МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
Оксана ШОВКОПЛЯС
(підпис)

грудня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 «Комп'ютерні науки»

освітньо-професійної програми «Інформатика»

на тему: Інформаційна технологія планування, координації та управління подіями й заходами.

здобувача групи ІН.м-34 Пономаренка Гліба Віталійовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Гліб Пономаренко
(підпис)

Керівник
асистент кафедри комп'ютерних наук, кандидат фізико-математичних наук
Ольга ШУТИЛЄВА

(підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Оксана ШОВКОПЛЯС
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістра

зі спеціальності 122 «Комп'ютерні науки», освітньо-професійної програми «Інформатика»
здобувача групи ІН.м-34 Пономаренка Гліба Віталійовича

1. Тема роботи: Інформаційна технологія планування, координації та управління подіями й заходами.
затверджую наказом по СумДУ від *«03» грудня 2024 року № 1257-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 06 грудня 2024 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Огляд технологій, що використовуються для вирішення поставлених задач 3) Розробка інформаційної технології планування, координації та управління подіями й заходами 4) Аналіз результатів.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проекту (роботи), із зазначенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
	Аналіз проблеми предметної області, постановка й формування завдань дослідження		
	Огляд технологій, що використовуються для прогнозування курсу валют		
	Розробка інтелектуальної системи з прогнозування курсу валют		
	Аналіз отриманих результатів		
	Оформлення пояснювальної записки до кваліфікаційної роботи		

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 75 стр., 45 рис., 1 додаток, 17 використаних джерел.

Обґрунтування актуальності теми роботи – Тема кваліфікаційної роботи є актуальною, оскільки присвячена розв'язанню важливої практичної задачі організації подій та заходів для розвитку культури суспільства, оптимізації існуючих процесів уникаючи розповсюдження інформації, використання якої становить небезпеку для життя людей.

Об'єкт дослідження — процес планування, координації та управління подіями й заходами

Предмет дослідження — Інформаційна технологія планування, координації та управління подіями й заходами.

Мета роботи — розробка інформаційної технологія планування, координації та управління подіями й заходами.

Методи дослідження — алгоритми координації та управління подіями й заходами використовуючи технології розробки веб-додатків.

Результати — розроблено інформаційну технологію, через яку користувачі створюють події та заходи, координують залученість відвідувачів, управляють деталями проведення заходів, інформування відвідувачів шляхом оголошень та прямих комунікацій. Проведено тестування розробки у режимі навантаження поток користувачів.

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ, КООРДИНАЦІЯ, УПРАВЛІННЯ, ПОДІЇ, ЗАХОДИ, PHP, LARAVEL, VUE.JS.

ЗМІСТ

ВСТУП

Розвиток технологічного процесу робить життя людей простішим, комфортнішим та безпечнішим. Завдяки новим технологія вирішення проблем стало набагато швидшим, враховуючи самі різні характери виникаючих проблем. Цьому сприяють технології, впровадження яких відбувається на рівні, як політичних, так і суспільних. Кожні дослідження, які проводяться на основі вивчення життя суспільства направлені не лише на спрощення процесів, але і на їх оптимізацію для подальшого покращення і розвитку.

Новітні технології відкривають людям нові можливості, якими вони можуть вільно скористатися. Однією з таких можливостей є організація та відкритий моніторинг культурних подій чи заходів. Раніше дошки оголошень на вулиці та буклети на касах театрів чи Домів Культури, зараз - новини, місцеві інформаційні групи та канали у соціальних мережах є основним джерелом інформації, звідки люди можуть дізнатися про скорі концерти, виступи у театрах, літературні вечори, тощо. Публікації оголошень є основним джерелом, звідки люди можуть дізнатися про подію, яка їх може зацікавити, на яку вони хотіли б піти.

З розвитком інтернету та бажанням передавати через нього більше, ніж загально-відому інформацію почали розвиватися форуми, сайти та соціальні мережі, де люди могли оприлюднювати інформацію стосовно локальних подій та заходів для заохочення людей приєднуватися та відвідувати їх, а також створювати локальні спільноти, щоб люди могли шукати нових друзів за інтересами.

Так сформувався термін «Спільноти» у соціальних мережах. Коли кожен може знайти будь-яку необхідну для нього інформацію лише у своєму пристрої, який має доступ до інтернету.

Мета: розробити інформаційну технологію планування, координації та управління подіями й заходами.

Об’єкт дослідження: планування, координації та управління подіями й заходами.

Предмет дослідження: інформаційна технологію планування, координації та управління подіями й заходами.

Для досягнення мети було сформовано ряд задач, які необхідно виконати. Результатом виконаних поставлених задач буде являти собою готова інформаційна технологія.

Задачі:

- Провести оцінку впровадження інформаційних технологій у сферу планування, координації та управління подіями й заходами;
- Провести детальний аналіз продуктів-аналогів та визначити їх функціональні можливості;
- Сформувати перелік функціональних вимог інформаційної технології;
- Розробити проект інформаційної технології;
- Розробити візуальну та функціональну частини додатку;

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Сучасний ринок відзначається великою кількістю подій: конференції, семінари, концерти, освітні курси, спортивні змагання тощо. З кожним роком зростає потреба в автоматизації процесів організації подій та реєстрації учасників. Враховуючи масштаб та частоту таких заходів, стає зрозумілим важливість інформаційної технології, яка дозволить зручно управляти всіма аспектами подій, знижуючи адміністративні витрати та підвищуючи задоволеність користувачів.

Основною проблемою в організації подій є складність управління інформацією про заходи та взаємодії з учасниками. Організаторам часто доводиться вручну вести облік місць, підтверджень участі, списків учасників, обробляти дані про доступні квитки та слідкувати за тим, щоб не було переповнення зали. Це потребує багато ресурсів та часу. Відсутність централізованої системи ускладнює контроль над організаційними аспектами та зменшує ефективність управління подією.

Для учасників основною проблемою є процес реєстрації та доступність інформації. Відвідувачам необхідно зручно переглядати доступні заходи, здійснювати пошук за категоріями, часом та датою, а також реєструватися на подію та бронювати місце. В умовах великих подій, де квитки можуть закінчитися швидко, є потреба у швидкій реєстрації, з можливістю бронювання місць онлайн без особистого відвідування. Туристи, які вирішили відвідати інше місто також мають змогу легко знайти подію чи захід згідно їх власних інтересів, вподобань та бажань без витрати великої кількості часу на моніторинг локальних груп у мережах та об’явах на місцевих сайтах. Додатково може виникнути потреба у пошуку компаньйона, з ким можна провести час.

Автоматизовані технології для управління подіями значно зменшують кількість рутинної роботи організаторів. Застосування технологій машинного навчання дозволяє передбачати популярність подій, розробляти стратегії залучення учасників та надавати персоналізовані рекомендації відвідувачам, що значно підвищує зацікавленість аудиторії.

Основні функції, які повинна включати інформаційна технологія, це можливість для організаторів створювати та редагувати події, управляти списками учасників і розподіляти зони для бронювання місць. Крім цього, система повинна забезпечувати зручний процес реєстрації для відвідувачів з функцією бронювання місць у режимі онлайн. Інтеграція сповіщень дозволить користувачам своєчасно отримувати повідомлення про події, нагадування та зміни в розкладі.

Технологія також має включати захищену базу даних для зберігання інформації про події, учасників та резервування місць. Потрібен модуль для адміністрування, який дозволить контролювати доступ до різних функцій та забезпечувати підтримку.

З появою соціальних мереж з’явилася можливість активно залучати відвідувачів та створювати спільноти навколо подій[1]. Інформаційні технології можуть інтегруватися з соціальними платформами для просування заходів та залучення цільової аудиторії. Технології також відслідковують реакції користувачів, дозволяючи організаторам коригувати свої підходи[2].

Розробка інформаційної технології для планування, координації управління подіями та бронювання місць є актуальним завданням, яке може суттєво підвищити ефективність організації подій, забезпечити зручність для користувачів та знизити витрати на управління процесами. Інноваційний підхід та чітке структурування функцій дозволять створити інтуїтивний та ефективний інструмент для сучасного ринку заходів[3].

Таким чином, до появи інформаційних технологій організація подій вимагала значно більше часу та зусиль, а також залежала від фізичних ресурсів і

обмеженої здатності до охоплення аудиторії. Інформаційні технології зробили цей процес набагато простішим, автоматизувавши управління подіями, розширивши канали інформування і надавши користувачам швидкий доступ до подій, що є невіддільною частиною сучасного життя.

1.2 Постановка задачі

За останні 4 роки у сучасному світі набули популярності ресурси онлайн-сповіщень та онлайн-комунікацій. Для того, щоб дізнатися про ту чи іншу подію, захід не обов’язково йти та перевіряти місцеві дошки для оголошень, новини.

Інформаційна технологія призначена для автоматизації процесів організації, координації подій (конференцій, концертів, семінарів) та управління бронюванням місць на заходи. Вона допомагає організаторам ефективно створювати події, управляти учасниками, а також дозволяє користувачам зручно реєструватися на заходи.

Метою розробки є створення універсальної інформаційної технології, яка оптимізує процеси координації, планування та управління заходами, спрощує бронювання місць та забезпечує користувачів зручним інтерфейсом для взаємодії з подіями.

1.3 Аналіз інформаційних технологій планування, координації та управління подіями й заходами

З набуттям великої кількості щоденних користувачів інтернету, бажаючих ділитися своїми враженнями, новинами та подіями, почали набувати популярність соціальні мережі, у яких люди відкривали для себе багато можливостей, для спрощення багатьох речей з повсякденного життя.

Прості соціальні мережі і стали прототипами інформаційних мереж для керування онлайн подіями та заходами.

Для аналізу веб-додатків для керування\пошуку подій та заходів було обрано «eventbrite.com», «ticketmaster.com», «meetup.com».

Розглянемо веб-додаток «eventbrite.com». Даний додаток являє собою платформу на якій можна подавати заявки на участь у вже існуючих запланованих заходах, або створювати власні заходи. Даний ресурс розповсюджений для користувачів країн ЄС та США.

На рисунку 1.1 зображено головну сторінку «eventbrite.com»[4], на які буде представлено оголошення про популярні оголошення, та кнопка для перегляду схожих заходів включаючи доступні кнопки для вибору категорії пошук подій за власними інтересами, а також доступним пошуком та навігацією у верхньому полі сторінки.

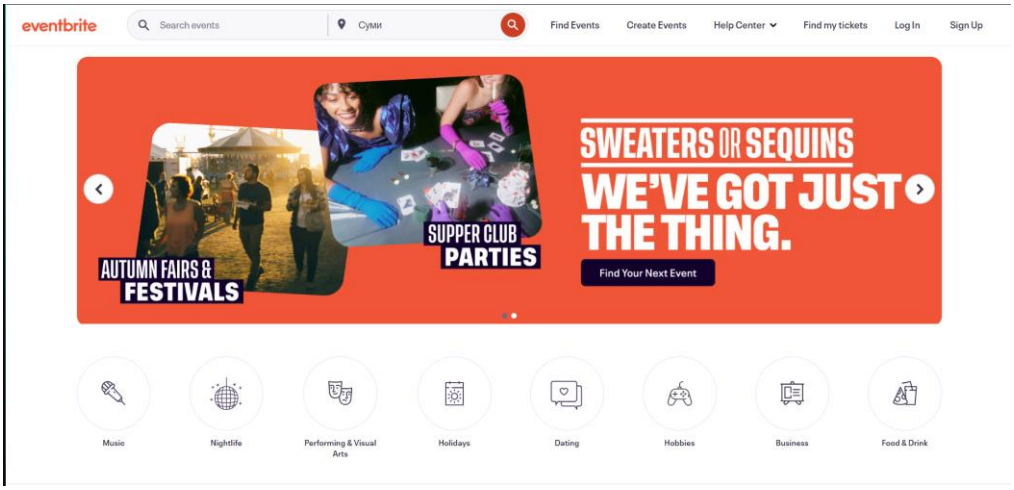

Рисунок 1.1 – Головна сторінка «eventbrite.com»

Нижче на головній сторінці додаток буде пропонувати заходи, які будуть проводитися у місті, яке сайт визначає за допомогою використання геоданих користувача, як зображено на рисунку 1.2.

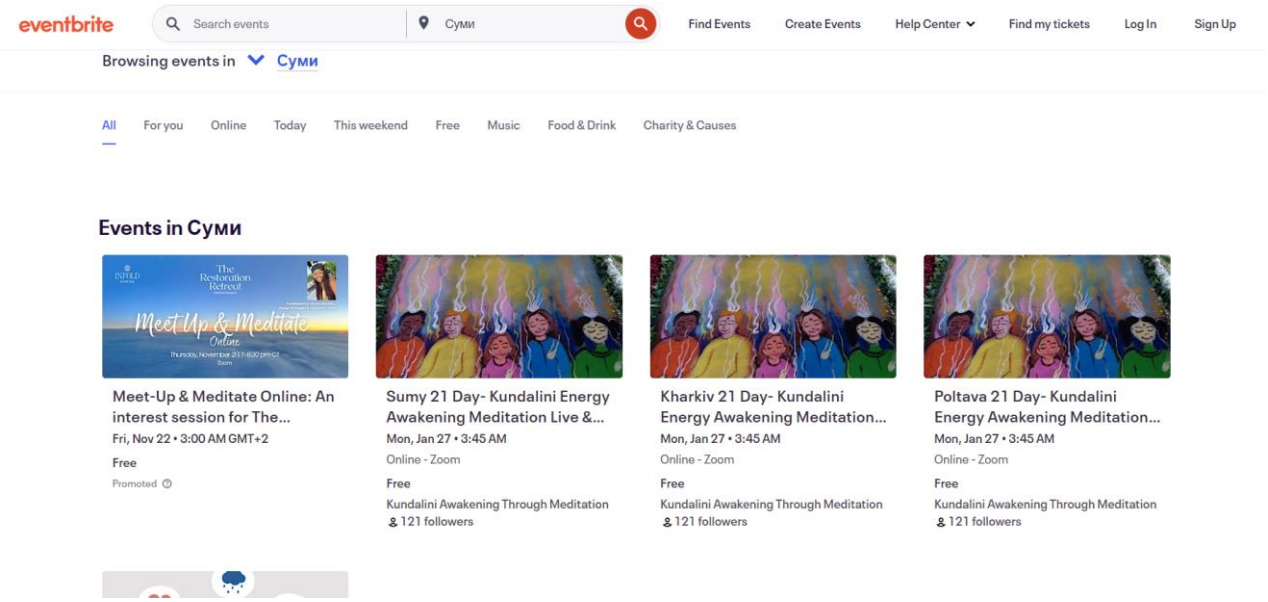

Рисунок 1.2 – Підбір заходів за геолокацією

При переході на сторінку пошуку заходів нас зустрічають оголошення про популярні заходи та події, а також список заходів які проводяться за визначеним містом, або онлайн, як зображено на рисунках 1.3-4.

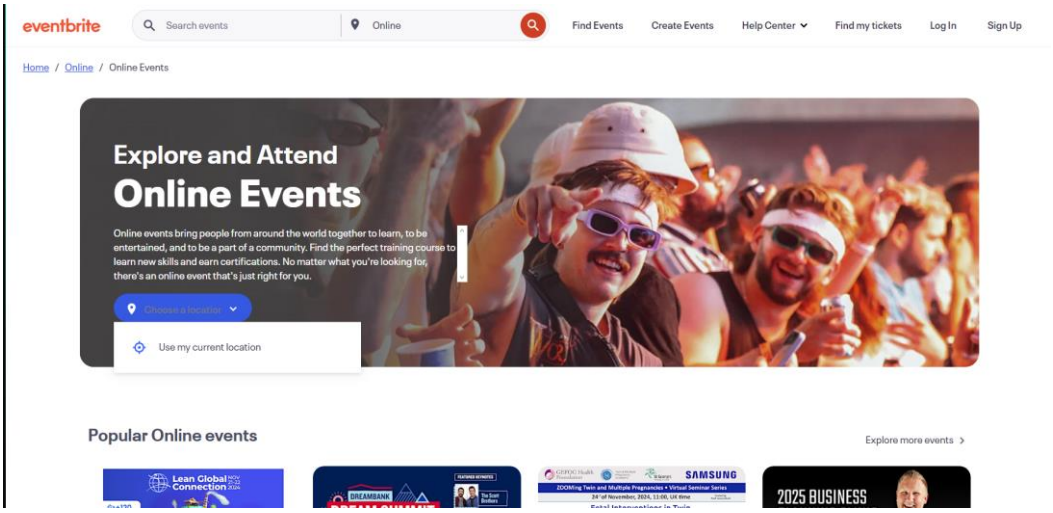

Рисунок 1.3 – сторінка пошуку подій та заходів

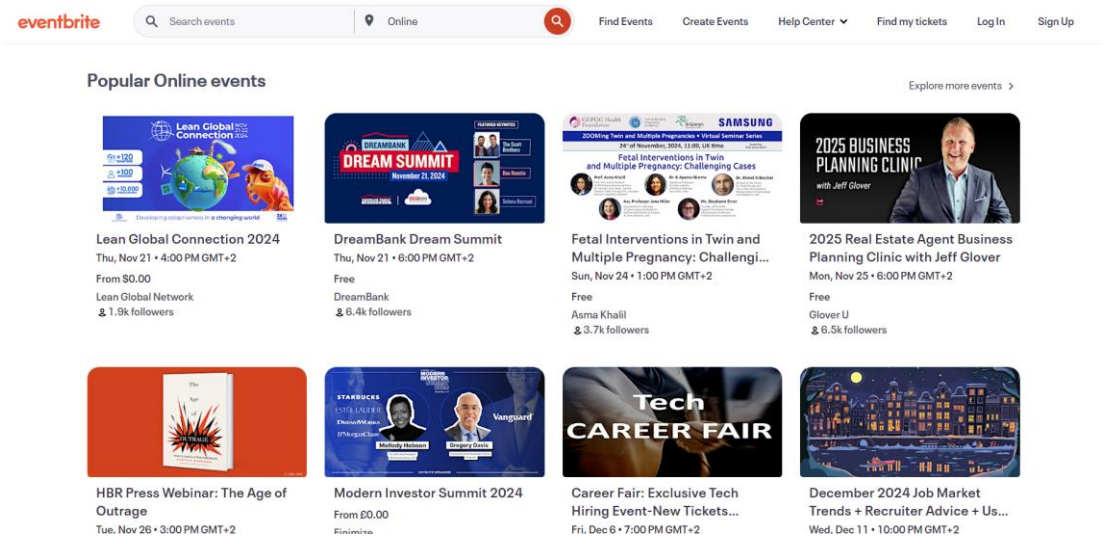

Рисунок 1.4 – Результат пошуку подій за визначеним містом

При переході на сторінку створення подій, користувачу відкривається панель керування організатора і функції створення заходів з детальним описом функцій, як зображено на рисунках 1.5-6.

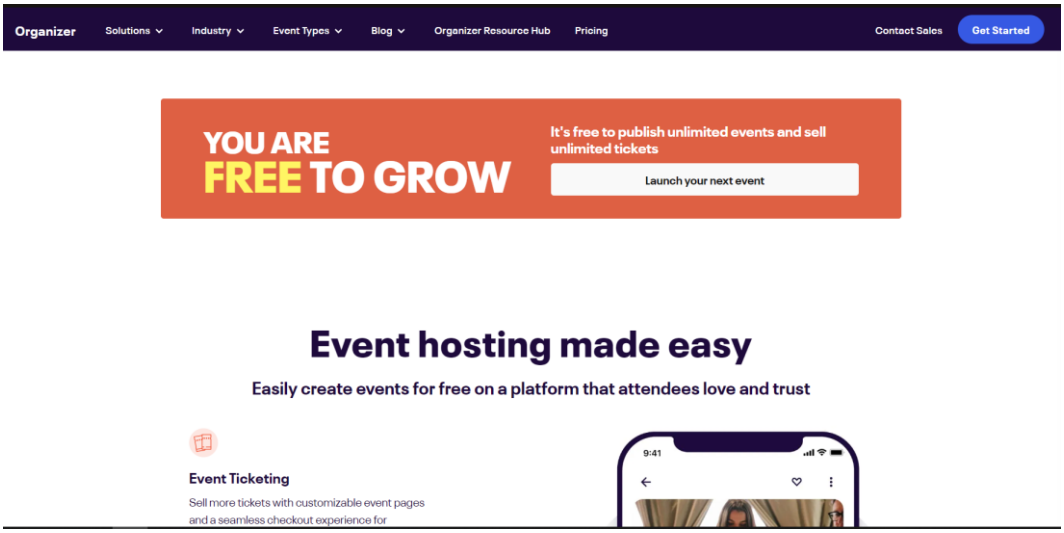

Рисунок 1.5 – Сторінка створення подій та заходів

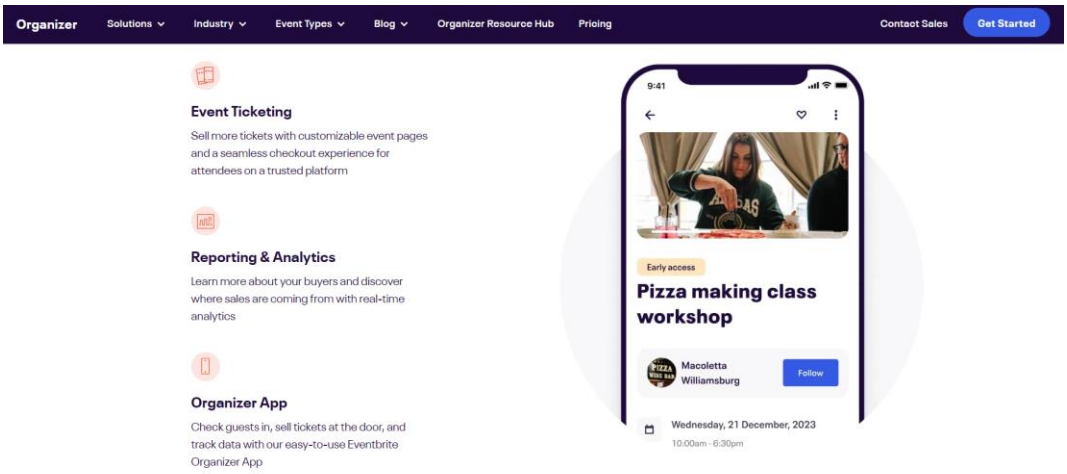

Рисунок 1.6 – Опис використання функціоналу

При виборі існуючих подій, перейшовши на головну сторінку заходу користувач спостерігає банер з виділеною інформацією, яка може одразу викликати зацікавленість та інтерес учасників, як зображено на рисунку 1.7.

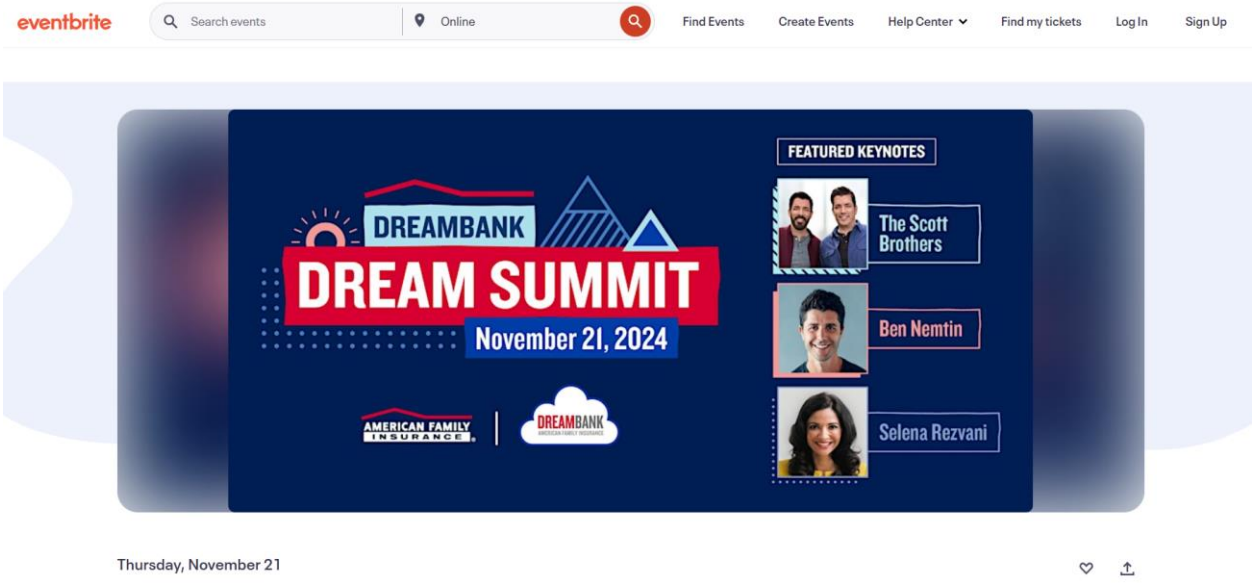

Рисунок 1.7 – Банер головної сторінки заходу

Нижче користувач може бачити всю необхідну інформацію про проведення події, а саме, дата, час та місце проведення, з подальшою можливістю слідкування за даною подією та реєстрацією, як зображено на рисунку 1.8.

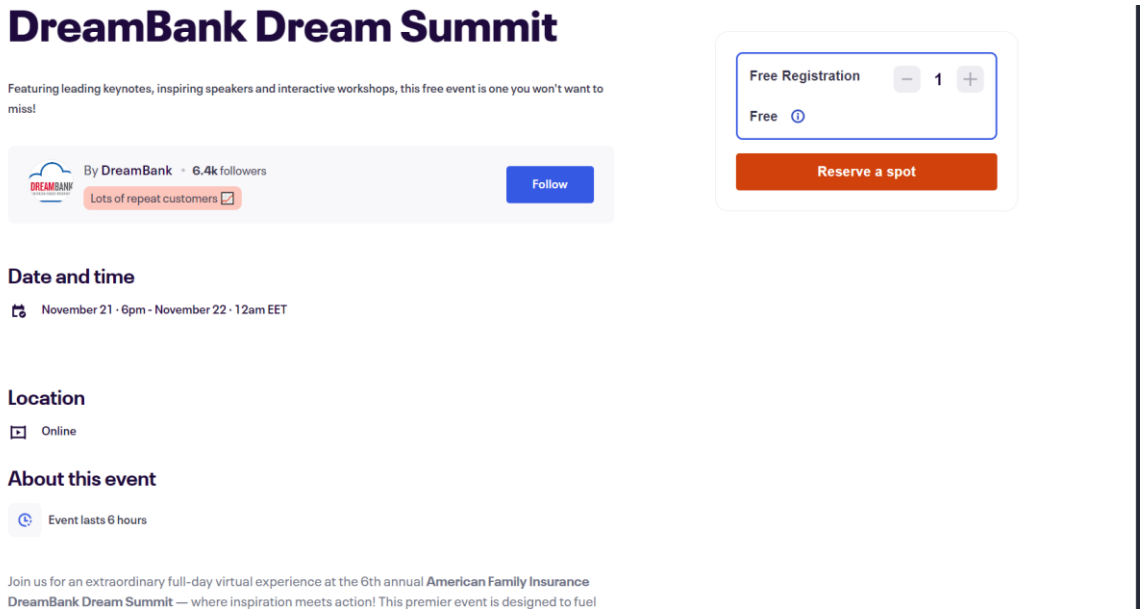

Рисунок 1.8 – Детальна інформація про захід та подію

Для даного виду інформації є уразливим розповсюдження детальної інформації для неавторизованих користувачів, що може поставити під загрозу безпеку учасників.

Для подальшої реєстрації користувачу необхідно заповнити анкету з особистими контактними даними та ознайомитись з політикою конфіденційності. Також користувач може отримати виписку про оплату, якщо така вимагається, як зображено на рисунку 1.9.

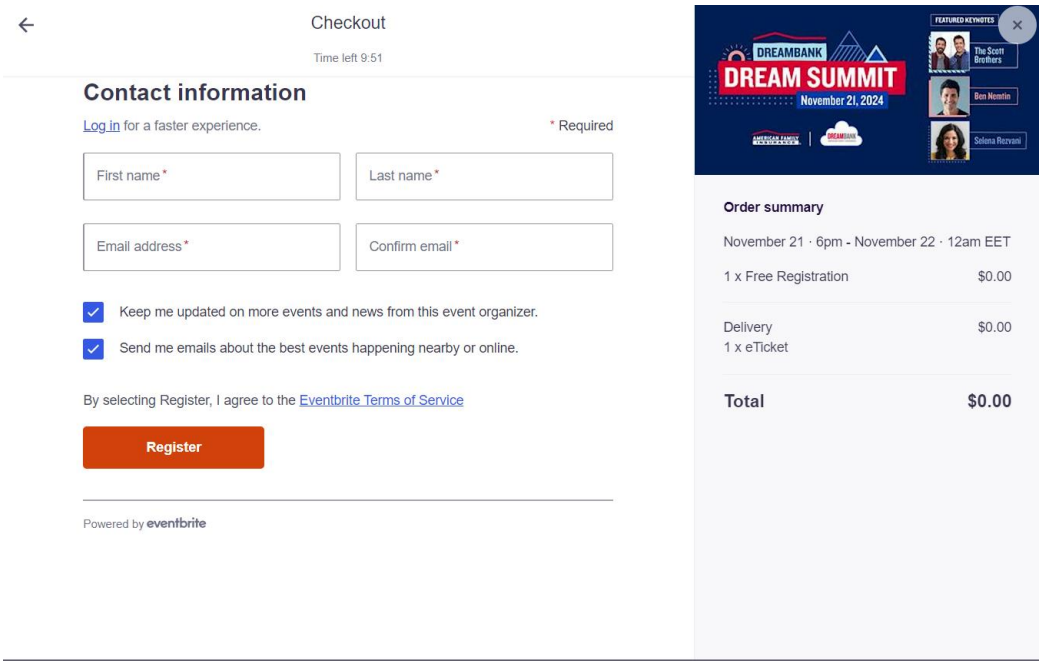

Рисунок 1.9 – Форма заповнення даних та виписка оплати

Розглянемо додаток «meetup.com», який являє собою веб-додаток для пошуку подій та заходів[5]. На головній сторінці ресурсу користувач може ознайомитись із загальною відомістю про веб-додаток та одразу ж зареєструватися на подію, як зображено на рисунку 1.10.

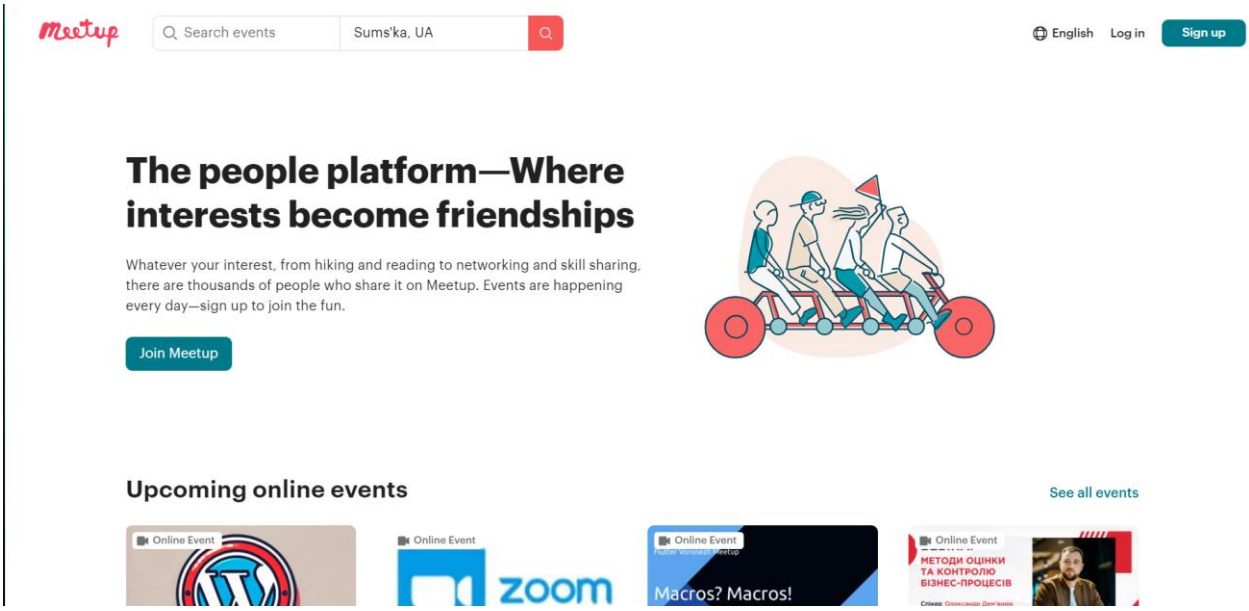

Рисунок 1.10 – головна сторінка «meetup.com»

Додатково на головній сторінці користувач може обрати для себе категорію інтересів та переглянути події, які підходять під обрану категорію, а також обрати місто, в рекомендаціях відображено популярні міста, користувач може обрати локацію власноруч, рисунок 1.11.

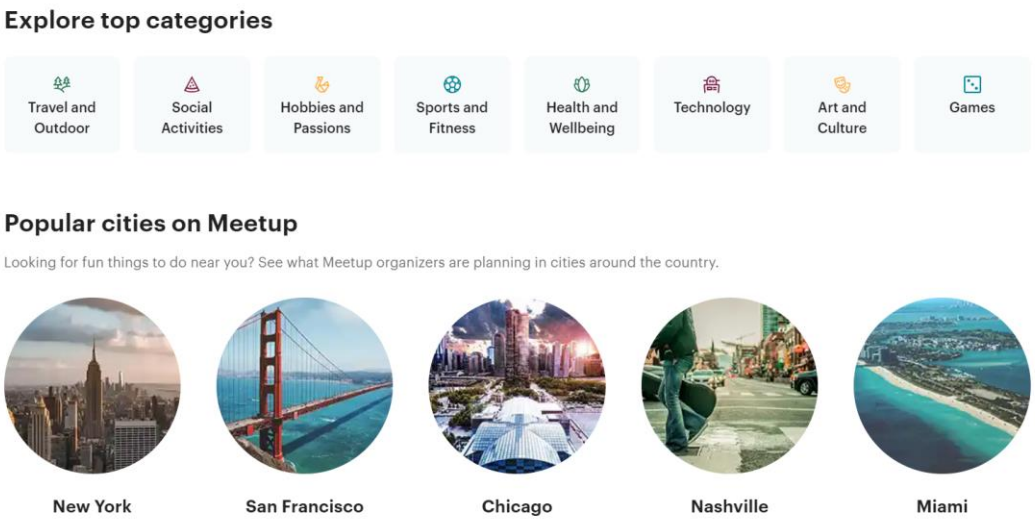

Рисунок 1.11 – Рекомендація категорій та популярні міста

Обравши категорію подій, користувач може переглянути список груп, які були створені тематично за інтересом, а також може створити власну групу, як зображено на рисунку 1.12.

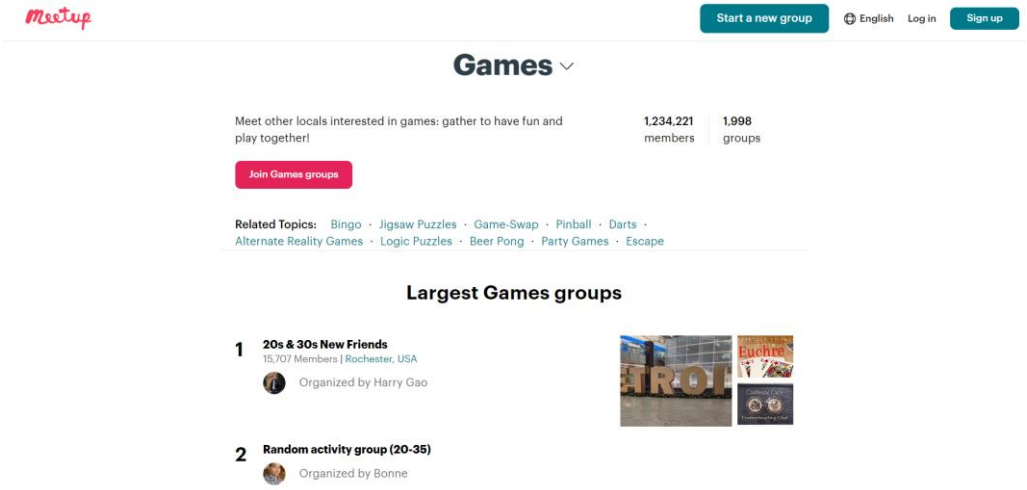

Рисунок 1.12 – Сторінка пошуку груп за інтересами

Створювати групи та події можуть тільки зареєстровані та авторизовані користувачі, як зображено на рисунку 1.13.

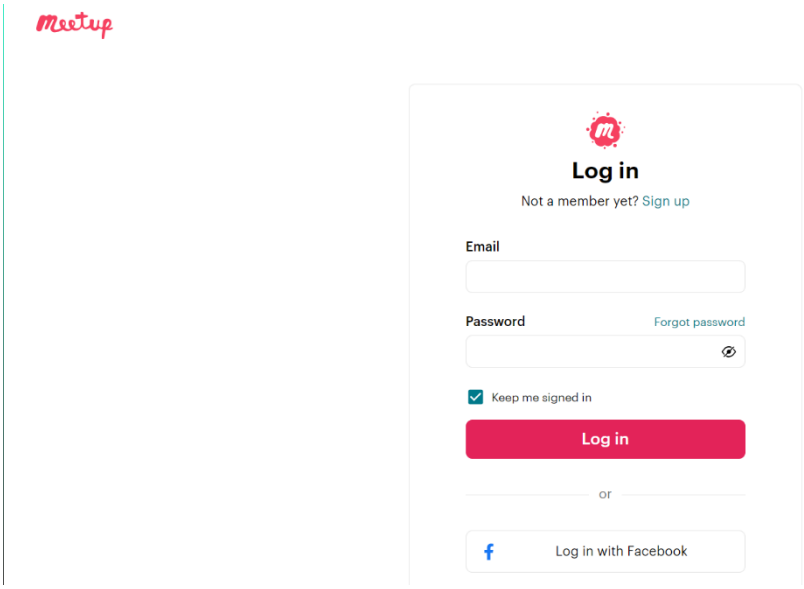

Рисунок 1.13 – Форма авторизації користувача для створення груп

При виборі обраної групи, користувач переноситься на головну сторінку групи, де він бачить загальні відомості про групу, список учасників(загроза конфіденційних даних), підключені посилання на соціальні мережі, та події, які вже відбулися, та відбудуться з даними про місце та час проведення, як це зображено на рисунках 1.14-15.

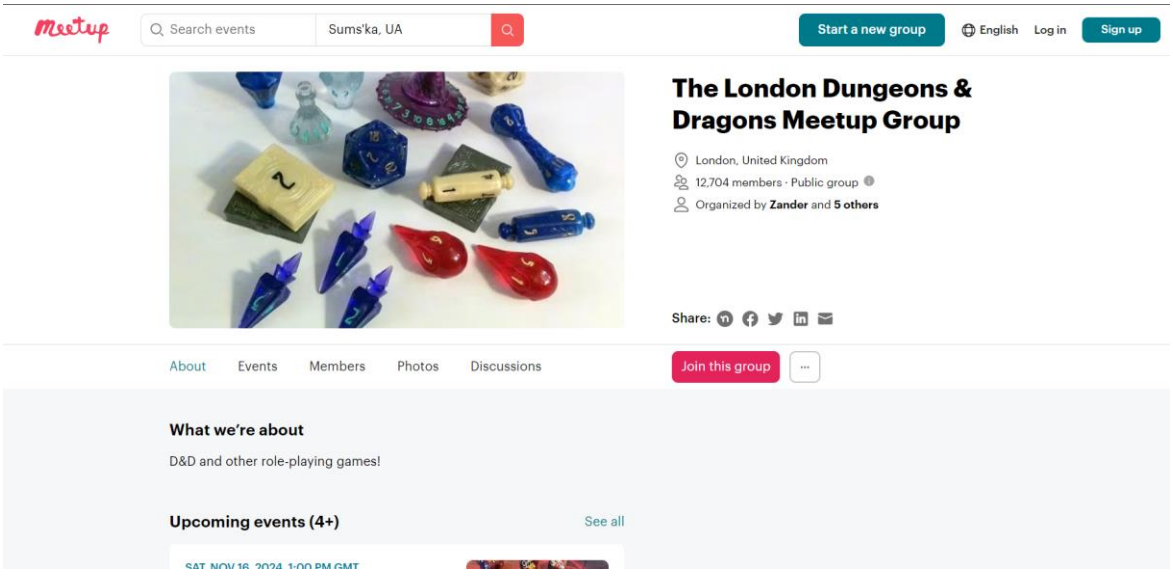

Рисунок 1.14 – Головна сторінка групи

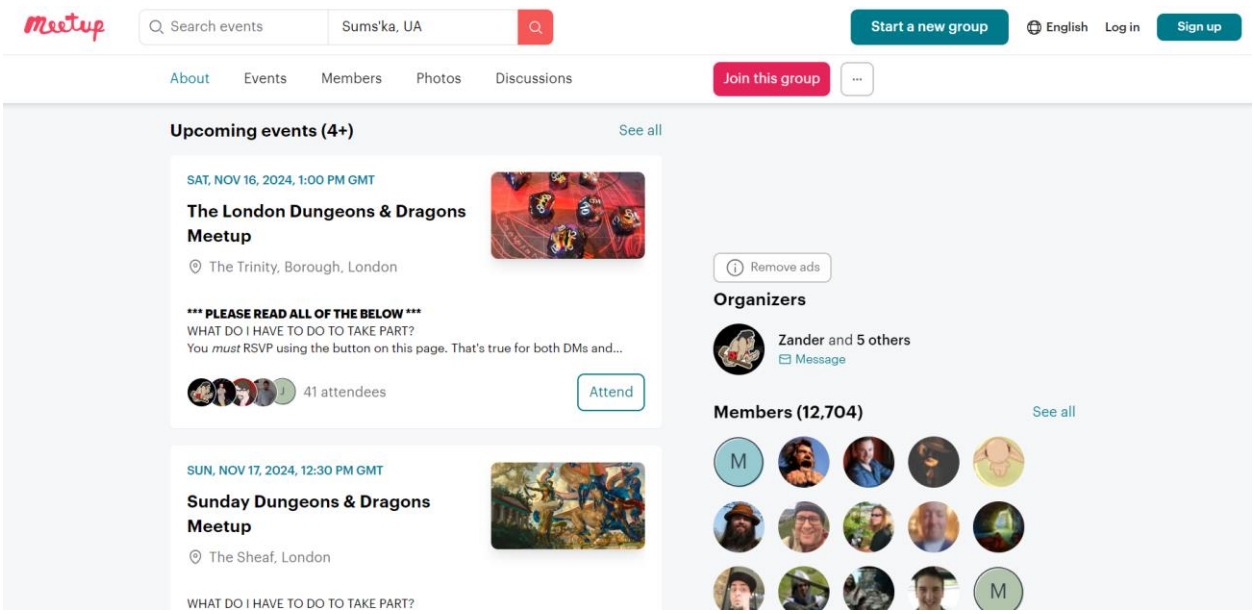

Рисунок 1.15 – Оновлення про майбутні події у групі

Для зручності інформування та швидкого планування подій та заходів, якщо їх планується декілька – є можливість використання календаря та перегляду розкладу подій на ньому, як це зображена на рисунку 1.16.

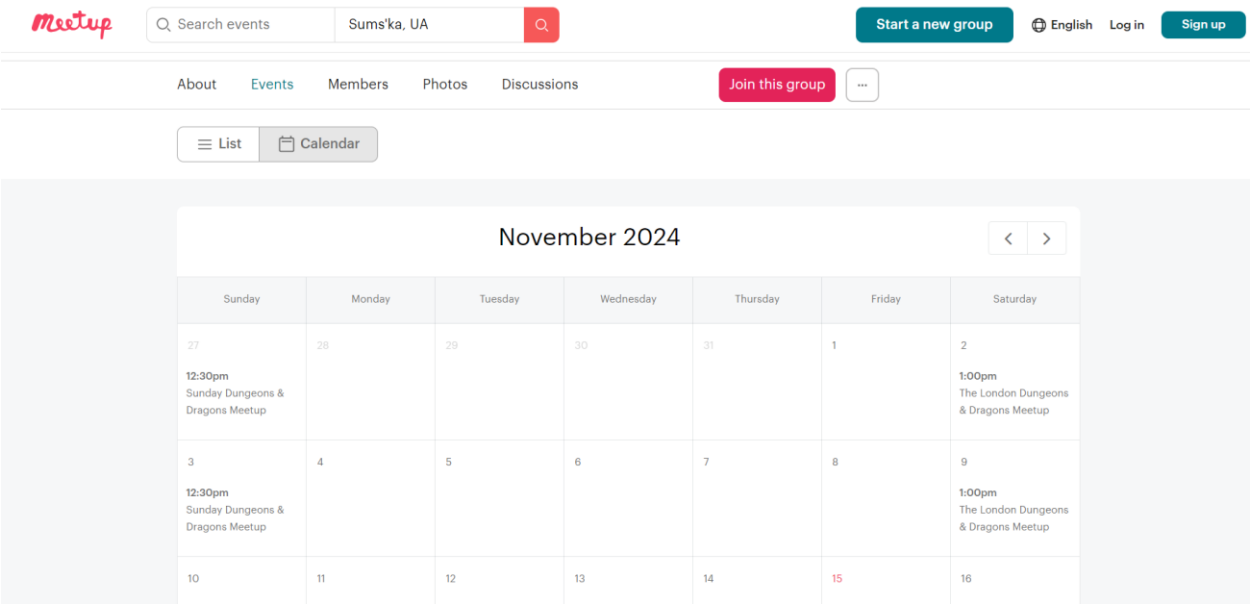

Рисунок 1.16 – Календар подій групи

2 ВИБІР МЕТОДУ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1 Структура та функції інформаційної технології

Для формування структури інформаційної технології планування, координації та управління подіями й заходами потрібно створити вимоги до функціоналу. Основний функціонал базується на тому, що за допомогою даного сервісу можна буде створювати заходи, а також реєструватися на вже існуючі. До переліку функцій технології відносяться:

1. Створення подій;
2. Управління подіями;
3. Перегляд доступних подій;
4. Управління списком учасників;
5. Пошук та фільтрація заходів;
6. Реєстрація на подію;
7. Отримання сповіщень про оновлення статусу подій;
8. Пошук друзів;
9. Знайомства;
10. Спілкування з учасниками;
11. Пошук компанії;
12. Редагування персональних даних;

Окрім того, при розробці інформаційної технології планування, координації та управління подіями й заходами буде виконуватися розподілення на користувачів, а саме організатори, учасники та адміністраторів. Виконаємо розподіл функцій за групами користувачів.

Варіації функцій для організаторів:

- Створення та видалення подій.
- Визначення кількості місць;
- Прийом заявок учасників;
- Видалення учасників;
- Редагування створених подій чи заходів;
- Можливість надсилань оновлень учасникам

- Чат з іншими організаторами/учасниками
- Написання постів

Варіації функцій для відвідувачів:

- Перегляд заходів;
- Реєстрація на заходи;
- Пошук друзів;
- Знайомства;
- Надсилання повідомлень;
- Запрошення друзів;

Варіації функцій для адміністраторів:

- Блокування користувачів;
- Редагування загальної інформації;
- Редагування всіх дисциплін;

Наступним етапом є розроблення структури інформаційної технології планування, координації та управління подіями й заходами. Детальний опис кожної із сторінок представлений в таблиці 1.1.

Таблиця 1.1 – Таблиця із описом структури інформаційної системи

Назва сторінки	Опис
Форма реєстрації	Форма складається з полів: - Прізвище - Ім’я - Ім’я По-батькові - Дата народження - Контактний номер телефону - Роль - Email - Пароль користувача - Підтвердження паролю - Кнопка «Реєструватися»

Форма авторизації	Форма складається з полів: - Email - Пароль - Підтвердження паролю - Кнопка «Вхід» - Кнопка «Відновлення паролю»
Форма відновлення паролю	Форма складається з: - ПІБ користувача - Email на який зареєстровано користувача
Головна сторінка	Містить: - Стрічка оновлень (для кожної ролі своя)
Заходи	Містить: - Список заходів у яких користувач бере участь - Створення нових заходів (для організаторів) - Реєстрація на існуючі заходи;
Створити поді\захід	Містить: - Назва події\заходу - Опис - Кнопка «Додати матеріали»
Сторінка обраної події	Містить: - Опис події; - Відомості (час, дата, місце, кількість місць)

	- Фото; - Кнопка «Подати заявку » (для учасників)
Додати	Містить: - Обрати дисципліну для якої буде заняття - Дату заняття - Час проведення
Друзі	Містить: - Пошук нових друзів; - Фільтр для пошуку - «Мої друзі» - Кнопка «відкрити чат» - Кнопка «видалити друга»
Підбір друга	Містить: - Фільтр для пропозицій; - Картки з пропозиціями друзів; - Кнопка «Так»; - Кнопка «Ні»
Особиста сторінка	Містить: - Зазначені при реєстрації ПІБ - Номер телефону - Email - Фото профілю

Отже, результатом роботи є розроблена структура інформаційної технології планування, координації та управління подіями й заходами. Також був сформований список, представлений у таблиці, із типів користувачів та переліком відкритих функцій з розподілом до відповідно визначеної ролі.

2.2 Вибір програмних засобів реалізації інформаційної технології

Під час створення візуальної частини IT (фронт-енд) та оформлення її зовнішньої оболонки були використані такі технології, як: HTML, CSS, JavaScript , зокрема, фреймворк Vue.js.

Відомо, що HTML являє собою стандартизованим стилем гіпертекстової розмітки. Вона застосовується для перегляду веб-сторінок у браузерах, які, у свою чергу, інтерпретують код файлу і відображають його у візуальному інтерфейсі на пристроях, з яких здійснюються запити[7]. Використовуючи HTML можна створити досить простий, але стандартизовано оформлений документ. Дана технологія використовується у якості основного інструменту для створення макету інформаційної технології.

Окрім HTML було використано CSS для опису дизайну сторінки[8]. Даний інструмент дозволяє власноруч виконувати налаштування стилю шрифтів, встановлювати спеціальні кольори елементів та визначати розташування об'єктів на екрані.

JavaScript є об'єктно-орієнтованою прототипною мовою програмування. Ця мова використовується для реалізації сценаріїв взаємодії користувача з серверною та інтерфейсною частинами веб-сторінки.

Vue.js є JavaScript-фреймворком, призначеним для створення інтерфейсів користувача. Основною метою Vue.js є спрощення та надання готових рішень для швидкої розробки складних інтерфейсів за допомогою використання компонентів[9]. Його легко налаштовувати та використовувати у проектів, що дає можливість швидко отримати бажані результати.

Даний фреймворк виділяється швидкістю своєї роботи, що позитивно впливає на загальну роботу проекту[10]. Не менш важливою властивістю Vue.js є легкість у розгортанні та написанні невеликих додатків. Даний фреймворк ідеально підходить для розробки інформаційної технології. Адже, код фреймворку має гарну читабельну здатність і однофайлові компоненти, що

робить зручним можливість швидко та просто розібратися у коді та спрощує процес демонстрації його використання.

Для написання бекенд-частини проекту були використані PHP та фреймворк Laravel.

Мову було обрано, як основу функціональної частини проекту. Даний вибір був сформований на основі того, що PHP є однією з найнадійніших для створення інформаційних технологій[11]. Крім того, робота з базами даних на мові PHP є однією із найпростіших у порівняннями з іншими мовами, так як PHP має власні вбудовані модулі[12].

Laravel представляє PHP фреймворк з відкритим кодом. Головне його призначення - це розробка веб-сторінок за шаблоном Model-View-Controller[13]. Обраний фреймворк має найрізноманітніші плагіни та додатки. У сукупності доповнення допомагають у розгортанні додатків, технічному обслуговуванні, а також використання різних баз даних. Даний фреймворк використовує велику варіативність шаблонів, які значно скорочують час написання коду. Про це свідчить його використання при написанні різноманітних методів для реалізації функціоналу інформаційних систем, наприклад фунціонал авторизації та реєстрації користувачів.

Крім того, Laravel сповнений значною кількістю об’єктно-орієнтованих бібліотек, які вагомо розширюють функціонал PHP[14]. Дане розширення також дає можливість розширити функціонал веб-додатку. Також завдяки даному фреймворку можна реалізувати системну роботу черг повідомлень з метою підтримки балансу навантаження без втрати даних.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Розробка дизайну

Дизайн був розроблений в додатку Figma. Макети представленні у додатку на рисунку 2.2-7.

Figma використовує як інструмент в найновіших та найкращих практиках дизайну, таких як інтерфейсу користувача та дизайну користувацького досвіду чи іншого. У додатку Figma було розроблено дизайн сторінок інформаційної системи.

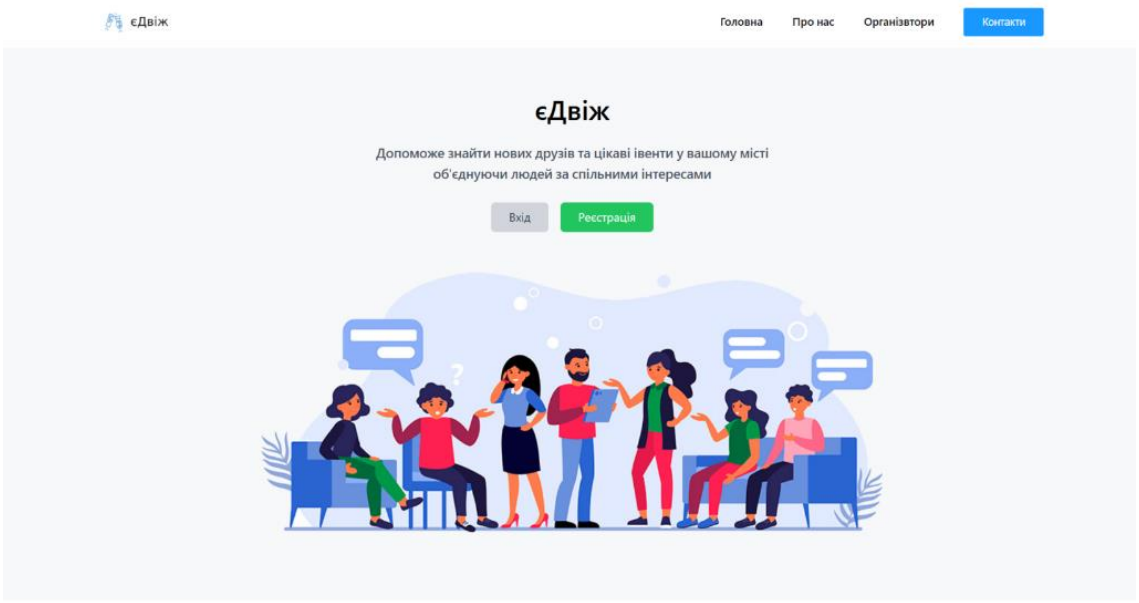

Рисунок 3.1 – Головна сторінка

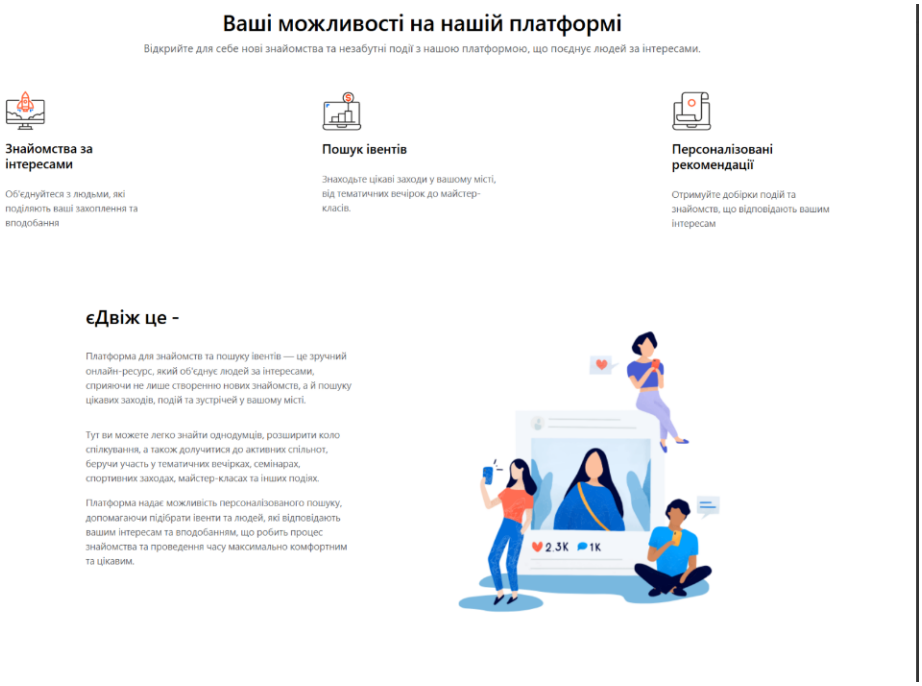

Рисунок 3.2 – Сторінка із додатковою інформацією

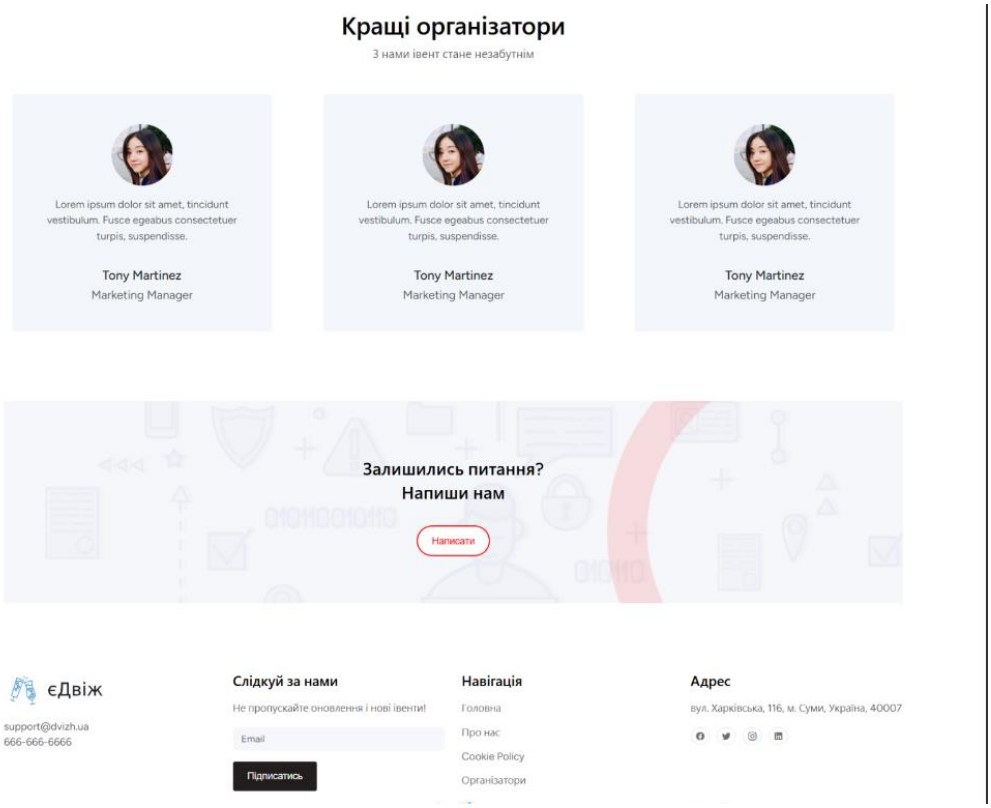

Рисунок 3.3 – Контактна інформація

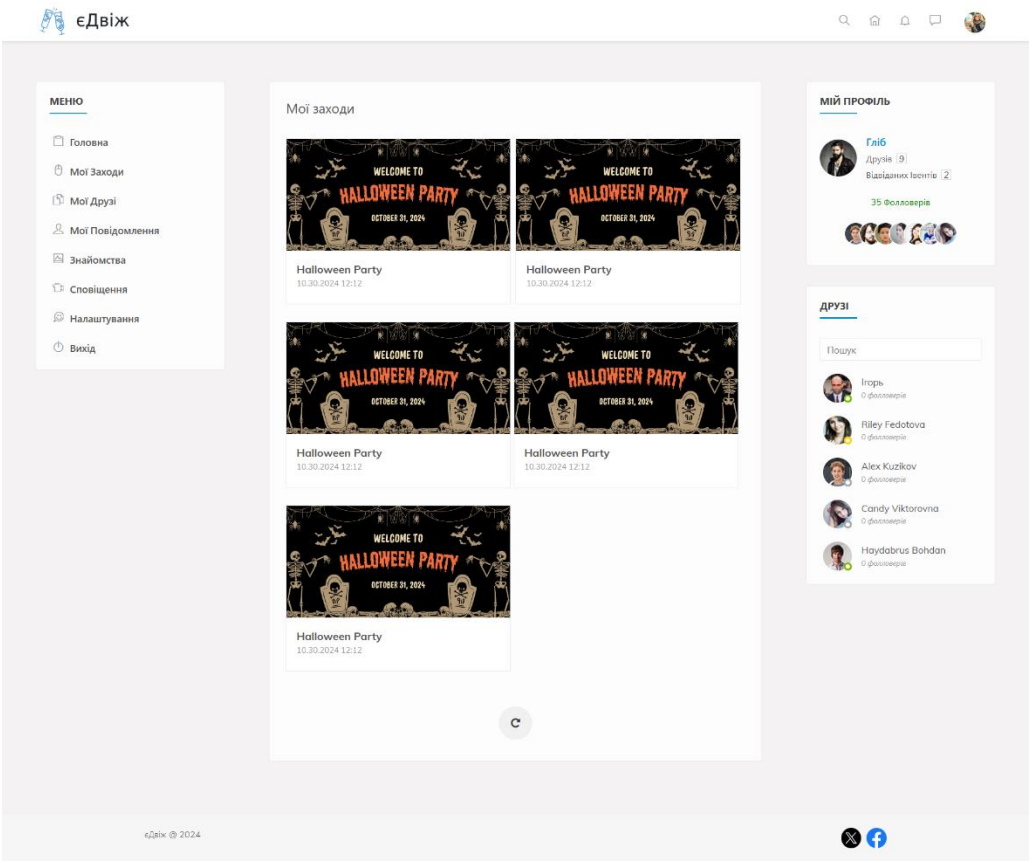

Рисунок 3.4 – Сторінка авторизованого учасника

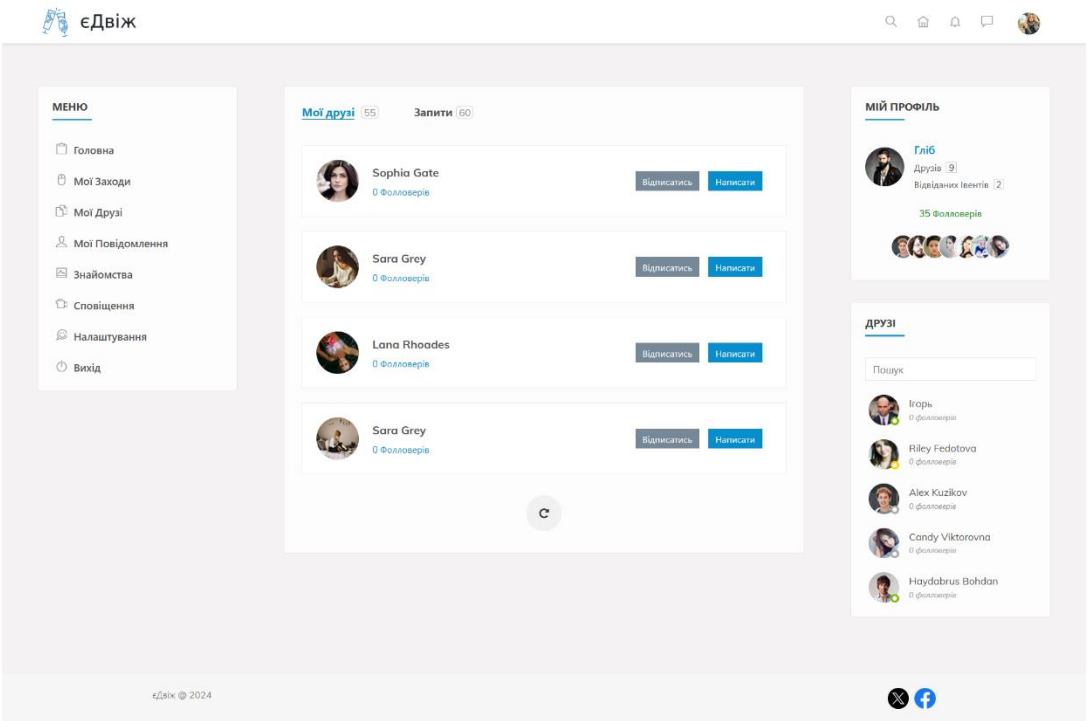

Рисунок 3.5 – Вкладка «друзі»

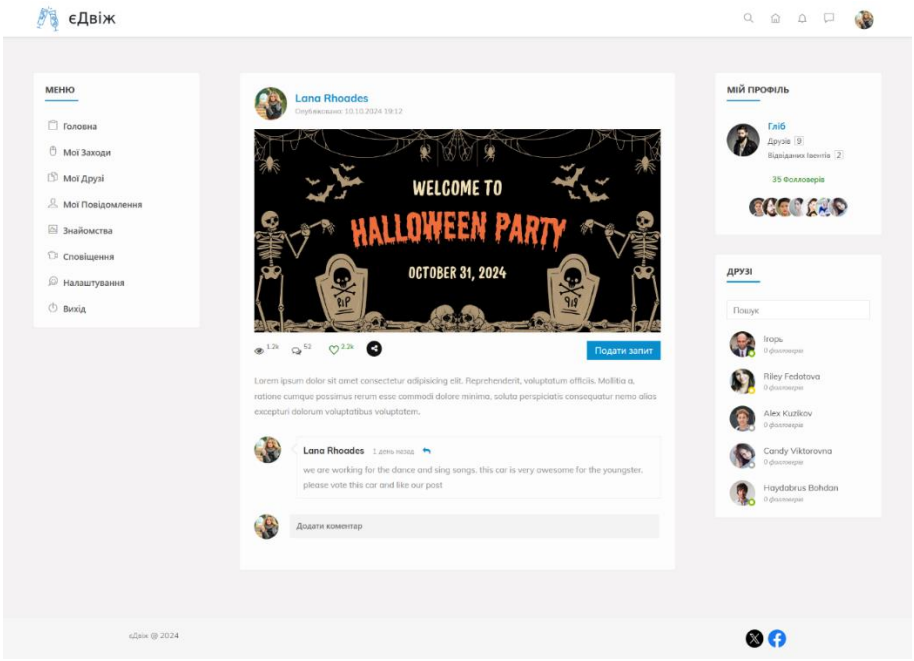

Рисунок 3.6 – Головна сторінка заходу

Як результат, завдяки інструменту Figma розроблено власний унікальний дизайн головних частин сторінок інформаційної технології планування, координації та управління подіями й заходами відповідно до розробленого функціоналу для взаємодії з користувачем.

3.2 Програмна реалізація інформаційної технології планування, координації та управління подіями й заходами

Програмна реалізація інформаційної технологія планування, координації та управління подіями й заходами виконана за допомогою використання фреймворків Laravel та Vue.js. На рисунку 3.8 зображено пакети та модулі, які створено в процесі реалізації проекту. Детальний код реалізації знаходиться у Додатку А.

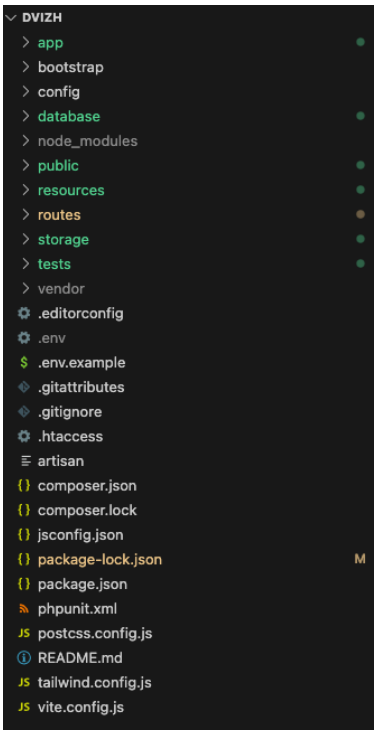

Рисунок 3.8 – інформаційної технологія планування, координації та управління подіями й заходами

Для інформаційної технології, відповідно до розробленої моделі функціоналу розроблено відповідну базу даних[15]. Дані будуть надалі оброблюватися відповідно до використання функцій інформаційної технології

На рисунку 3.6 зображено ERD-діаграму, де зображено модель бази даних і її таблиці.

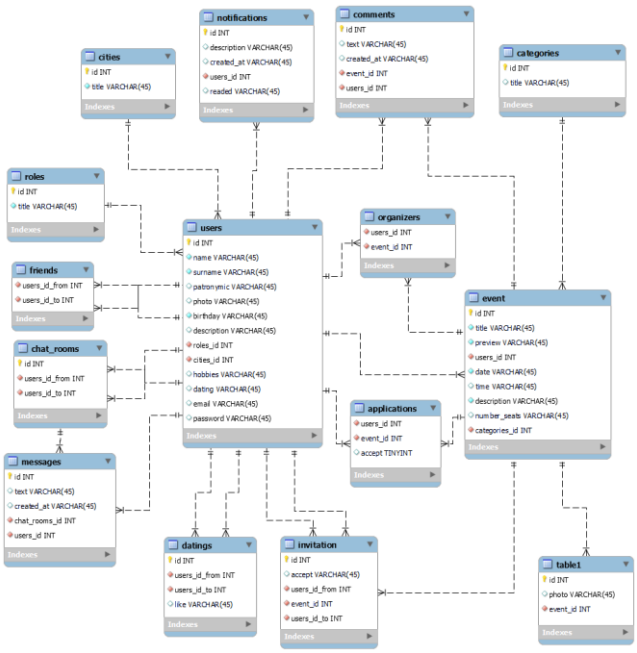

Рисунок 3.9 – ERD-діаграма бази даних інформаційної технології планування, координації та управління подіями й заходами

У реалізації роботи було розроблено відповідну до вимог та функціоналу базу даних, яка містить в собі 15 таблиць. Надалі ці таблиці будуть використовуватися для збереження в собі необхідних даних про користувачів, подій та заходів.

3.3 Використання інформаційної технології планування, координації та управління подіями й заходами зі сторони користувача

Головна сторінка інформаційної технології являється інформативною, щоб користувач міг ознайомитися з додатком, який він буде використовувати. Для цього у хедері є додаткові кнопки, а також на головній сторінці прописано детальну інформацію з можливістю авторизації або реєстрації, як зображено на рисунках 3.10 та 3.11.

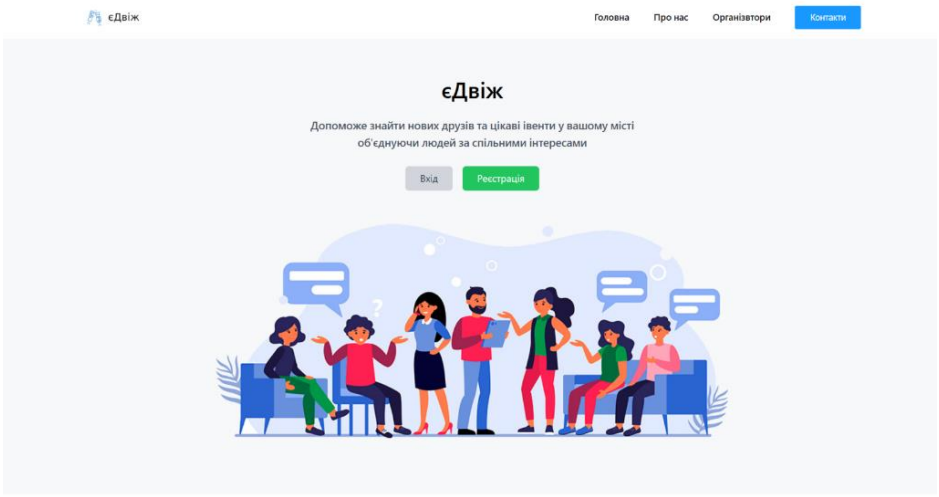

Рисунок 3.10 – Головна сторінка

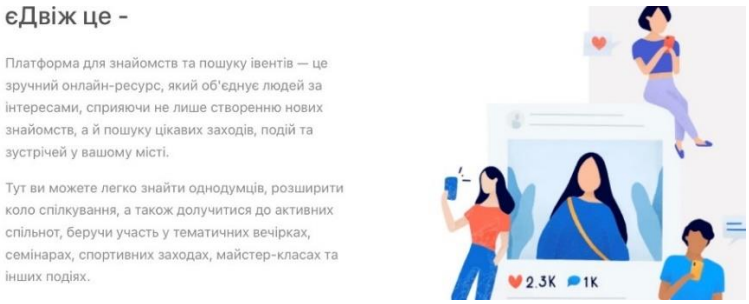

Рисунок 3.11 – Інформація про технологію на головній сторінці

Коли користувач вперше ознайомився із інформаційною технологією за описом на головній сторінці, обравши функцію «реєстрація», його буде на правлено на форму створення особистої сторінки, з можливістю вибору ролі «відвідувач», чи «організтор» (рис. 3.12).

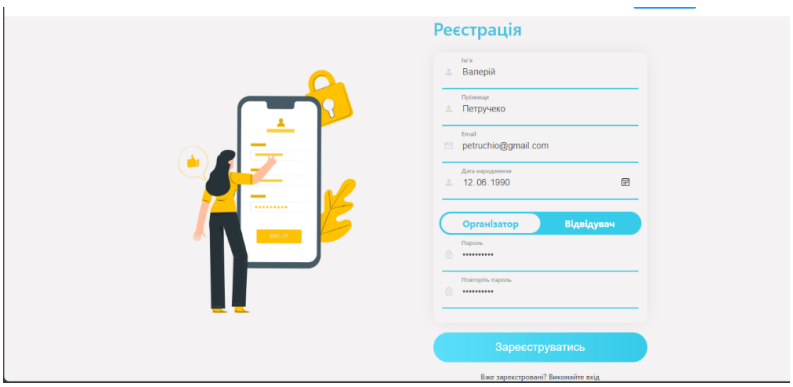

Рисунок 3.12 – Форма реєстрації користувача

Після проведення авторизації або реєстрації, користувач потрапляє на головну сторінку інформаційної технології, де відображаються вже створенні активні події з можливістю здійснювати пошук, використовуючи фільтри та введення деталізованої інформації. Також користувачу буде запропоновано подати запит на вже створені події та заходи, шляхом кнопки «подати запит», як це зображено на рисунку 3.13.

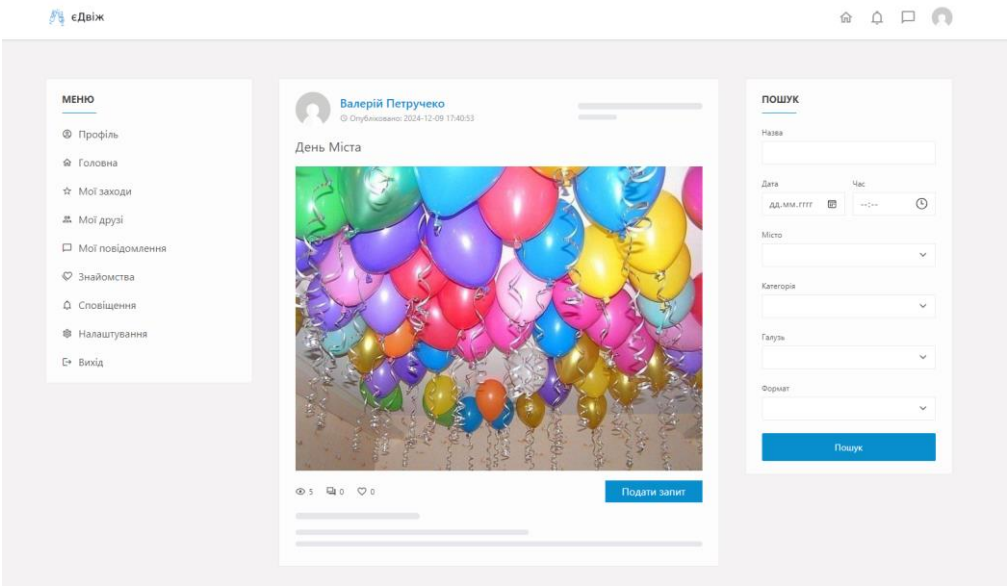

Рисунок 3.13 – Головна сторінка авторизованого користувача

З метою забезпечення безпеки відвідувачів подій та заходів, користувачі які не отримали погодження запиту на участь у події не будуть бачити деталі проведення заходів. Користувачі, участь яких була підтверджена організатором отримають всю інформацію про деталі події, як зображено на рисунку 3.14.

Рисунок 3.14 – детальна інформація про подію після погодження участі

Користувач, який отримав погодження на участь у події чи заході отримує сповіщення та подальші оновлення по подіям у вкладці «Сповіщення», як зображено на рисунку 3.15.

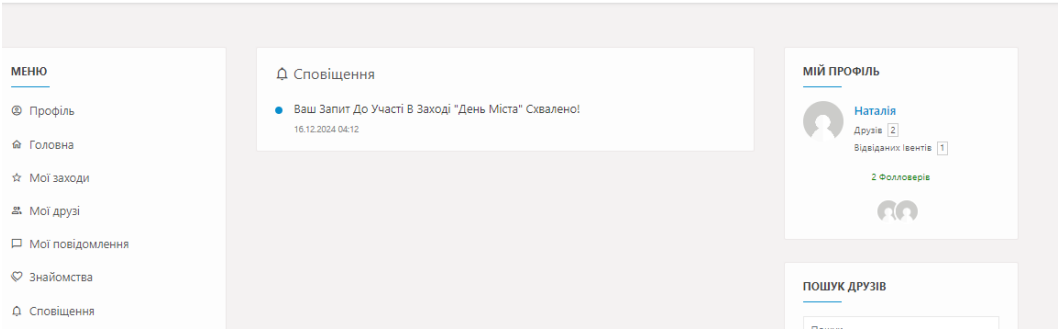

Рисунок 3.15 – Сторінка «Сповіщення»

Авторизований користувач може переглядати особисті дані свого профілю, що були вказані при реєстрації, переглядати івенти у яких бере участь, додавати опис своєї сторінки та обкладинку профілю, як це зображено на рисунку 3.16.

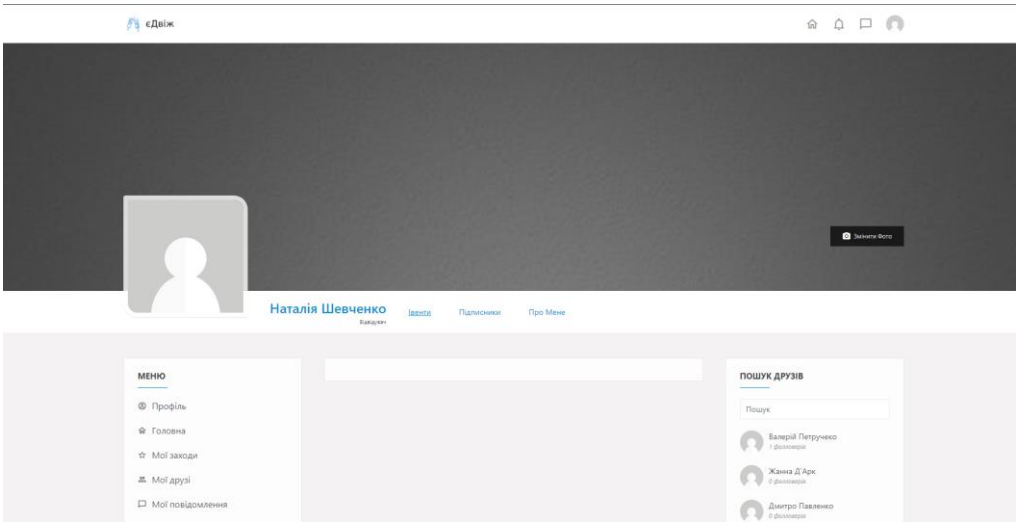

Рисунок 3.16 – Сторінка профілю авторизованого користувача

Авторизований учасник може переглядати список подій у яких бере участь та отримав підтвердження на участь у вкладці «мої заходи», як це зображено на рисунку 3.17.

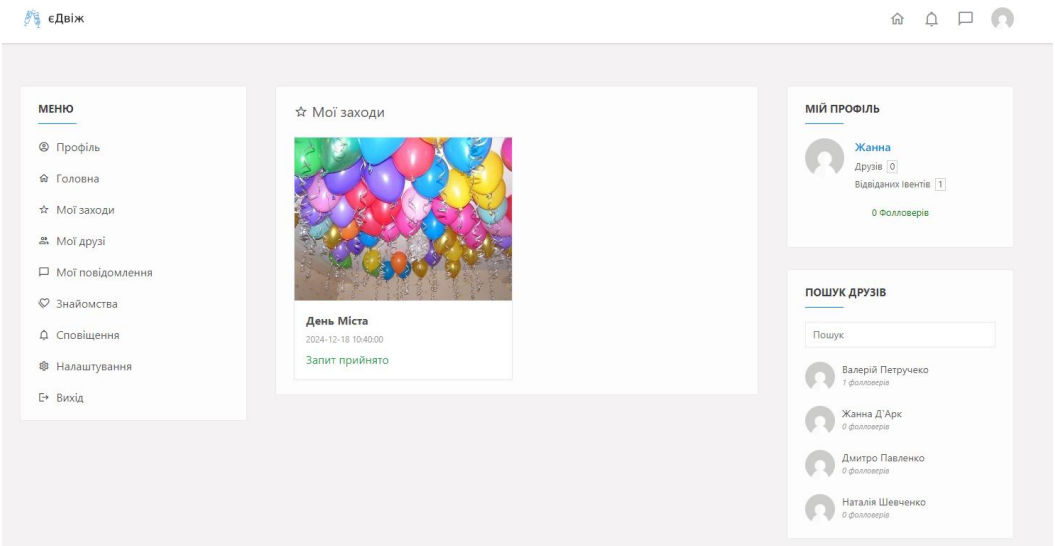

Рисунок 3.17 – Сторінка подій, на які отримано погодження

Для забезпечення комунікацій користувачів між собою відвідувачі можуть використовувати функціонал «Мої повідомлення» для спілкування з іншими користувачами, як зображено на рисунку 3.18.

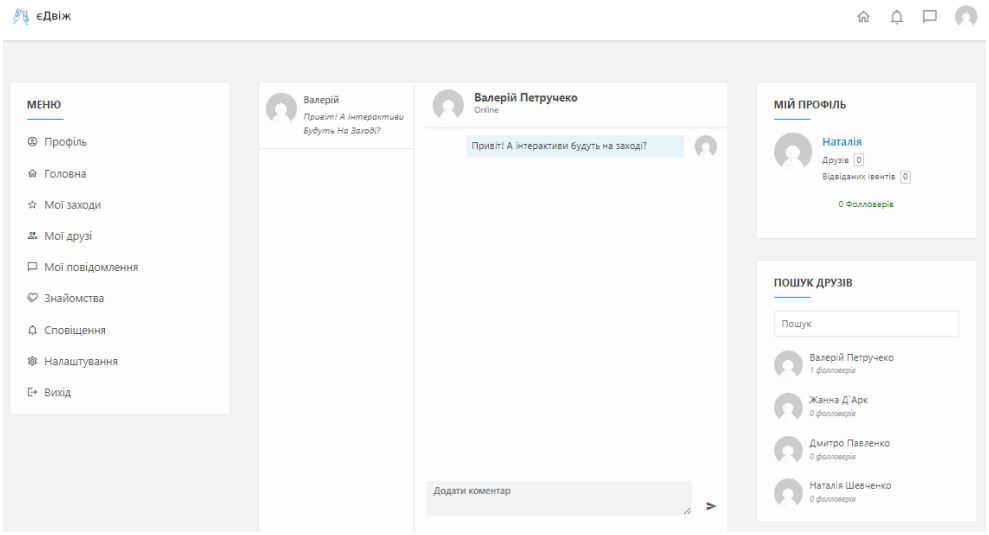

Рисунок 3.18 – Комунікація користувачів у «мої повідомлення»

Кожному користувачу доступний фунціонал підписок на інших користувачів шляхом додавання у список друзів. Перегляд списку та подальша взаємодія з друзями (написати повідомлення, видалення друзів) можлива у вкладці «Мої друзі» як зображено на рисунку 3.19.

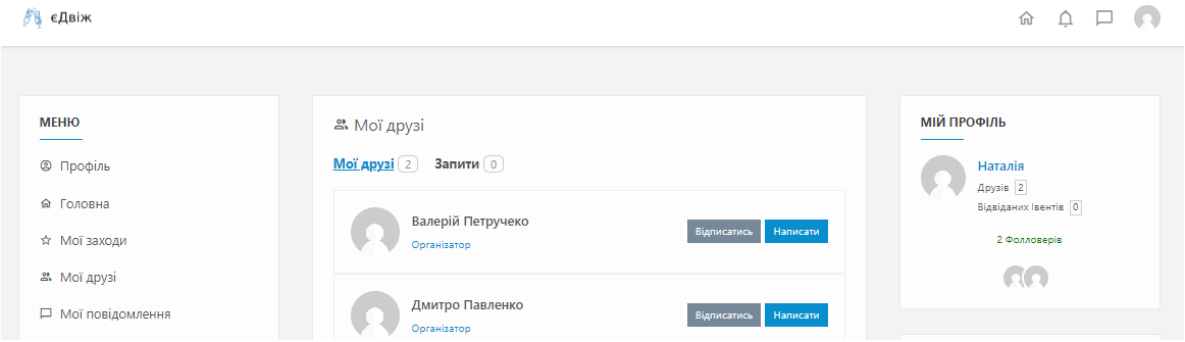

Рисунок 3.19 – Сторінка «Мої друзі»

Для забезпечення функціоналу пошуку нових друзів за інтересами, користувачу доступна вкладка «Знайомства» з можливістю відправити профілю «лай», або пропустити картку, як зображено на рисунку 3.20.

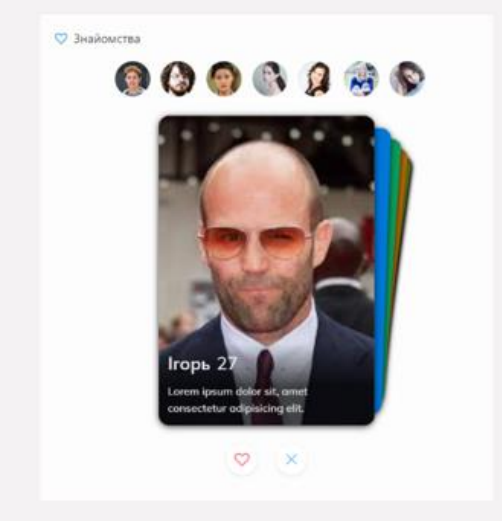

Рисунок 3.20 – Вкладка «Знайомства»

Пошук нових друзів забезпечено за інтересами, які можна обрати через фільтр, як зображено на рисунку 3.21.

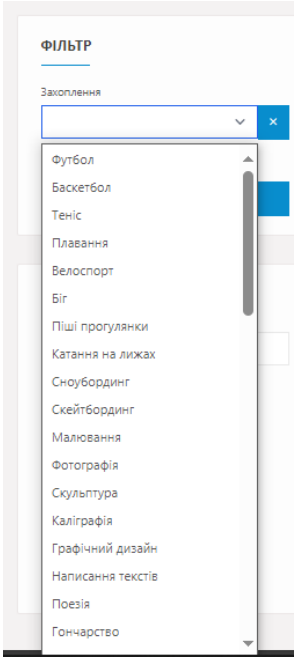

Рисунок 3.21 – Фільтр інтересів у вкладці «Знайомства»

3.4 Використання інформаційної технології планування, координації та управління подіями й заходами зі сторони організатора

Авторизовані організатори можуть створювати події та заходи через реалізований функціонал у вкладці «Мої заходи», як зображено на рисунку 3.22.

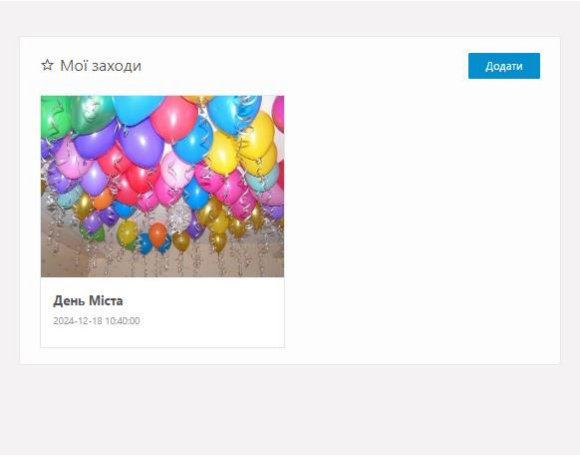

Рисунок 3.22 – Функція «Додати» подію

Під час створення події чи заходу організатор зазначає точні дані про проведення події (назву, місто, дата, час, формат [онлайн, офлайн, гібридний], категорія, галузь, аудиторія, опис) та може персоналізувати обкладинку події, як зображено на рисунку 3.23.

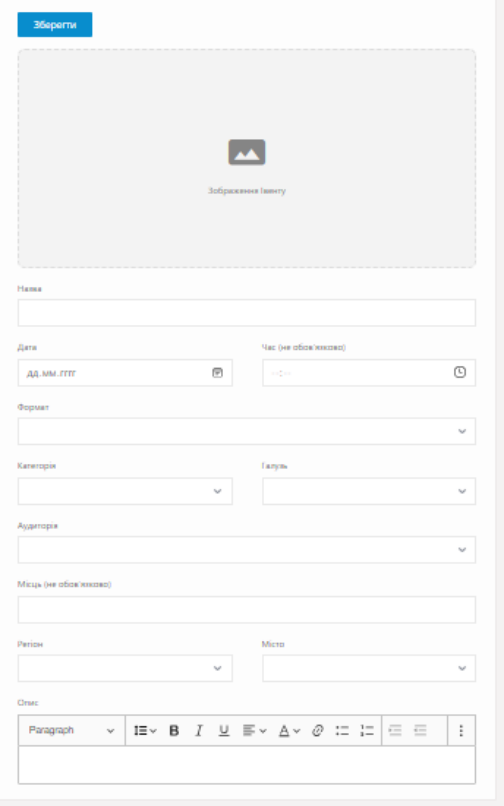

Рисунок 3.23 – Створення нової події

Після того, як подію було створено, організатор може отримувати запити від відвідувачів на участь у заході. Модерація заявок відбувається завдяки кнопок біля учасників. Додатково організатор може назначити організаторами інших осіб для полегшення керування, модерації та відстеженням списку учасників, як зображено на рисунку 3.24.

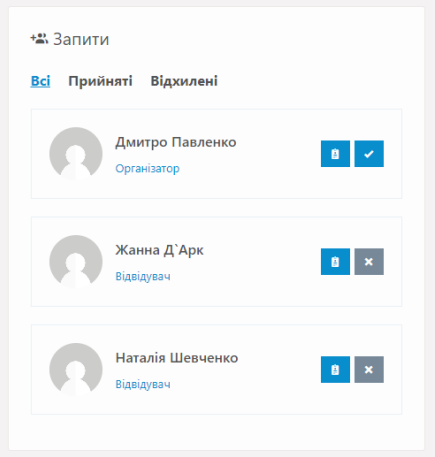

Рисунок 3.24 – Список заявок взяти участь у заході

3.5 Використання інформаційної технології планування, координації та управління подіями й заходами зі сторони адміністратора

Для забезпечення додаткової безпеки користувачів реалізовано функціонал адміністрування інформаційної технології.

Адміністратор має доступ до перегляду бази даних користувачів, з функцією блокування, як зображено на рисунку 3.25.

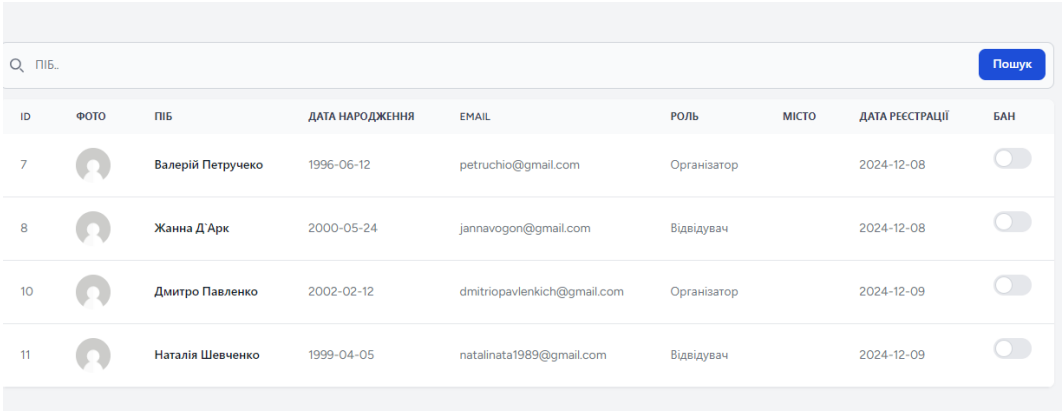

Рисунок 3.25 – База даних користувачів у панелі адміна

Для можливості подальшого забезпечення «чистоти» даних, адміністратор має доступ до видалення подій та заходів х метою очищення даних за період від 5 років, або у випадку втрати організатора доступу до його особистих даних, як зображено на рисунку 3.26.

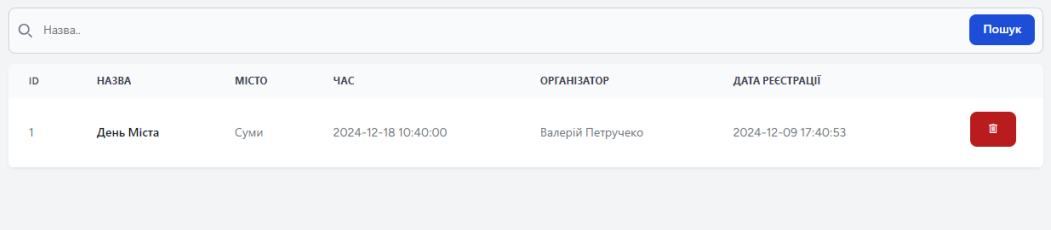

Рисунок 3.26 – База даних подій у панелі адміна

3.6 Тестування навантаження інформаційної технології

У сучасних інформаційних технологіях важливою є здатність обробляти велику кількість одночасних запитів[16]. Потокове навантаження дозволяє оцінити межі продуктивності системи, виявити "вузькі місця" та визначити оптимальні налаштування.

Тестування продуктивності додатку проводилось з використанням Wrk. Wrk – це високопродуктивний інструмент для тестування навантаження на веб-сервіси[17]. Він дозволяє генерувати велику кількість запитів до сервера за короткий проміжок часу та аналізувати продуктивність вашої програми.

Установка:

```
brew install wrk
```

Запуск:

```
wrk -t4 -c10 -d30s http://127.0.0.1:8000/api/users
```

-t4: Число потоків (4 потоки).

-C10: кількість одночасних з'єднань (10 клієнтів).

-d30s: Тривалість тестування (30 секунд).

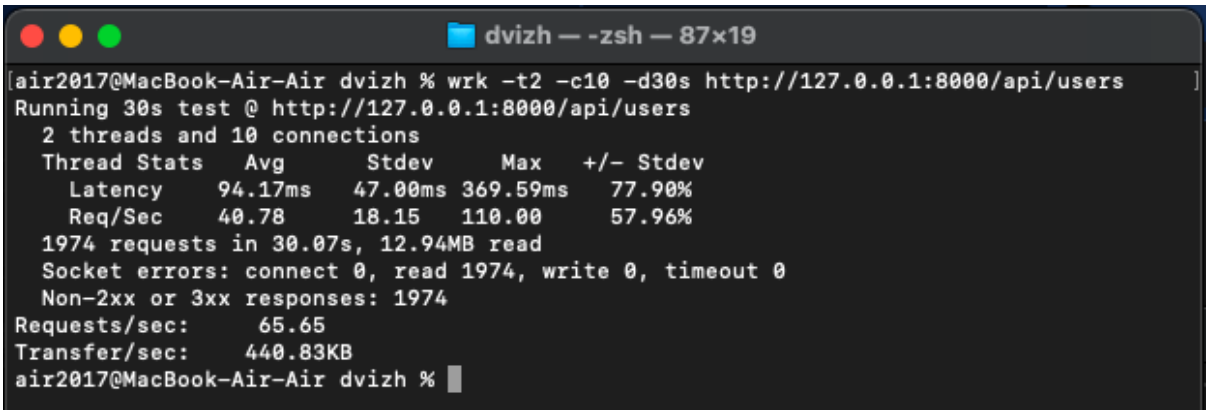

Рисунок 3.27 – результат тестування для 10 клієнтів

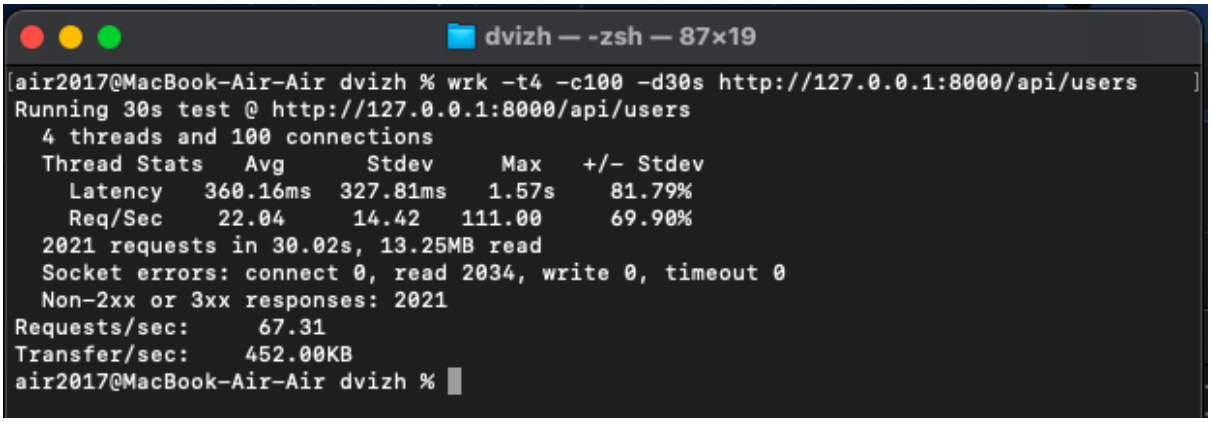

Рисунок 3.28 – результат тестування для 100 клієнтів

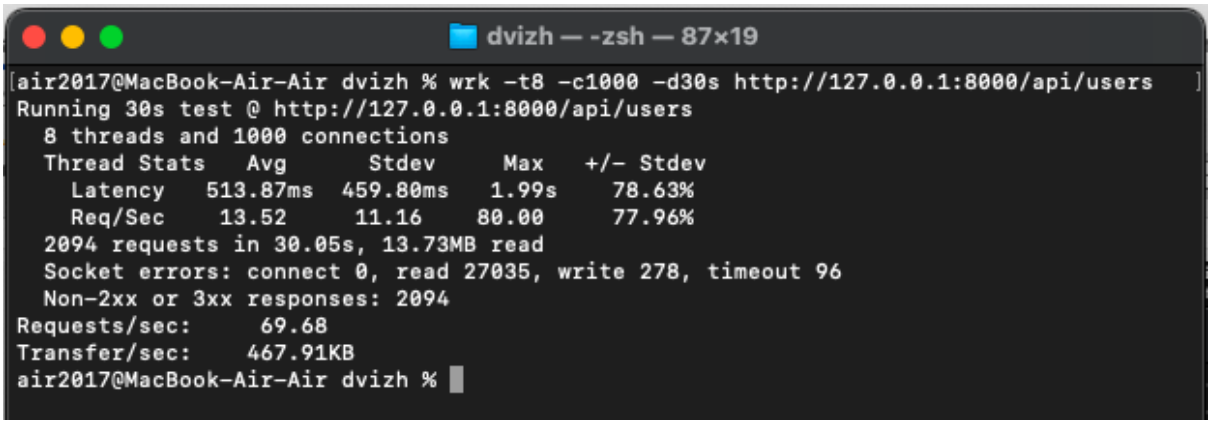

Рисунок 3.29 – результат тестування для 1000 клієнтів

Таблиця 1 – Загальні показники тестування

Тестове навантаження	Середня затримка (ms)	Запитів в секунду	Пропускна здатність (MB/sec)
10 клієнтів	94	65	440KB
100 клієнтів	360	67	452KB
1000 клієнтів	513	69	467KB

Інформаційна технологія показала себе стабільною під легким і середнім навантаженням, обробляючи запити без помилок.

ВИСНОВКИ

Під час виконання та розробки інформаційної технології планування, координації та управління подіями й заходами було проведено дослідження та аналіз актуальних веб-додатків та сайтів у напрямку організаційних мереж. Завдяки аналізу було виявлено їх переваги та недоліки, що дало змогу чітко визначити подальші функціональні вимоги для їх подальшої реалізації.

У ході виконання кваліфікаційної роботи було виконано такі завдання:

- Проведено аналіз веб-додатків аналогів;
- Розроблено інтуітивно-зрозумілий інтерфейс;
- Створено функцію реєстрації для учасників подій та організаторів;
- Створено повний функціонал створення подій та заходів з можливістю їх редагування, кастомізації;
- Розроблено системи подачі заявок для участі у подіях;
- Розроблено систему додавання друзів;
- Розроблено систему спілкування користувачів між собою.

Реалізовано функціонал, який дозволяє використовувати одну інформаційну технологію, як для пошуку подій та їх організацію з можливістю редагування всіх складових.

При виконанні кваліфікаційної роботи було поставлено чітку мету створити інформаційну технологію планування, координації та управління подіями й заходами. При виконанні аналізу було підібрано програмне забезпечення та інструменти завдяки яким відбувалась реалізація. Було продемонстровано роботу інформаційної технології у повному обсязі у ролі учасника, організатора та адміністратора. Результатом кваліфікаційної роботи магістра є розроблена інформаційна технологія планування, координації та управління подіями й заходами.

Практичне призначення даної інформаційної технології являє собою спрощення організації та пошуку подій. Це дає змогу користувачам створювати власні заходи і ділитися ними з іншими людьями розвиваючи культуру свого міста.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Pros and Cons of Social Media [Електронний ресурс] – https://www.brownhealth.org/be-well/social-media-good-bad-and-ugly

2. 13 Positive Effects of Social Media on Our Society Today [Електронний ресурс] – https://www.kubbco.com/blog/13-positive-effects-of-social-media-on-our-society-today/

3. What Are The Different Types Of Social Media? 10 Key Types [Електронний ресурс] – https://www.indeed.com/career-advice/career-development/types-of-social-media

4. Eventbrite [Електронний ресурс] – https://www.eventbrite.com

5. Meetup [Електронний ресурс] – https://www.meetup.com

6. Ticketmaster [Електронний ресурс] – https://www.ticketmaster.com

7. Advantages of HTML [Електронний ресурс] - https://www.scaler.com/topics/advantages-of-html/

8. Advantages of CSS [Електронний ресурс] - https://www.tutorialspoint.com/What-are-the-advantages-of-CSS

9. Why Vue.js [Електронний ресурс] - https://www.sam-solutions.com/blog/why-vue-js/

10. Benefits of Vue.js [Електронний ресурс] - https://www.tatvasoft.com/outsourcing/2021/10/what-is-vue-js-and-its-benefits.html

11. Advantages of PHP [Електронний ресурс] - https://www.geeksforgeeks.org/advantages-and-disadvantages-of-php/

12. PHP benefits [Електронний ресурс] - https://anywhere.epam.com/business/pros-and-cons-of-php

13. Laravel framework benefits [Електронний ресурс]- https://www.netsolutions.com/insights/laravel-framework-benefits/

14. Why Laravel [] - https://www.fastfwd.com/why-choose-laravel-for-your-next-web-project/

15. J. Shute et al., F1: A distributed SQL database that scales. Proceedings of the VLDB Endowment. 6, 1068–1079 (2013).

16. HTTPS stress test - https://www.investopedia.com/terms/s/stresstesting.asp

17. Wrk test - https://www.networknt.com/tool/wrk-perf/

ДОДАТОК А. КОД РЕАЛІЗАЗІЇ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

ProfileController.php

Контролер для роботи з профілем користувача.

```
<?php

namespace App\Http\Controllers;

use App\Http\Resources\EventResoure;
use App\Http\Resources\UserFull;
use App\Models\Events;
use App\Models\Friends;
use App\Models\Notifications;
use App\Models\Regions;
use App\Models\User;
use Illuminate\Support\Facades\Auth;
use Inertia\Inertia;
use Illuminate\Http\Request;
use Log;
use Storage;
use Illuminate\Support\Facades\Hash;

class ProfileController extends Controller
{
    function show()
    {
        if (Auth::user()->roles_id == 2) {
            $events = Events::where('user_id', Auth::id())->get();
        } else {
            $events = Events::whereHas('applications', function ($q) {
                $q->where('user_id', Auth::id())->where('accept', 1);
            })->get();
        }
        return Inertia::render('Profile', [
            'user' => new UserFull(Auth::user()),
            'events' => EventResoure::collection($events)
        ]);
    }

    function updateHeader(Request $request)
```

```
{
    $header = Storage::disk('public')->put('headers', $request->header);
    User::find($request->user_id)->update([
        'header' => $header
    ]);
}

function friends()
{
    $data = Friends::where('user_1_id', Auth::id())->pluck('user_2_id')->toArray();

    $users = User::whereHas('followers', function ($q) use ($data) {
        $q->whereIn('user_2_id', $data);
    })->get();

    $requests = User::whereHas('myFollowers', function ($q) use ($data) {
        $q->where('user_2_id', Auth::id())->whereNotIn('user_1_id', $data);
    })->get();

    return Inertia::render('MyFriends', [
        'users' => UserFull::collection($users),
        'requests' => UserFull::collection($requests)
    ]);
}

function subscribe($id)
{
    $model = new Friends();
    if (!Friends::where('user_1_id', Auth::id())->where('user_2_id', $id)->exists()) {
        $model->create([
            "user_1_id" => Auth::id(),
            "user_2_id" => $id
        ]);
    }
    return response('ok', 200);
}

function unSubscribe($id)
{
    Friends::where('user_1_id', Auth::id())->where('user_2_id', $id)->delete();
    return response('ok', 200);
```

```
    }

    function edit()
    {
        $regions = Regions::with('cities')->get();
        return Inertia::render('Settings', [
            'user' => new UserFull(Auth::user()),
            'regions' => $regions
        ]);
    }

    function update(Request $request)
    {
        $id = Auth::id();
        $data = $request->except('photo', 'header');

        if (isset($data['password'])) {
            $data["password"] = Hash::make($request->password);
        }

        if (isset($request['newPhoto']) && $request['newPhoto'] != "null" && $request['newPhoto'] != "undefined") {
            $photo = Storage::disk('public')->put('avatars', $request->newPhoto);
            $data['photo'] = $photo;
        }

        User::find($id)->update($data);
    }

    function notifications() {
        $notifications = Notifications::where('user_id', Auth::id())->orderBy('created_at', 'desc')->get();
        return Inertia::render('Notifications', [
            'notifications' => $notifications
        ]);
    }

    function notificationsReaded($id) {
        Notifications::find($id)->update([
            'readed' => 1
        ]);
    }
```

}

EventsController.php

Контролер для роботи з заходами.

```
<?php

namespace App\Http\Controllers;

use App\Http\Resources\EventResoure;
use App\Http\Resources\EventsCommentsResoure;
use App\Http\Resources\EventsResoure;
use App\Http\Resources\UserApplications;
use App\Http\Resources\UserFull;
use App\Models\Applications;
use App\Models\Categories;
use App\Models\Cities;
use App\Models\EventArea;
use App\Models\EventAudience;
use App\Models\EventComments;
use App\Models\EventLikes;
use App\Models\Events;
use App\Models\EventType;
use App\Models\EventViews;
use App\Models\Regions;
use App\Models\User;
use Illuminate\Http\Request;
use Inertia\Inertia;
use Illuminate\Support\Facades\Auth;
use Log;
use Storage;
use URL;

class EventsController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        if (Auth::user()->roles_id == 2) {
            $events = Events::where('user_id', Auth::id())->get();
        } else {
```

```
        $events = Events::whereHas('applications', function ($q) {
            $q->where('user_id', Auth::id())->where('accept', 1);
        })->get();
    }
    return Inertia::render('MyEvents', [
        'events' => EventsResoure::collection($events)
    ]);
}

/**
 * Show the form for creating a new resource.
 */
public function create()
{
    $categories = Categories::get();
    $eventType = EventType::get();
    $eventAudience = EventAudience::get();
    $eventArea = EventArea::get();
    $regions = Regions::with('cities')->get();
    $event = Events::query();
    return Inertia::render('NewEvent', [
        'categories' => $categories,
        'eventType' => $eventType,
        'eventAudience' => $eventAudience,
        'eventArea' => $eventArea,
        'regions' => $regions,
        'event' => $event,
    ]);
}

/**
 * Store a newly created resource in storage.
 */
public function store(Request $request)
{
    $data = $request->all();
    $preview = Storage::disk('public')->put('', $request->newPhoto);
    $data['preview'] = $preview;
    $data['user_id'] = Auth::id();
    $event = Events::create($data);
    return response()->json($event);
```

```
}

/**
 * Display the specified resource.
 */
public function show($id)
{
    $event = Events::find($id);

    EventViews::create([
        'user_id' => Auth::id(),
        'event_id' => $id
    ]);

    return Inertia::render('Event', [
        'event' => new EventResoure($event)
    ]);
}

/**
 * Show the form for editing the specified resource.
 */
public function edit($id)
{
    $categories = Categories::get();
    $eventType = EventType::get();
    $eventAudience = EventAudience::get();
    $eventArea = EventArea::get();
    $event = Events::find($id);
    $regions = Regions::with('cities')->get();

    $event["preview"] = URL::asset('storage/' . $event->preview);
    $event["regions_id"] = $event->city->regions_id;

    return Inertia::render('NewEvent', [
        'categories' => $categories,
        'eventType' => $eventType,
        'eventAudience' => $eventAudience,
        'eventArea' => $eventArea,
        'event' => $event,
        'regions' => $regions,
```

```
        ]);
    }

    /**
     * Update the specified resource in storage.
     */
    public function update(Request $request, $id)
    {
        $data = $request->except('preview');

        if ($request->hasFile('newPhoto')) {
            $photo = Storage::disk('public')->put('events', $request->newPhoto);
            $data['preview'] = $photo;
        }

        Events::find($id)->update($data);
    }

    /**
     * Remove the specified resource from storage.
     */
    public function destroy($id)
    {
        Events::find($id)->delete();
    }

    function likes($id)
    {
        if (EventLikes::where('user_id', Auth::id())->where('event_id', $id)->exists()) {
            EventLikes::where('user_id', Auth::id())->where('event_id', $id)->delete();
        } else {
            EventLikes::create([
                'user_id' => Auth::id(),
                'event_id' => $id
            ]);
        }
        return response()->json(EventLikes::where('event_id', $id)->count());
    }

    function comment(Request $request, $id)
    {
```

```
    $response = EventComments::create([
        'user_id' => Auth::id(),
        'event_id' => $id,
        'text' => $request->comment
    ]);
    return response()->json(new EventsCommentsResoure($response));
}

function apply($id)
{
    Applications::create([
        'user_id' => Auth::id(),
        'event_id' => $id
    ]);
    return response()->json(Applications::where('user_id', Auth::id())->first());
}

function updateApply(Request $request, $id)
{
    Applications::find($id)->update([
        'acceps' => $request->status
    ]);
}

function home(Request $request)
{
    $categories = Categories::get();
    $eventType = EventType::get();
    $eventAudience = EventAudience::get();
    $eventArea = EventArea::get();
    $events = Events::get();
    $cities = Cities::get();

    return Inertia::render('Index', [
        'categories' => $categories,
        'eventType' => $eventType,
        'eventAudience' => $eventAudience,
        'eventArea' => $eventArea,
        'cities' => $cities,
        'events' => EventsResoure::collection($events)
    ]);
```

```
    }

    function search(Request $request)
    {
        $events = Events::query();

        if (isset($request->title)) {
            $events->where('title', 'like', '%' . $request->title . '%');
        }
        if (isset($request->date)) {
            $events->whereDate('date', $request->date);
        }
        if (isset($request->time)) {
            $events->whereTime('time', $request->time);
        }
        if (isset($request->category)) {
            $events->where('category_id', $request->category);
        }
        if (isset($request->event_type)) {
            $events->where('event_type_id', $request->event_type);
        }
        if (isset($request->event_area)) {
            $events->where('event_area_id', $request->event_area);
        }
        if (isset($request->city)) {
            $events->where('cities_id', $request->city);
        }

        $events = $events->get();

        return response()->json(EventsResoure::collection($events));
    }

    function allEvents() {
        $events = Events::get();
        return Inertia::render('Events', [
            'events' => EventsResoure::collection($events)
        ]);
    }
}
```

DatingController.php

Контролер для роботи модуля знайомств знайомств.

```
<?php

namespace App\Http\Controllers;

use App\Http\Resources\UserFollowers;
use App\Http\Resources\UserFull;
use App\Models\Dating;
use App\Models\Hobby;
use App\Models\User;
use Illuminate\Http\Request;
use Inertia\Inertia;
use Illuminate\Support\Facades\Auth;

class DatingController extends Controller
{
    function index()
    {
        $hobbies = Hobby::get();
        return Inertia::render('Dating', [
            'hobbies' => $hobbies
        ]);
    }

    function hobby()
    {
        $hobbies = Hobby::get();
        return response()->json($hobbies);
    }

    function activeUsers()
    {
        $notActiveUsers = Dating::where('user_1_id', Auth::id())
            ->whereNotNull('sympathy')
            ->select('user_2_id')
            ->get()
            ->toArray();
        $users = User::whereNotIn('id', $notActiveUsers)->get();
        return response()->json(UserFull::collection($users));
    }
```

```
    function store(Request $request)
    {
        Dating::create([
            "user_1_id" => Auth::id(),
            "user_2_id" => $request->user_id,
            "sympathy" => $request->sympathy
        ]);
        $user = new UserFull(User::find($request->user_id));
        return response()->json($user);
    }

    function mySympathy()
    {
        $data = Dating::where('user_1_id', Auth::id())->where('sympathy', 1)->orderBy('created_at', 'desc')->get();
        return response()->json(UserFollowers::collection($data));
    }
}
```

ApplicationsController.php

Контролер для обробки заявок.

```
<?php

namespace App\Http\Controllers;

use App\Http\Resources\UserApplications;
use App\Http\Resources\UserFull;
use App\Models\Applications;
use App\Models\Notifications;
use App\Models\User;
use Illuminate\Http\Request;
use Inertia\Inertia;

class ApplicationsController extends Controller
{
    function index($id)
    {
        $applications = Applications::where('event_id', $id)->get();
        return Inertia::render('Applications', [
            'users' => UserApplications::collection($applications)
        ]);
    }
    function update(Request $request, $id)
```

```
    {
        $app = Applications::with('event')->find($id);
        $app->update([
            'accept' => $request->accept
        ]);

        if ($request->accept == 1) {
            Notifications::create([
                'description' => 'Ваш запит до участі в заході "' . $app->event['title'] . '" схвалено!',
                'user_id' => $app->user_id
            ]);
        }
        if ($request->accept == 0) {
            Notifications::create([
                'description' => 'Ваш запит до участі в заході "' . $app->event['title'] . '" відхилено((',
                'user_id' => $app->user_id
            ]);
        }
    }
}
```

Profile.vue

Компонент профілю.

```
<script setup>
import { Head } from '@inertiajs/vue3';
import { ref } from 'vue';
import Menu from '@/Components/Menu.vue';
import Header from '@/Components/Header.vue';
import Footer from '@/Components/Footer.vue';
import Friends from '@/Components/Friends.vue';
import FriendCard from '@/Components/FriendCard.vue';
import Event from '@/Components/Event.vue';
import Modal from '@/Components/Modal.vue';

const writeMessagePopupRef = ref(null);
const message = ref("");

const props = defineProps({
    user: {
        type: Object,
    },
    events: {
```

```
        type: Object,
      }
    });

    const writeMessage = () => {
      writeMessagePopupRef.value.open();
    }

    const sendMessage = () => {
      axios.post('/messages', {
        user_id: props.user.data.id,
        message: message.value
      })
        .then(() => {
          message.value = ""
          writeMessagePopupRef.value.close();
        })
    }

    </script>
    <template>

      <Head title="Головна" />

      <Header />

      <div class="prfile-header" :style="`background: url('${user.data.header}') no-repeat; background-size: cover`">
        <div class="container flex justify-end items-end h-full py-20">
          <div class="add-btn" v-if="user.data.id != $page.props.auth.user.data.id">
            {{ user.data.followers.length }} Підписників
            <div class="flex gap-1">
              <button class="btn" @click="writeMessage()">Написати</button>
              <button class="btn" @click="unSubscribe(user.data.id)"
                v-if="user.data.followers.find(item => item.user_1_id == $page.props.auth.user.data.id)">Відписатись</button>
              <button class="btn" @click="subscribe(user.data.id)" v-else>Підписатись</button>
            </div>
          </div>
          <div v-else>
            <form class="edit-photo">
```

```
            <span class="mdi mdi-camera"></span>
            <label class="fileContainer">
              Змінити фото
              <input type="file" @change="previewFiles">
            </label>
          </form>
        </div>
      </div>
      <div class="navigation">
        <div class="container flex items-center gap-10 py-3">
          <img :src="user.data.photo" class="avatar">
          <div class="name">
            {{ user.data.name }} {{ user.data.surname }}
            <span>{{ user.data.role }}</span>
          </div>
          <ul class="menu">
            <li @click="tab = 1" :class="tab == 1 ? 'active' : ''">Івенти</li>
            <li @click="tab = 2" :class="tab == 2 ? 'active' : ''">Підписники</li>
            <li @click="tab = 3" :class="tab == 3 ? 'active' : ''">Про мене</li>
          </ul>
        </div>
      </div>
    </div>

    <div class="container">
      <div class="grid grid-cols-12 gap-10">
        <div class="col-span-3">
          <Menu />
        </div>
        <div class="col-span-6">
          <div class="block">
            <div v-if="tab == 1">
              <Event v-for="item in events.data" :data="item" />
            </div>
            <div v-if="tab == 2">
              <div class="friend-card" v-for="item in user.data.followers" :data="item">
                <div class="username">
                  <img :src="item.user_1.photo" class="rounded-full avatar" alt="">
                  <div class="name">
                    {{ item.user_1.name }} {{ item.user_1.surname }}
                    <span>{{ item.user_1.role }}</span>
```

```
                        </div>
                    </div>
                    <div class="actives">
                        <button class="btn more-action" @click="unSubscribe(item.user_1_id)"
                            v-if="item.user_1.followers.find(i => i.user_1_id ==
$page.props.auth.user.data.id)">Відписатись</button>
                        <button class="btn" @click="subscribe(item.user_1_id)" v-else>Підписатись</button>
                        <button class="btn">Написати</button>
                    </div>
                </div>
            </div>
            <div v-if="tab == 3">
                {{ user.data.description }}
            </div>
        </div>
    </div>
    <div class="col-span-3">
        <Friends />
    </div>
</div>
</div>

<Modal ref="writeMessagePopupRef">
    <template #title>
        Нове повідомлення
    </template>
    <template #body>
        <div class="modal-body">
            <textarea placeholder="Ваше повідомлення.." v-model="message"></textarea>
            <button class="btn mt-2" @click="sendMessage()">Написати</button>
        </div>
    </template>
</Modal>

<Footer />
</template>
<script>
export default {
    data() {
        return {
            header: null,
```

```
        tab: 1,
        message: ""
    }
},
methods: {
    previewFiles(event) {
        var input = event.target;
        if (input.files && input.files[0]) {
            var reader = new FileReader();
            reader.onload = (e) => {
                this.header = input.files[0];
                this.saveHeader();
            }
            reader.readAsDataURL(input.files[0]);
        }
    },
    saveHeader() {
        var data = new FormData;
        data.append('header', this.header);
        data.append('user_id', this.$page.props.auth.user.data.id);
        axios.post('/header', data)
            .then(() => {
                window.location.reload();
            })
    },
    subscribe(user_id) {
        axios.post('/subscribe/' + user_id)
            .then(() => {
                window.location.reload();
            })
    },
    unSubscribe(user_id) {
        axios.post('/unsubscribe/' + user_id)
            .then(() => {
                window.location.reload();
            })
    },
    removeFriend(data) {
        let index = this.user.data.myFollowers.indexOf(data)
        this.user.data.myFollowers.splice(index, 1)
    }
```

```
    }
}
</script>
<style lang="scss" scoped>
.edit-photo {
    background: rgba(0, 0, 0, 0.7) none repeat scroll 0 0;
    color: #fff;
    padding: 5px 20px;
    border: 1px solid transparent;
    display: flex;
    align-items: center;
    overflow: hidden;
}

.fileContainer {
    color: #d8d8d8;
    font-size: 11px;
    margin: 0 0 0 5px;
    position: relative;
    text-transform: capitalize;
}

.fileContainer [type=file] {
    cursor: pointer;
    display: none;
    position: absolute;
    text-align: right;
}

.prfile-header {
    height: 444px;
    background-size: cover;
    background-position: center;
    margin-top: -60px;
    margin-bottom: 120px;

    .add-btn {
        color: #fff;
        text-shadow: 0 2px 0 #4a4a4a;
        display: flex;
        gap: 20px;
```

```
}

.navigation {
  background: #fff;

  .avatar {
    border: 8px solid rgba(255, 255, 255, 0.8);
    border-radius: 3px 25px 0;
    box-shadow: 0 1px 0 #e1e8ed;
    float: right;
    margin-top: -200px;
    overflow: hidden;
    position: relative;
    width: 223px;
    height: 218px;
    object-fit: cover;
  }

  .name {
    display: flex;
    flex-direction: column;
    align-items: end;
    color: #088dcd;
    font-size: 24px;
    font-weight: 500;

    span {
      color: #3a3a3a;
      font-size: 10px;
      font-weight: 400;
    }
  }

  .menu {
    display: flex;
    align-items: center;

    li {
      margin-right: 50px;
      color: #088dcd;
      font-size: 14px;
```

```
            position: relative;
            line-height: initial;
            text-transform: capitalize;
            font-weight: 400;
            cursor: pointer;
        }

        .active {
            text-decoration: underline;
        }
    }
  }
}

textarea {
  background: #f3f3f3 none repeat scroll 0 0;
  border-color: transparent;
  border-radius: 3px;
  color: #000000;
  font-size: 13.5px;
  font-weight: 500;
  height: 40px;
  line-height: 16px;
  width: 100%;
  height: 50px;
  min-height: 100px
}

.btn {
  border-radius: 2px;
  font-size: 14px;
  padding: 4px 20px;
  border: 1px solid transparent;
  background: #088dcd;
  color: #fff;
}

.frend-card {
  margin-bottom: 20px;

  &:last-child {
```

```
        margin-bottom: 0;
    }
}
</style>
```

Event.vue

Компонент заходу.

```
<script setup>
import Comment from './Comment.vue';
import { Link } from '@inertiajs/vue3';
</script>
<template>
    <div>
        <div class="event">
            <div class="event__header">
                <div class="user">
                    <img :src="data.user.photo" class="rounded-full avatar" alt="">
                    <div>
                        <Link class="event__user" :href="`/user/${data.user.id}`">{{ data.user.name }} {{
                            data.user.surname }}</Link>
                        <div>
                            <span class="mdi mdi-clock-outline"></span> Опубліковано: {{ data.created_at }}
                        </div>
                    </div>
                </div>
                <div class="date"
                    v-if="(data.apply && data.apply.accept === 1) || data.user.id == $page.props.auth.user.data.id">
                    {{ data.date }} {{ data.time }}
                    <span>{{ data.city }}</span>
                </div>
                <div v-else role="status" class="mt-4">
                    <div class="h-2.5 bg-gray-200 rounded-full dark:bg-gray-700 w-48 mb-2"></div>
                    <div class="h-2 bg-gray-200 rounded-full dark:bg-gray-700 max-w-[60px] mb-2.5"></div>
                </div>
            </div>
            <Link :href="`/events/${data.id}`">
            <h1>{{ data.title }}</h1>
            </Link>
            <div class="event__preview mt-4">
                <Link :href="`/events/${data.id}`">
                <img :src="data.preview" alt="">
```

```
            </Link>
        </div>
        <div class="event__actives">
            <ul class="flex gap-5">
                <li>
                    <span class="mdi mdi-eye-outline"></span>
                    <span class="count">{{ data.views }}</span>
                </li>
                <li>
                    <span class="mdi mdi-forum-outline"></span>
                    <span class="count">{{ data.comments.length }}</span>
                </li>
                <li @click="addLike()">
                    <span class="mdi mdi-heart-outline"></span>
                    <span class="count">{{ data.likes }}</span>
                </li>
            </ul>

            <ul class="flex gap-5" v-if="data.user.id == $page.props.auth.user.data.id">
                <li>
                    <Link :href="`/events/${data.id}/applications`">
                    <span class="mdi mdi-account-multiple-plus" style="font-size: 20px;"></span>
                    <span class="count">{{ data.applications }}</span>
                    </Link>
                </li>
                <li>
                    <Link :href="`/events/${data.id}/edit`">
                    <span class="mdi mdi-pencil"></span>
                    </Link>
                </li>
            </ul>

            <button class="btn" @click="apply()"
                v-if="!data.is_apply && data.user.id != $page.props.auth.user.data.id">Подати запит</button>
            <span v-if="data.apply && data.apply.accept === null">Запит розглядається</span>
            <span class="text-red-600" v-if="data.apply && data.apply.accept === 0">Запит відхилено</span>
            <span class="text-green-600" v-if="data.apply && data.apply.accept === 1">Запит
прийнято</span>
        </div>
        <div v-if="(data.apply && data.apply.accept === 1) || data.user.id == $page.props.auth.user.data.id">
            <p class="event__description" v-html="data.description"></p>
```

```
                    <div class="event__comments">
                        <Comment v-for="item in data.comments" :item="item"></Comment>
                    </div>
                    <div class="event__form">
                        <img :src="$page.props.auth.user.data.photo" class="rounded-full avatar" alt="">
                        <textarea placeholder="Додати коментар" v-model="comment" style="margin-bottom:
0"></textarea>
                        <button @click="addComment()"><span class="mdi mdi-send"></span></button>
                    </div>
                </div>
                <div v-else role="status" class="mt-4">
                    <div class="h-2.5 bg-gray-200 rounded-full dark:bg-gray-700 w-48 mb-4"></div>
                    <div class="h-2 bg-gray-200 rounded-full dark:bg-gray-700 max-w-[360px] mb-2.5"></div>
                    <div class="h-2 bg-gray-200 rounded-full dark:bg-gray-700 mb-2.5"></div>
                </div>
            </div>
        </template>
        <script>
        export default {
            props: {
                data: Object
            },
            data() {
                return {
                    comment: ""
                }
            },
            methods: {
                addComment() {
                    axios.post(`/events/${this.data.id}/comment`, {
                        comment: this.comment
                    })
                        .then((response) => {
                            this.data.comments.push(response.data)
                            this.comment = ""
                        })
                },
                addLike() {
                    axios.post(`/events/${this.data.id}/likes`)
                        .then((response) => {
```

```
                this.data.likes = response.data
            })
        },
        apply() {
            axios.post(`/events/${this.data.id}/apply`)
                .then((response) => {
                this.data.is_apply = true
                this.data.apply = response.data
            })
        }
    }
}
</script>
```

Dating.vue

Компонент сторінки знайомств.

```
<template>

    <Header />

    <div class="container">
        <div class="grid grid-cols-12 gap-10">
            <div class="col-span-3">
                <Menu />
            </div>
            <div class="col-span-6">
                <div class="block">
                    <div class="page-title">
                        <span class="mdi mdi-heart-multiple-outline"></span> Знайомства
                    </div>
                    <div>
                        <LikesUsers :data="sympathy" />
                    </div>
                    <swiper class="swiper" :allowTouchMove="false" :effect="'cards'" :grabCursor="true"
                        @swiper="onSwiper" :modules="modules" :slides-per-view="1" ref="swiperRef">
                        <swiper-slide v-for="item in users">
                            <div class="pic-and-actions">
                                <div class="pic" :style="`background: url(${item.photo}) center center/cover`">
                                    <div class="pic-text">
                                        <div class="pic-name-and-age">
                                            <h2>{{ item.name }}</h2>
                                            <h2>27</h2>
```

```
                    </div>
                    <div class="pic-bio">
                        {{ item.description }}
                    </div>
                </div>
            </div>
        </swiper-slide>
    </swiper>

    <div class="actions flex justify-center">
        <div class="action" @click="active(1)">
            <span class="mdi mdi-heart"></span>
        </div>

        <div class="action" @click="active(0)">
            <span class="mdi mdi-close-thick"></span>
        </div>
    </div>
  </div>
</div>
<div class="col-span-3">
  <div class="block">
    <div class="title">Фільтр</div>
    <label for="">Захоплення</label>
    <div class="flex items-center mb-1" v-for="(itemFilter, index) in formFilter.hobby">
        <select style="margin-bottom: 0" v-model="formFilter.hobby[index]">
            <option v-for="item in hobby" :value="item">{{ item.title }}</option>
        </select>
        <button class="btn">
            <span class="mdi mdi-close"></span>
        </button>
    </div>

    <button class="w-full" @click="addHobby()">
        <span class="mdi mdi-plus"></span>
    </button>

    <button class="btn w-full mt-4">Пошук</button>
  </div>
  <Friends />
```

```
            </div>
        </div>
    </div>
    <Footer />
</template>
<script>
import { Swiper, SwiperSlide } from 'swiper/vue';
import 'swiper/css';

import Menu from '@/Components/Menu.vue';
import Header from '@/Components/Header.vue';
import Footer from '@/Components/Footer.vue';
import Profile from '@/Components/Profile.vue';
import Friends from '@/Components/Friends.vue';
import LikesUsers from '@/Components/LikesUsers.vue';

import { ref } from 'vue';

import 'swiper/css/effect-cards';

import { EffectCards, Navigation } from 'swiper/modules';

export default {
    components: {
        Swiper,
        SwiperSlide,
        Menu,
        Header,
        Footer,
        Profile,
        Friends,
        LikesUsers
    },
    setup() {
        const swiperInstance = ref()

        function onSwiper(swiper) {
            swiperInstance.value = swiper
        }
        const swiperNextSlide = () => {
            swiperInstance.value.slideNext()
```

```
    };
    const swiperPrevSlide = () => {
        swiperInstance.value.slidePrev()
    };

    return {
        modules: [EffectCards, Navigation],
        swiperPrevSlide,
        swiperNextSlide,
        onSwiper,
    };
},
data() {
    return {
        users: [],
        sympathy: [],
        hobby: [],
        activeIndex: 0,
        formFilter: {
            hobby: [{
                title: "",
                id: null
            }]
        }
    }
},
created() {
    this.getData();
    this.mySympathy();
    this.getHobby();
},
methods: {
    Navigation() {
        return Navigation
    },
    addHobby() {
        this.formFilter.hobby.push({
            title: "",
            id: null
        })
    },
```

```
        getData() {
            axios.get('/datings/active-users')
                .then((response) => {
                    this.users = response.data;
                })
        },
        getHobby() {
            axios.get('/hobby')
                .then((response) => {
                    this.hobby = response.data;
                })
        },
        mySympathy() {
            axios.get('/datings/my-sympathy')
                .then((response) => {
                    this.sympathy = response.data;
                })
        },
        active(value) {
            axios.post('/datings/store', {
                user_id: this.users[this.activeIndex].id,
                sympathy: value
            })
                .then(() => {
                    this.mySympathy()
                    this.swiperNextSlide()
                    this.activeIndex++
                })
        }
    }
};
</script>
<style lang="scss" scoped>
.swiper {
    width: 350px;
    height: 500px;
    margin-top: 30px;
}

.swiper-slide {
    display: flex;
```

```
    align-items: center;
    justify-content: center;
    border-radius: 18px;
    font-size: 22px;
    font-weight: bold;
    box-shadow: 0 2px 10px 0 rgb(0 0 0 / 77%);
}

.pic-and-actions {
    width: 100%;
    height: 100%;
    display: flex;
    flex-direction: column;
    align-items: center;
}

.pic {
    width: 100%;
    height: 100%;
    border-radius: 10px;
    display: flex;
    align-items: flex-end;
    color: #eee;
    box-shadow: 0 2px 10px 0 rgba(136, 136, 136, 0.77);
}

.pic-text {
    padding: 15px;
    width: 100%;
    background: rgb(2, 0, 36);
    background: rgb(2, 0, 36);
    background: linear-gradient(180deg,
            rgba(2, 0, 36, 1) 0%,
            rgba(35, 34, 65, 0) 0%,
            rgba(0, 0, 0, 0.7) 52%);
    border-radius: 10px;
}

.pic-name-and-age {
    display: flex;
    align-items: center;
```

```
    margin-bottom: 6px;
}

.pic-name-and-age h2 {
    font-size: 1.8rem;
}

.pic-name-and-age h2:nth-child(2) {
    margin-left: 10px;
    font-weight: 500;
}

.pic-bio {
    line-height: 1.7rem;
    font-weight: 500;
    font-size: 1.1rem;
}

/* Actions */
.actions {
    flex: auto;
    display: flex;
    align-items: center;
    margin: 1rem 0;
    display: flex;
    gap: 30px;
    margin-top: 40px;
}

.action {
    display: flex;
    align-items: center;
    justify-content: center;
    background: #fff;
    height: 60px;
    width: 60px;
    border-radius: 50%;
    font-size: 2.5rem;
    box-shadow: 0 2px 6px 0 rgba(112, 125, 134, 0.14);
}
```

```
.actions .action:nth-child(1) {
    color: #fd5068;
}

.actions .action:nth-child(2) {
    color: #2db1ff;
}

.btn {
    color: #fff;
    font-size: 13px;
    padding: 9px 10px;
    text-transform: capitalize;
    background: #088dcd;
}
</style>
```