String(150), unique=True, nullable=False)
    password = db.Column(db.String(200), nullable=False)
    role = db.Column(db.String(20), nullable=False)
    is_military = db.Column(db.Boolean, default=False)


class Result(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    participant_id = db.Column(db.Integer,
db.ForeignKey('registration.id'), nullable=False)
    post_id = db.Column(db.Integer, db.ForeignKey('post.id'),
nullable=False)
    result_data = db.Column(db.String(100), nullable=False)


    participant = db.relationship('Registration',
back_populates='result')
    post = db.relationship('Post', back_populates='results')
```

ДОДАТОК Б

Лістинг коду сторінки зі всіма змаганнями

```
{% extends 'base.html' %}

{% block title %}Всі змагання{% endblock %}

{% block body %}

<h1 class="mb-4">Всі змагання</h1>


<form method="get" action="{{ url_for('posts') }}" class="mb-3">

  <div class="form-check">

    <input class="form-check-input" type="checkbox"
name="military_only" id="military_only" value="yes" {% if
request.args.get('military_only') == 'yes' %}checked{% endif %}>

    <label class="form-check-label" for="military_only">Показати
тільки події для військових</label>

  </div>

  <button type="submit" class="btn btn-sm btn-outline-secondary
mt-2">Застосувати</button>

</form>


{% if posts %}

    <div class="row row-cols-1 row-cols-md-2 row-cols-lg-3 g-4">

        {% for post in posts %}

            <div class="col">

                <div class="card h-100 shadow-sm border-0 d-flex
flex-column">

                    <img src="{{ url_for('static',
filename=post.image_path) }}" alt="{{ post.title }}">

                    <div class="card-body d-flex flex-column flex-
grow-1">

                        <h5 class="card-title">

                            {{ post.title }}

                            {% if post.is_military %}

                                <span class="badge bg-danger ms-
2">Для військових</span>
```

```
                            {% endif %}
                        </h5>
                        <p class="card-text">{{
post.description[:100] }}{% if post.description|length > 100
%}...{% endif %}</p>
                        <div class="mt-auto">
                            <a href="{{ url_for('post_detail',
id=post.id) }}" class="btn btn-outline-primary w-100 mb-
2">Детальніше</a>
                            <span class="text-muted small d-block
text-center">📅 {{ post.date }}</span>
                        </div>
                    </div>
                </div>
            </div>
        {% endfor %}
    </div>
{% else %}
    <p>Немає доступних змагань.</p>
{% endif %}
{% endblock %}
```

ДОДАТОК В

Лістинг коду сторінки для створення змагання

```
{% extends 'base.html' %}

{% block title %}Додати запис для головної сторінки{% endblock %}

{% block body %}

<h1>Додавання запису</h1>


<form method="post" enctype="multipart/form-data" class="form-
control">

    <input type="text" name="title" placeholder="Введіть
заголовок" class="form-control"><br>

    <input type="text" name="description" placeholder="Введіть
опис події" class="form-control"><br>

    <input type="file" name="image_file" accept="image/*"
class="form-control"><br>

    <input type="date" name="date" class="form-control"><br>


    <div class="form-check">

        <input class="form-check-input" type="checkbox"
value="yes" name="is_military" id="is_military">

        <label class="form-check-label" for="is_military">Подія
для військових</label>

    </div><br>


    <button class="btn btn-success" type="submit">Додати</button>

</form>

{% endblock %}
```

ДОДАТОК Г

Лістинг коду back-end частини створення змагання

```
@app.route("/create", methods=['GET', 'POST'])
def create():
    if session.get('user_role') != 'organizer':
        return "Доступ заборонено"

    if request.method == 'POST':
        title = request.form['title']
        description = request.form['description']
        date_str = request.form['date']
        try:
            date = datetime.strptime(date_str, '%Y-%m-%d').date()
        except ValueError:
            return 'Невірний формат дати. Використовуйте YYYY-MM-
DD.'

        image_file = request.files['image_file']
        if image_file.filename == '':
            return 'Файл не вибрано'

        filename = secure_filename(image_file.filename)
        image_path = os.path.join('image', filename)
        image_path = image_path.replace("\\", "/")
        full_path = os.path.join(app.root_path, 'static',
image_path)
        image_file.save(full_path)

        organizer_id = session.get('user_id')

        is_military = request.form.get('is_military') == 'yes'
```

```
        post = Post(title=title, description=description,
image_path=image_path, date=date, organizer_id=organizer_id,
is_military=is_military)


        try:
            db.session.add(post)
            db.session.commit()
            return redirect('/')
        except:
            return 'Виникла помилка при додаванні запису'
    else:
        return render_template('create.html')
```

ДОДАТОК Д

Лістинг коду обробки форми реєстрації учасника на змагання

```
@app.route("/posts/<int:id>/register", methods=['GET', 'POST'])
def register(id):
    post = Post.query.get_or_404(id)
    user = None

    if 'user_id' in session:
        user = User.query.get(session['user_id'])

    if request.method == 'POST':
        try:
            if user:
                name = user.username
                email = user.email
                user_id = user.id
            else:
                name = request.form.get('participant_name')
                email = request.form.get('email')
                user_id = None

                if not name or not email:
                    flash("Будь ласка, заповніть усі обов’язкові
поля.")
                    return redirect(request.url)

            phone = request.form.get('phone', '')

            existing = Registration.query.filter_by(email=email,
post_id=post.id).first()
            if existing:
                flash("Ви вже зареєстровані на це змагання.")
```

```
                return redirect(url_for('post_detail',
id=post.id))

            new_reg = Registration(
                participant_name=name,
                email=email,
                phone=phone,
                post_id=post.id,
                user_id=user_id
            )

            db.session.add(new_reg)
            db.session.commit()

            flash("Реєстрацію успішно завершено!")
            return redirect(url_for('participants', id=post.id))

        except Exception as e:
            db.session.rollback()
            flash("Помилка при реєстрації: " + str(e))
            return redirect(request.url)

    return render_template("register.html", post=post, user=user)
```