МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет

імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри _____ Є. О. Давиденко

підпис

«___» _____20____р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Вебзастосунок реєстрації подій

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 409.22010901

Здобувач _____І. І. Адамчук

підпис

«__» _____20__р.

Керівник ст. викладачка _____С. Ю. Боровльова

підпис

«__» _____20__р.

Консультант канд. техн. наук, доцент _____А. О. Алексєєва

підпис

«__» _____20__р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Зав. кафедри _____ Є. О. Давиденко

«_____» _____ 20__ р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи бакалавра

Видано студенту групи 409 факультету комп'ютерних наук.

Адамчуку Івану Івановичу

(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи

Вебзастосунок реєстрації подій

Затверджена наказом по ЧНУ від « 22 » грудня 2023 р. № 269

2. Строк представлення кваліфікаційної роботи «____» _____

20____ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні Очікуваним результатом є вебзастосунок для реєстрації на заходи, що мінімізує час та зусилля, необхідні для процесу реєстрації, та сприяє залученню громадян до активної участі в громадському житті та волонтерських заходах. Застосунок має включати функціонал «одноразової реєстрації», простий та інтуїтивно зрозумілий інтерфейс.

4. Перелік питань, що підлягають розробці:

провести аналіз наявних систем реєстрації на заходи та визначити їхні переваги та недоліки; обрати технологічний стек для розробки вебзастосунку; розробити мінімально життєздатний продукт концепції «одноразової реєстрації» для спрощення процесу участі в заходах; створити простий і легко запам'ятовуваний дизайн інтерфейсу; реалізувати необхідний функціонал для зберігання та обробки даних користувачів; протестувати вебзастосунок на відповідність бізнес-вимогам користувачів та безпеки даних.

5. Перелік графічних матеріалів

Презентація.

6. Завдання до спеціальної частини

Дослідження питань охорони праці в умовах, які безпосередньо пов'язані з діяльністю волонтерів.

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Алексєєва А. О.	Кафедра екології	Спеціальна частина з охорони праці

Керівник роботи ст. викладач Боровльова С. Ю.

(посада, прізвище, ім'я, по батькові)

(підпис)

Завдання прийнято до виконання

Адамчук І. І.

(прізвище, ім'я, по батькові студента)

(підпис)

Дата видачі завдання «12» січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: Вебзастосунок реєстрації подій

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРБ	08.01.2024 р.	12.01.2024 р.	Виконано
2.	Огляд літератури за темою роботи	15.01.2024 р.	17.01.2024 р.	Виконано
3.	Складання календарного плану КРБ	18.01.2024 р.	19.01.2024 р.	Виконано
4.	Аналіз предметної області	22.01.2024 р.	26.01.2024 р.	Виконано
5.	Розробка проєктних рішень	26.01.2024 р.	02.02.2024 р.	Виконано
6.	Моделювання та конструювання ПЗ	05.02.2024 р.	16.02.2024 р.	Виконано
7.	Розробка серверної частини застосунка	19.02.2024 р.	15.03.2024 р.	Виконано
8.	Розробка інтерфейсу користувача та його підключення до серверної частини	18.03.2024 р.	12.04.2024 р.	Виконано
9.	Тестування бізнес-вимог готового рішення та виправлення виявлених проблем	15.04.2024 р.	20.05.2024 р.	Виконано
10.	Розробка спеціальної частини з охорони праці	21.05.2024 р.	03.06.2024 р.	Виконано
11.	Оформлення КРБ та презентації	03.06.2024 р.	04.06.2024 р.	Виконано
12.	Попередній захист	05.06.2024 р.	05.06.2024 р.	Виконано
13.	Виконання рекомендацій, отриманих при попередньому захисті	06.06.2024	12.06.2024 р.	Виконано
13.	Рецензування	13.06.2024	14.06.2024	Виконано
14.	Відгук керівника КРБ	17.06.2024	21.06.2024	Виконано
16.	Захист кваліфікаційної роботи	24.06.2024	24.06.2024	

Розробив студент Адамчук І. І. _____

(прізвище, ім'я, по батькові) *(підпис)*

«19» січня 2024 р.

Керівник роботи ст. викладач Боровльова С. Ю. _____

(посада, прізвище, ім'я, по батькові) *(підпис)*

«19» січня 2024 р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Вебзастосунок реєстрації подій»

Студент 409 гр.: Адамчук Іван Іванович

Керівник: ст. викладачка, Боровльова С. Ю.

В рамках даної кваліфікаційної роботи бакалавра розроблено спеціалізований вебзастосунок для реєстрації на громадські заходи, що відповідає актуальним потребам суспільства в контексті підвищення ефективності участі в громадських заходах та волонтерстві. Об'єктом дослідження є процеси реєстрації на громадські заходи, а предметом – підходи та інформаційні технології для оптимізації та автоматизації цих процесів. Метою роботи є створення зручного вебзастосунку, який дозволяє користувачам реєструватися на різні заходи без необхідності повторного введення своїх даних. Результатом є створення ефективного інструменту для спрощення участі в заходах, що сприяє активній громадській участі та волонтерству.

У першому розділі проводиться детальний аналіз існуючих рішень у сфері реєстрації на громадські заходи та визначення основних вимог до розроблюваного вебзастосунку.

Другий розділ присвячений моделюванню архітектури вебзастосунку та деталізації його компонентів.

Третій розділ наводить різні діаграми для проєктування та конструювання вебзастосунку.

Четвертий розділ описує реалізацію вебзастосунку на основі попередньо спроєктованої архітектури та визначених технічних вимог.

КРБ викладена на 72 сторінки, вона містить 4 розділи, 25 ілюстрацій, 9 таблиць, 20 джерел в переліку посилань

Ключові слова: *вебзастосунок, реєстрація на заходи, громадська активність, волонтерство, автоматизація процесів, оптимізація реєстрації*.

ABSTRACT

of the Bachelor’s Thesis

"Web application for event registration"

Student of group 409: Adamchuk Ivan Ivanovych

Supervisor: Senior Lecturer, Borovlova S. Y.

Within the scope of this bachelor's thesis, a specialized web application for public event registration was developed, meeting the current societal needs in the context of enhancing participation efficiency in civic events and volunteerism. The research object is the processes of registering for public events, and the subject is the approaches and information technologies for optimizing and automating these processes. The aim of the work is to create a convenient web application that allows users to register for various events without the necessity to re-enter their data repeatedly. The result is the creation of an effective tool to simplify participation in events, thereby fostering active civic participation and volunteerism.

The first section provides a detailed analysis of existing solutions in the field of registration for public events and defines the main requirements for the developed web application.

The second section is devoted to modeling the architecture of the web application and detailing its components.

The third section provides various diagrams for designing and constructing a web application.

The fourth section describes the implementation of the web application based on the previously designed architecture and defined technical requirements.

The thesis is laid out on 72 pages, it contains 4 chapters, 25 illustrations, 9 tables, 20 sources in the reference list.

Keywords: *web application, event registration, civic engagement, volunteerism, process automation, registration optimization.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД	–	база даних
ПЗ	–	програмне забезпечення
СКБД	–	система керування базами даних
API	–	application programming interface
CRUD	–	create read update delete
DTO	–	data transfer object
REST	–	representational state transfer
SQL	–	Structured Query Language
UI	–	user interface
UX	–	user experience

ВСТУП

Темою кваліфікаційної роботи є «Вебзастосунок реєстрації подій».

Актуальність визначається тим, що у сучасному світі, де громадська активність та волонтерство набувають особливої ваги, виникає потреба в ефективних інструментах для організації та реєстрації на заходи. Ключовим результатом кваліфікаційної роботи є спеціалізований вебзастосунок, спрямований на спрощення та оптимізацію процесу реєстрації на різноманітні заходи. Основна проблема, яку цей проєкт має на меті вирішити, полягає в необхідності повторного введення даних користувачами при реєстрації на різні події, що є часозатратним і може стримувати потенційну участь.

Спеціалізований вебзастосунок з фокусом на «одноразову реєстрацію» розв’язує цю проблему, дозволяючи користувачам реєструватися на будь-які заходи без необхідності повторного введення своїх даних. Це не тільки покращує користувацький досвід, але й стимулює більшу участь в громадській активності та волонтерстві, важливих аспектах соціального життя сучасного суспільства. Таким чином, розробка такого вебзастосунку має велике практичне значення, адже вона сприяє підвищенню ефективності організаційних процесів і залученню громадян до активної участі в заходах, що, у свою чергу, може мати позитивний вплив на розвиток волонтерства і активної громадської участі в Україні.

В умовах війни росії проти України, актуальність розробки цього проєкту набуває нового рівня важливості. Військовий конфлікт зміцнив потребу в згуртованості населення та волонтерської підтримки, а також підкреслив значення освіти як одного з основоположних принципів, за які Україна бореться. Спрощення процесів реєстрації на волонтерські та освітні заходи через вебзастосунок стає не лише питанням зручності, але й необхідністю.

Об'єктом кваліфікаційної роботи є процеси, пов'язані з реєстрацією на громадські заходи, що включають в себе аналіз існуючих методологій реєстрації, оцінку їх ефективності та зручності для кінцевих користувачів.

Предметом кваліфікаційної роботи є підходи та інформаційні технології для оптимізації та автоматизації процесу реєстрації на заходи, з акцентом на функціонал «одноразової реєстрації».

Мета роботи – розробка вебзастосунку для оптимізація процесу реєстрації на заходи, з особливою увагою до потреб волонтерських організацій.

Для досягнення визначеної мети необхідно вирішити такі завдання:

1) провести аналіз наявних систем реєстрації на заходи та визначити їхні недоліки;
2) обрати технологічний стек для розробки вебзастосунку;
3) розробити мінімально життєздатний продукт концепції «одноразової реєстрації» для спрощення процесу участі в заходах;
4) створити простий і легко запам’ятовуваний дизайн інтерфейсу;
5) реалізувати необхідний функціонал для зберігання та обробки даних користувачів;
6) протестувати вебзастосунок на відповідність бізнес-вимогам користувачів та безпеки даних.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИЗНАЧЕННЯ ФУНКЦІЙ СИСТЕМИ

1.1 Огляд предметної області

У сфері організації подій з'являються все новіші технологічні рішення, спрямовані на поліпшення взаємодії між організаторами та учасниками. Вебзастосунки для реєстрації подій займають важливе місце в цьому процесі, дозволяючи користувачам легко знаходити, реєструватися та взаємодіяти з подіями. Вони інтегрують різноманітні функціональні можливості: від календарів подій до систем оплати та отримання зворотного зв'язку.

Центральне місце у розробці таких систем займає забезпечення зручного і інтуїтивно зрозумілого користувацького інтерфейсу, що сприяє підвищенню взаємодії та задоволеності користувачів. Сучасні платформи включають інструменти для моніторингу та аналізу взаємодії користувачів, дозволяючи організаторам оптимізувати процеси та покращити загальний досвід учасників.

Динаміка розвитку цифрових технологій також стимулює інтеграцію вебзастосунків із соціальними медіа, електронними поштовими службами та іншими онлайн-платформами, що забезпечує розширену можливість для маркетингу та реклами подій. Однак це також висуває підвищені вимоги до безпеки та конфіденційності даних, вимагаючи від розробників забезпечення високих стандартів захисту інформації.

Таким чином, область вебзастосунків для реєстрації подій виявляється в центрі перехрестя між технологічним прогресом і потребами споживачів, що змінюються. Це змушує індустрію не лише адаптуватися до поточних трендів, але й передбачати майбутні зміни, щоб залишатися конкурентоспроможними та відповідати зростаючим очікуванням користувачів.

1.2 Дослідження аналогів застосунку

Під час аналізу існуючих рішень на ринку вебзастосунків для реєстрації подій, особлива увага приділяється кільком ключовим аспектам. Ці аспекти дозволяють глибше зрозуміти поточний стан ринку та ідентифікувати потенційні можливості для інновацій у власному проєкті. Серед них:

1) функціональність: огляд функцій, які пропонують аналоги, допомагає визначити стандартні та унікальні можливості, які можуть бути важливими для користувачів, зокрема, аналізуються такі можливості, як реєстрація на події, управління подіями, створення та просування подій, а також спеціалізовані функції, такі як інтеграція з календарями;

2) технологічна платформа: важливо розглянути, які технології використовують конкуренти для реалізації своїх рішень, включаючи мови програмування, хмарні платформи та підходи до дизайну інтерфейсу, що допоможе вибрати технології, які забезпечать надійність, швидкість розвитку та зручність користування продуктом;

3) користувацький досвід (UX) та дизайн: аналізуються елементи дизайну інтерфейсу та загального користувацького досвіду, що пропонуються на ринку, це включає розгляд навігації сайту, візуального оформлення, інтуїтивності використання та доступності на різних пристроях;

4) ринкова ніша та цільова аудиторія: визначення цільових аудиторій різних аналогів дозволяє зрозуміти, які групи користувачів вони обслуговують, та як можна адаптувати свій продукт для задоволення потреб специфічних груп;

5) відгуки та репутація: вивчення відгуків користувачів і критики може виявити слабкі місця існуючих рішень, які можна вдосконалити у власному застосунку.

Цей комплексний підхід до дослідження аналогів не тільки допоможе зрозуміти поточні тенденції та виклики ринку, але й ідентифікувати ключові фактори, що впливають на успішність подібних вебзастосунків.

Виокремлення цих аспектів сприятиме формуванню ефективного та інноваційного застосунку, здатного забезпечити високий рівень задоволеності користувачів та ефективність використання.

Платформа «Волонтерство в Україні»

Розробник: SoftServe, в співпраці з UNICEF в Україні та Українською службою волонтерів [1].

Посилання: https://platforma.volunteer.country/.

Архітектура: вебзастосунок, доступний через веббраузер.

Мова реалізації: HTML, CSS, JS, інші.

Перелік функцій:

– публікація волонтерських проєктів;

– пошук волонтерських можливостей;

– можливість публікування матеріалів у вигляді блогів та статтей;

– єднання волонтерів по всій Україні;

– підтримка розвитку волонтерського руху в Україні.

Переваги: платформа «Волонтерство в Україні» дозволяє волонтерам легко знайти можливості та контакти організаторів, а організаторам – публікувати свої волонтерські проєкти.

Недоліки: на платформі не представлено можливості реєстрації на подію, лише покликання на контакт організатора або Google форма. Вебсайт перевантажений зайвими елементами дизайну, а також не зрозумілий з точки зору UX.

Eventleaf

Розробник: Jolly Technologies [2].

Посилання: https://www.eventleaf.com/.

Архітектура: хмарне рішення, яке можна використовувати через веббраузер або мобільний застосунок.

Мова реалізації: немає інформації.

Перелік функцій:

– створення та просування подій;
– надсилання запрошень учасникам;
– реєстрація учасників;
– створення власного дизайну сайту події;
– продаж квитків онлайн.

Переваги: Eventleaf надає все необхідне для організації подій, включаючи створення вебсторінок подій, керування списками учасників, надсилання запрошень та підтверджень, відстеження статусу запрошень, продаж квитків онлайн.

Недоліки: ціна за одного учасника може досягатись до 2 доларів США, що в економічних реаліях України не сприятиме розвитку галузей, де можна задіяти активних громадян.

Календар DOU

Розробник: DOU [3].

Посилання: https://dou.ua/calendar/.

Архітектура: вебзастосунок, доступний через веббраузер.

Мова реалізації: HTML, CSS, JS, інші.

Перелік функцій:

– публікація подій;
– пошук подій;
– фільтрація подій за датою та місцем;
– коментарі до подій;
– посилання на реєстрацію на події.

Переваги: сформована найбільша ІТ-спільнота України має серед функцій DOU Календар, який дозволяє користувачам легко знайти та реєструватися на ІТ-події в Україні.

Недоліки: на платформі не представлено можливості реєстрації на подію, лише покликання на контакт організатора або Google форма.

Bizzabo

Розробник: Bizzabo [4].

Посилання: https://www.bizzabo.com/.

Архітектура: хмарне рішення, яке можна використовувати через веббраузер або мобільний застосунок.

Мова реалізації: немає інформації.

Перелік функцій:

– створення та просування подій;
– організація оффлайн подій під ключ;
– створення власного дизайну сайту події;
– продаж квитків онлайн;
– управління подіями з одного центрального місця.

Переваги: Bizzabo дозволяє керувати всіма аспектами найбільших та найскладніших конференцій в одній повністю завантаженій платформі управління подіями.

Недоліки: сервіс спрямований на бізнес-авдиторію, не передбачаючи можливості організації подій для волонтерів та інших подій, що можуть бути цікавими не з комерційної точки зору.

1.3 Специфікація вимог до ПЗ

Система, для якої розробляється програмне забезпечення, є вебзастосунком для реєстрації подій. Цей застосунок призначений для спрощення процесу організації та участі у різноманітних заходах, від невеликих зустрічей та семінарів до великих конференцій та фестивалів. Основна мета системи – надати користувачам ефективний та зручний інструмент для швидкої реєстрації та управління подіями.

Межі проєкту

Проєкт обмежується розробкою вебзастосунку без розгортання декстопних програм чи інших платформ, фокусуючись на максимальній інтеграції з існуючими вебтехнологіями та сервісами.

Область застосування

Застосунок буде використовуватися для реєстрації на різноманітні події, включаючи корпоративні зустрічі, публічні заходи, семінари та конференції.

Характеристика користувачів

Для вебзастосунку реєстрації подій розрізняються різні категорії користувачів, кожна з яких має свої власні потреби, ролі та відповідальності. Зрозуміння цих категорій є важливим для належного проєктування і розробки функціоналу, який забезпечує належний досвід користування для кожного типу користувачів.

Звичайні користувачі

Користувачі, які бажають знайти і взяти участь у подіях. Вони можуть переглядати перелік подій, здійснювати пошук за різними критеріями, реєструватися на події та залишати відгуки.

Потреби: інтуїтивно зрозумілий інтерфейс, зручні функції пошуку та реєстрації, можливість легкого доступу до деталей подій.

Особливості взаємодії: використання фільтрів для пошуку, реєстрація через соціальні мережі.

Організатори подій

Організатори відповідають за створення, управління та моніторинг подій. Вони можуть додавати нові події, редагувати інформацію про події, відслідковувати реєстрації та взаємодіяти з учасниками.

Потреби: розширені інструменти управління подіями, аналітика відвідуваності, інструменти звітності.

Особливості взаємодії: доступ до адміністративної панелі, управління учасниками.

Адміністратори системи

Адміністратори відповідають за загальне управління вебплатформою, забезпечення її працездатності, безпеки та оновлення. Вони мають доступ до всіх ділянок системи.

Потреби: розширені права доступу до управління користувачами, контентом, БД і серверним середовищем.

Особливості взаємодії: налаштування системи, управління ролями і правами доступу, моніторинг стану системи.

Кожна з цих груп користувачів вимагає уваги до деталей щодо функціональності, доступу та інтерфейсу, що має бути відображено в дизайні та реалізації системи. Розуміння потреб кожного типу користувача допоможе створити ефективну та безпечну платформу, яка задовольнить очікування всіх користувачів.

Функції системи

Авторизація

Дозволяє користувачам входити в систему за допомогою облікових даних для доступу до персонального профілю.

Вхідна інформація: логін, пароль.

Вихідна інформація: підтвердження успішного входу або помилка.

Функціональні вимоги: перевірка введених даних, повідомлення про помилки, підтримка відновлення паролів.

Реєстрація

Дозволяє новим користувачам створювати обліковий запис, вводячи основні особисті дані.

Вхідна інформація: ім'я, електронна пошта, телефон, пароль.

Вихідна інформація: підтвердження реєстрації.

Функціональні вимоги: перевірка унікальності електронної пошти, валідація даних.

Особистий кабінет з переглядом подій

Надає інтерфейс для перегляду подій, на які користувач зареєстрований, та управління своїми реєстраціями.

Вхідна інформація: ідентифікатор користувача.

Вихідна інформація: список подій з деталями.

Функціональні вимоги: забезпечення конфіденційності, можливість скасування реєстрацій, актуальність даних.

Зберігання основних даних користувача (ім’я, пошта, телефон)

Відповідає за зберігання та обробку персональних даних користувачів для забезпечення ефективної комунікації.

Вхідна інформація: ім'я, електронна пошта, телефон.

Вихідна інформація: підтвердження збереження або оновлення даних.

Функціональні вимоги: можливість оновлення даних користувачем, валідація даних при введенні або оновленні.

Перегляд доступних подій

Дозволяє користувачам переглядати список всіх доступних подій, що плануються, включно з деталями кожної події.

Вхідна інформація: не потрібна.

Вихідна інформація: список доступних подій з деталями.

Функціональні вимоги: надання актуальної інформації про події, фільтрація подій за категоріями, датами тощо.

Швидка реєстрація на події в один клік

Дозволяє користувачам швидко реєструватися на події, використовуючи збережені дані, з мінімальними зусиллями.

Вхідна інформація: ідентифікатор користувача, ідентифікатор події.

Вихідна інформація: підтвердження реєстрації.

Функціональні вимоги: використання збережених даних користувача для автоматичної реєстрації, надання опції підтвердження перед реєстрацією.

Пошук подій за різними параметрами

Дозволяє користувачам шукати події за специфічними параметрами, такими як дата, місце, тип події тощо.

Вхідна інформація: параметри пошуку (дата, місце, тип події).

Вихідна інформація: список подій, що відповідають критеріям.

Функціональні вимоги: розширені функції пошуку, висока точність результатів, інтерфейс для легкого введення параметрів пошуку.

Відгуки та рейтинги подій

Дозволяє учасникам залишати відгуки та оцінки для подій, на яких вони були, з метою покращення якості майбутніх подій.

Вхідна інформація: ідентифікатор користувача, ідентифікатор події.

Вихідна інформація: підтвердження публікації відгуку.

Функціональні вимоги: забезпечення конфіденційності користувачів, фільтрація спаму, аналіз відгуків для покращення подій.

Створення та публікація подій (для організаторів)

Надає організаторам інструменти для створення нових подій та їх публікації на платформі.

Вхідна інформація: деталі події (назва, дата, час, місце, опис).

Вихідна інформація: підтвердження створення події.

Функціональні вимоги: інтерфейс для вводу інформації про подію, валідація даних, інтеграція з календарем для планування подій.

Функція обов’язкового донату при реєстрації

Вимагає від учасників здійснення донату (пожертву) при реєстрації на певні події, особливо благодійні або волонтерські заходи.

Вхідна інформація: ідентифікатор події, інформація про платіж.

Вихідна інформація: підтвердження успішної транзакції.

Функціональні вимоги: забезпечення легкої взаємодії з платіжною системою та введення відомостей про платіж.

Управління подіями (для організаторів)

Дозволяє організаторам керувати подіями, які вони створили, включаючи редагування деталей події, управління реєстраціями тощо.

Вхідна інформація: ідентифікатор події, оновлені дані події.

Вихідна інформація: підтвердження оновлення даних події.

Функціональні вимоги: інтерфейс для редагування подій, відстеження змін у реєстраціях, надання доступу до аналітики події.

Відстеження реєстрацій на події (для організаторів)

Надає організаторам можливість відслідковувати кількість та дані реєстрацій на їхні події, включаючи статистику та аналітику в реальному часі.

Вхідна інформація: ідентифікатор події.

Вихідна інформація: статистика реєстрацій на подію.

Функціональні вимоги: розробка інтуїтивного дашборду для аналітики, інтеграція з базою даних для збору і зберігання інформації, забезпечення точності даних.

Експорт таблиці з учасниками

Уможливлює експорт даних учасників подій у таблицю для звітності чи подальшого аналізу, особливо у складних умовах, як-от воєнний стан.

Вхідна інформація: ідентифікатор події.

Вихідна інформація: файл з експортованими даними учасників.

Функціональні вимоги: забезпечення безпеки при експорті даних, підтримка різних форматів файлів (наприклад, CSV, XLS), легкість використання інтерфейсу.

Інтеграція з календарем

Дозволяє користувачам додавати події до їх особистих календарів, таких як Google Calendar, для кращого планування та нагадувань.

Вхідна інформація: ідентифікатор події, вибір календаря користувача.

Вихідна інформація: підтвердження успішного додавання події в календар.

Функціональні вимоги: автоматизація додавання подій в календарі, синхронізація з основними календарними службами, забезпечення точності дат і часу подій.

Мобільна версія сайту

Надає адаптовану для мобільних пристроїв версію сайту, що забезпечує зручний доступ та навігацію з смартфонів та планшетів.

Вхідна інформація: ширина екрану пристрою.

Вихідна інформація: адаптований мобільний інтерфейс.

Функціональні вимоги: розробка адаптивного дизайну, забезпечення функціональності всіх основних функцій сайту в мобільній версії, оптимізація продуктивності для мобільних гаджетів.

Архітектура програмної системи

Архітектура програмної системи вебзастосунку для реєстрації подій базується на монолітному підході. Застосунок розбитий на різні сервіси, які пов’язані між собою, всі вони використовують одну базу даних

Системне програмне забезпечення

Visual Studio 2022, Visual Studio Code, СКБД Microsoft SQL Server Management.

Мережне програмне забезпечення

Мережне програмне забезпечення включає інструменти та протоколи для забезпечення комунікації між компонентами системи, такі як HTTPS.

Програмне забезпечення ведення інформаційної бази

СКБД SQL Server Management Studio (SSMS) є основним інтерфейсом для адміністрування та розробки у середовищі Microsoft SQL Server. Цей інструмент надає багатофункціональні можливості, які спрощують роботу з базами даних, включаючи створення, модифікацію, управління безпекою, та налаштування параметрів сервера. SSMS забезпечує інтуїтивно зрозумілий графічний інтерфейс, який полегшує доступ до різних адміністративних функцій і забезпечує зручне управління даними [5].

Мова і технологія розробки ПЗ

Для написання back-end використовувались наступні технології: .NET Core, .NET Identity, AutoMapper. Для front-end: React, Redux, Tailwind CSS. Для бази даних використовувалась MSSQL база даних.

Інтерфейс користувача

Інтерфейс користувача має бути інтуїтивно зрозумілим та зручним для користувачів з різними технічними навичками. Використання сучасних UI/UX практик допомагає забезпечити легку навігацію та доступність основних функцій.

Доступність

Система має бути доступною 24/7, забезпечуючи надійний доступ до послуг без істотних перерв в роботі.

Супроводжуваність

Програмне забезпечення має бути легким для супроводу та оновлення, з чіткою документацією та підтримкою зворотного зв'язку з користувачами.

Переносимість

Забезпечення можливості легкого перенесення системи між різними платформами та середовищами без значних змін у функціональності, шляхом відокремлення частин front-end та back-end.

Продуктивність

Система має ефективно обробляти великі обсяги даних та запитів користувачів з мінімальним часом відгуку і не перевищувати 3 секунд.

Надійність

Забезпечення стабільної роботи системи, мінімізація помилок та відмов в критичних компонентах.

Безпека

Високий рівень захисту даних користувачів від несанкціонованого доступу, використання сучасних методів шифрування та аутентифікації, використовуючи токени та зберігаючи їх в місці, не доступному локально користувачу.

Висновки до розділу 1

У першому розділі кваліфікаційної роботи бакалавра роботи здійснено ґрунтовний аналіз існуючих вебзастосунків для реєстрації подій, з акцентом на їхню функціональність, технологічну платформу, користувацький досвід, ринкову нішу, та відгуки користувачів. Цей аналіз виявив ключові аспекти, такі як важливість інтеграції з календарями, необхідність створення інтуїтивно зрозумілих інтерфейсів, і значення мобільної адаптації. Такий підхід дозволив ідентифікувати потенційні можливості для інновацій, що можуть забезпечити системі конкурентну перевагу. Також визначено основні функції для розроблюваної системи, включаючи єдину реєстрацію, персоналізовані опитування, інтеграцію з календарями, а також функції аналітики для організаторів подій. Визначення технологічного середовища та функцій системи має велике значення для розробки ефективного, зручного, і безпечного застосунку, який задовольнятиме потреби організаторів і учасників подій, а також вирішуватиме їх проблеми, полегшуючи роботу.

2 МОДЕЛЮВАННЯ ВЕБЗАСТОСУНКУ РЕЄСТРАЦІЇ ПОДІЙ

2.1 Сценарії використання системи

На наступному рисунку 2.1 можемо побачити діаграму використання системи.

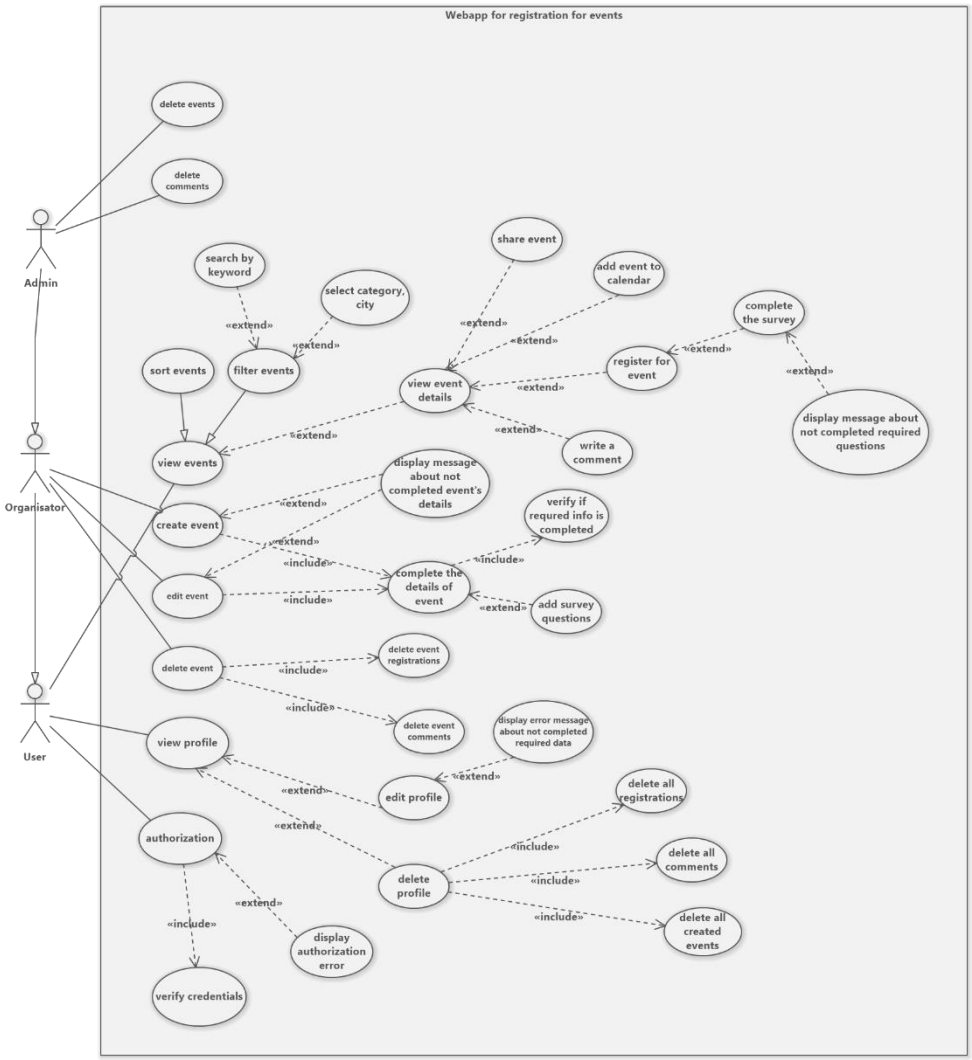

Рисунок 2.1 – Use-case діаграма

Діаграма використання, представлена вище, детально описує різні взаємодії користувачів (адміністратора, організатора та звичайного користувача) з вебзастосунком для реєстрації подій. Вона ілюструє ключові функціональні можливості системи та зв'язки між різними діями [6].

Додатково створимо 3 різні типи для опису випадків використання: коротку, поверхневу, та повну.

Опишемо сценарій «Реєстрація користувача» коротким методом. Користувач заходить на сайт реєстрації подій, обирає опцію «Створити обліковий запис», вводить необхідні особисті дані, такі як ім'я, електронну пошту та пароль, та підтверджує реєстрацію. Система перевіряє введені дані, надсилає лист з підтвердженням на електронну пошту для верифікації облікового запису. Користувач переходить за посиланням у листі, що активує обліковий запис. Після активації користувач може увійти на сайт під своїм обліковим записом та почати користуватися послугами сайту.

Застосуємо поверхневий метод опису сценарію використання, для прикладу візьмемо case «Реєстрація на подію». Користувач, вже зареєстрований у системі, увійшовши на сайт, переходить до розділу подій та вибирає подію, яка його цікавить. Переглядає деталі події, включаючи дату, час, місце проведення, та вимоги до учасників. Натискає кнопку «Зареєструватися» та, якщо потрібно, заповнює додаткову інформацію, на вимогу організаторів. Система обробляє реєстрацію, відображає підтвердження участі та надсилає лист з деталями події на електронну пошту користувача.

Альтернативні сценарії:

1) подія заповнена: якщо учасник намагається зареєструватися на подію, яка вже не має вільних місць, система повідомляє про це та пропонує альтернативні події;

2) технічні проблеми при реєстрації: у випадку виникнення технічних проблем під час процесу реєстрації, користувач отримує повідомлення з пропозицією спробувати пізніше або звернутися за допомогою;

3) зміна інформації про учасника: користувач має можливість оновити свою особисту інформацію перед реєстрацією на подію, якщо це необхідно для участі;

4) спеціальні вимоги до учасників: якщо для участі в події існують спеціальні вимоги (наприклад, донат), система повідомляє користувача перед реєстрацією;

5) відміна реєстрації: якщо користувач бажає відмінити свою реєстрацію на подію, система дозволяє це зробити через інтерфейс користувача, надаючи інформацію про можливі наслідки (наприклад, втрату місця або зборів за участь).

Розглянемо повний сценарій використання система на прикладі основної функції системи – реєстрація на подію.

Use Case Name: реєстрація події.

Scope: System.

Level: User-goal.

Primary Actor: організатор події.

Stakeholders and Interests: організатори подій бажають легко створювати та управляти подіями, залучаючи учасників. Користувачі шукають різноманітні події для участі.

Preconditions: Організатор має обліковий запис і увійшов в систему.

Success Guarantee: Подія успішно створена та доступна для реєстрації учасниками.

Main Success Scenario:

1) організатор входить до системи за допомогою свого облікового запису;

2) переходить до розділу для створення подій;

3) вводить інформацію про подію, включаючи назву, дату, час, місце проведення, та опис;

4) вказує додаткові питання (якщо є);

5) додає зображення (якщо є);

6) встановлює ліміт учасників (якщо потрібно);

7) вибирає категорію події для легшого пошуку користувачами;

8) обирає які контактні дані відображати;

9) надсилає форму для створення події;

10) система перевіряє введені дані та публікує подію на сайті.

Extensions:

– зміна деталей події після публікації: організатор може оновити інформацію про подію, що вже опублікована, якщо змінилися дата, час, місце проведення або інші важливі деталі;

– скасування події: організатор має можливість скасувати подію через непередбачені обставини;

– перевищення ліміту реєстрацій: якщо кількість реєстрацій перевищує ліміт учасників, система автоматично закриває можливість реєстрації та повідомляє організатора;

– технічні проблеми з публікацією: у випадку технічних проблем з публікацією події, система повідомляє організатора та пропонує можливі шляхи вирішення.

Special Requirements: мобільна адаптація для створення та управління подіями.

Technology and Data Variations List: використання API для імпорту подій з зовнішніх календарів.

Frequency of Occurrence: 40%.

Miscellaneous: забезпечення доступності та зручності інтерфейсу для організаторів.

2.2 Побудова діаграм взаємодії

Створення діаграм взаємодії дає можливість надати детальний огляд процесів взаємодії користувачів із системою, а також внутрішніх процесів обробки даних системою [6]. Наступні діаграми ілюструють кроки, що виконуються при основних сценаріях використання системи. Ці діаграми

важливі для забезпечення зрозумілості логіки роботи системи і для визначення точок інтеграції різних компонентів.

Діаграма реєстрації користувача (див. рис. 2.2) показує етапи, які користувач проходить для створення нового облікового запису, від доступу до форми реєстрації до отримання підтвердження реєстрації. В цьому процесі реєстрації дані перевіряються і зберігаються у базі даних, з використанням сервісу автентифікації.

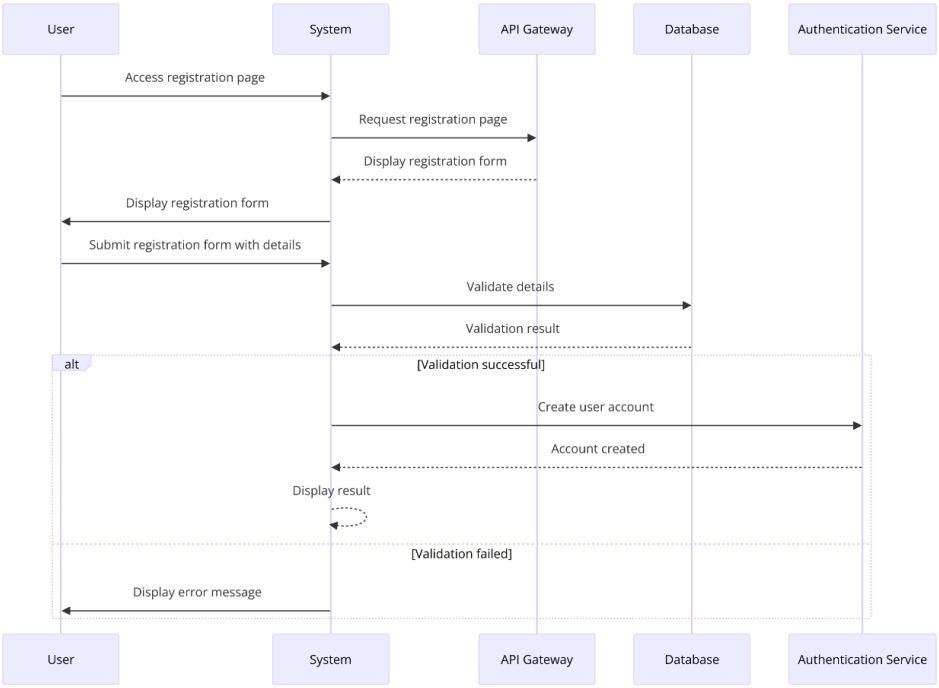

Рисунок 2.2 – Діаграма «Реєстрація користувача»

Діаграма створення події (див. рис. 2.3) ілюструє процес, який організатори використовують для додавання нової події до системи. Вона включає в себе валідацію інформації про подію, перевірку прав доступу організатора і збереження деталей події у базі даних через сервіс управління подіями.

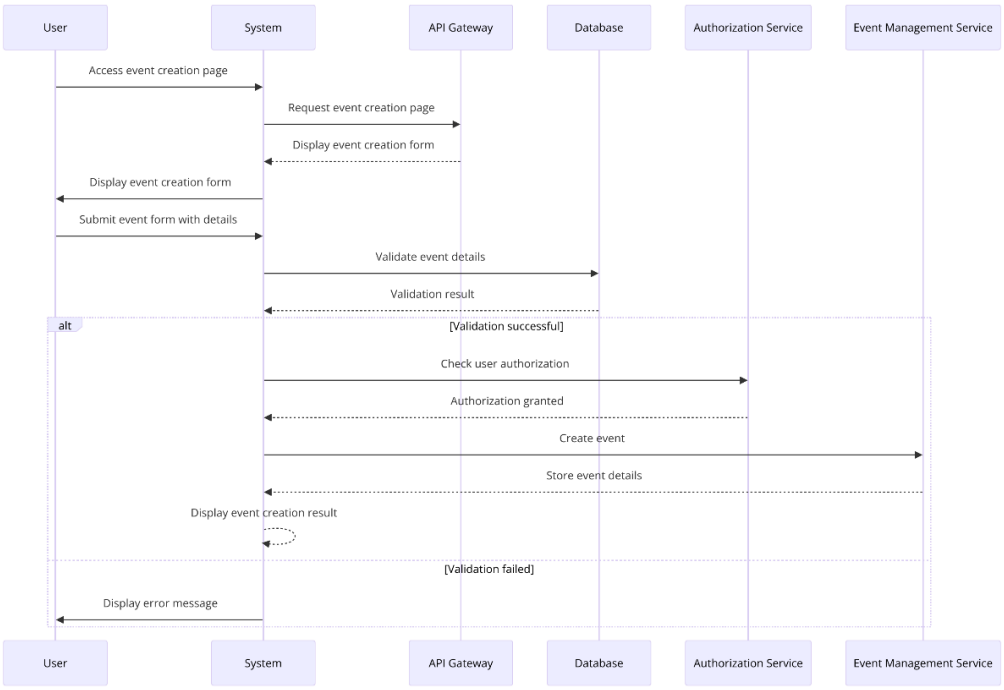

Рисунок 2.3 – Діаграма «Створення події»

Діаграма перегляду подій (див. рис. 2.4) показує, як користувачі можуть переглядати різні події. Вона включає процеси збору інформації про події з бази даних, застосування фільтрів користувача і надання доступу до переліку подій.

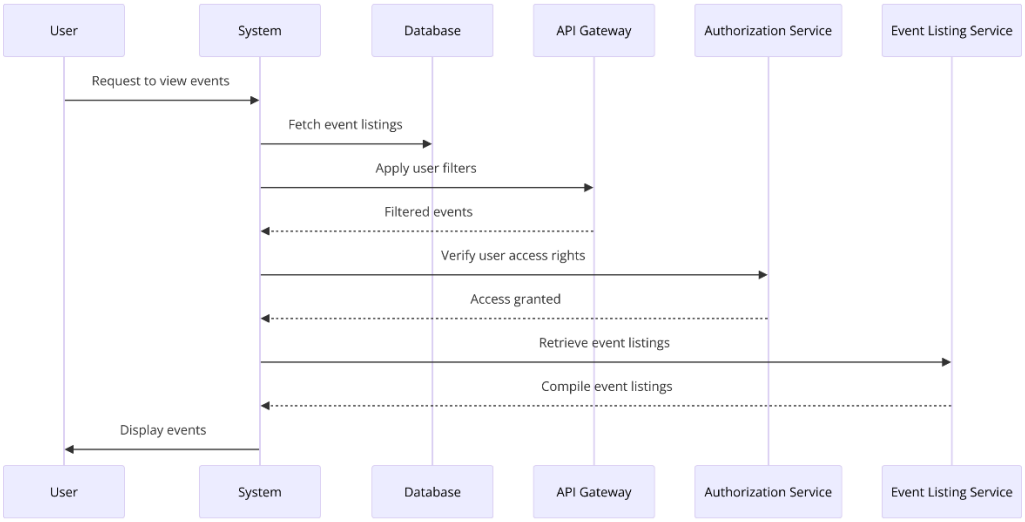

Рисунок 2.4 – Діаграма «Перегляд всіх подій»

Наступна діаграма (див. рис. 2.5) описує кроки, необхідні для реєстрації на події, включаючи перевірку доступності місць, авторизацію користувача, можливого опитування (якщо є) та обробку інформації про участь.

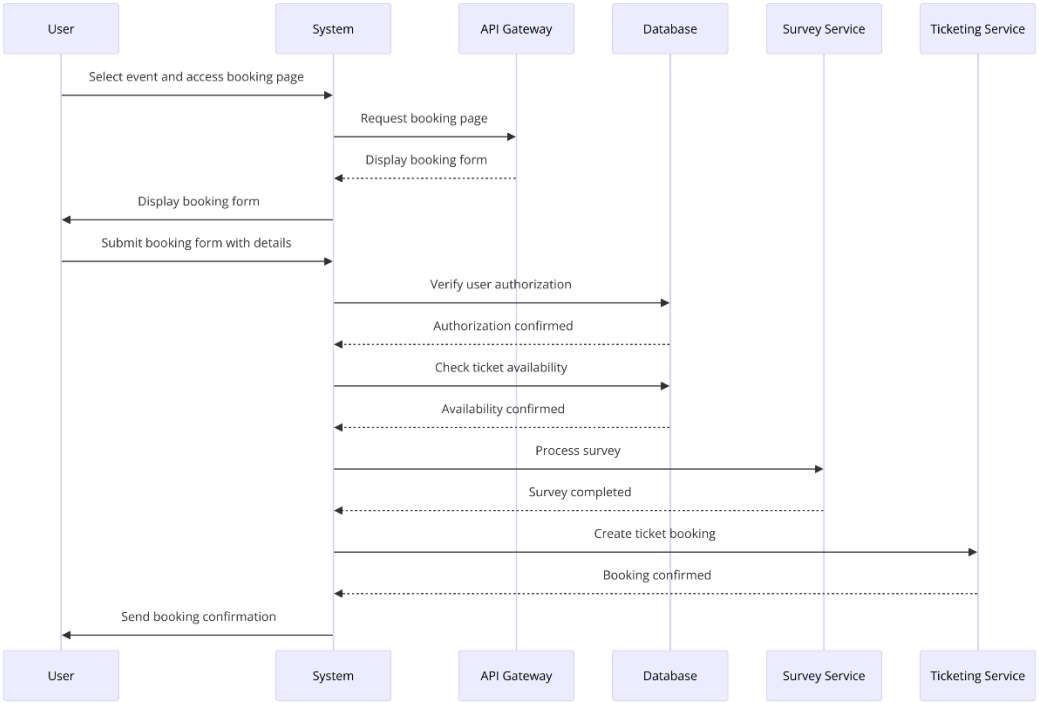

Рисунок 2.5 – Діаграма «Реєстрація на подію»

Діаграма залишення коментаря (див. рис. 2.6) відображає процес, за допомогою якого учасники події можуть залишити свої думки до та після події. Відбувається перевірка авторизації користувача, збір деталей відгуку і їх збереження у базі даних.

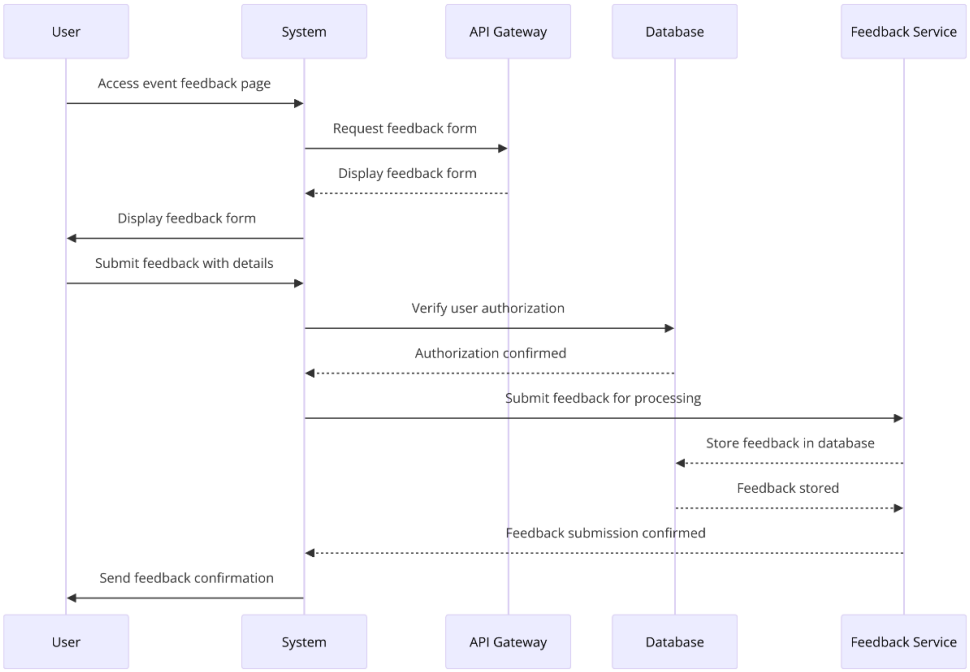

Рисунок 2.6 – Діаграма «Коментування події»

Діаграма пошуку подій (див. рис. 2.7) демонструє, як користувачі можуть шукати події, вводячи критерії пошуку, які потім обробляються для повернення відповідних результатів.

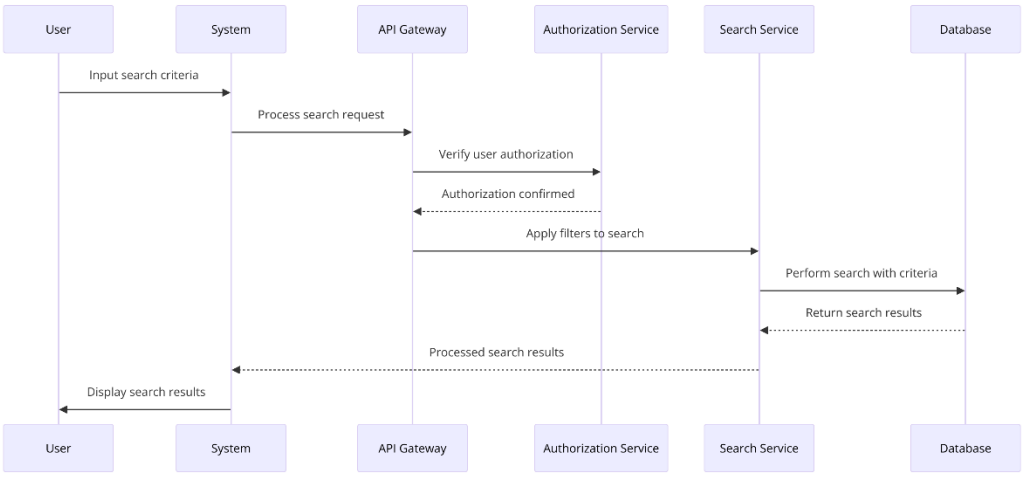

Рисунок 2.7 – Діаграма «Пошук подій»

У цьому процесі відбувається перевірка прав користувача та фільтрація подій згідно із запитом.

2.3 Створення діаграми станів та переходів

Діаграми станів та переходів є важливими для візуалізації поведінки системи, зокрема як система реагує на різні події чи взаємодії з користувачами [6].

На прикладі поданої діаграми (див. рис. 2.8) можна побачити візуалізацію станів та переходів для типових користувацьких взаємодій на вебзастосунку реєстрації подій. Діаграма охоплює весь процес, від входу на головну сторінку до реєстрації на події та інтеграції з календарем.

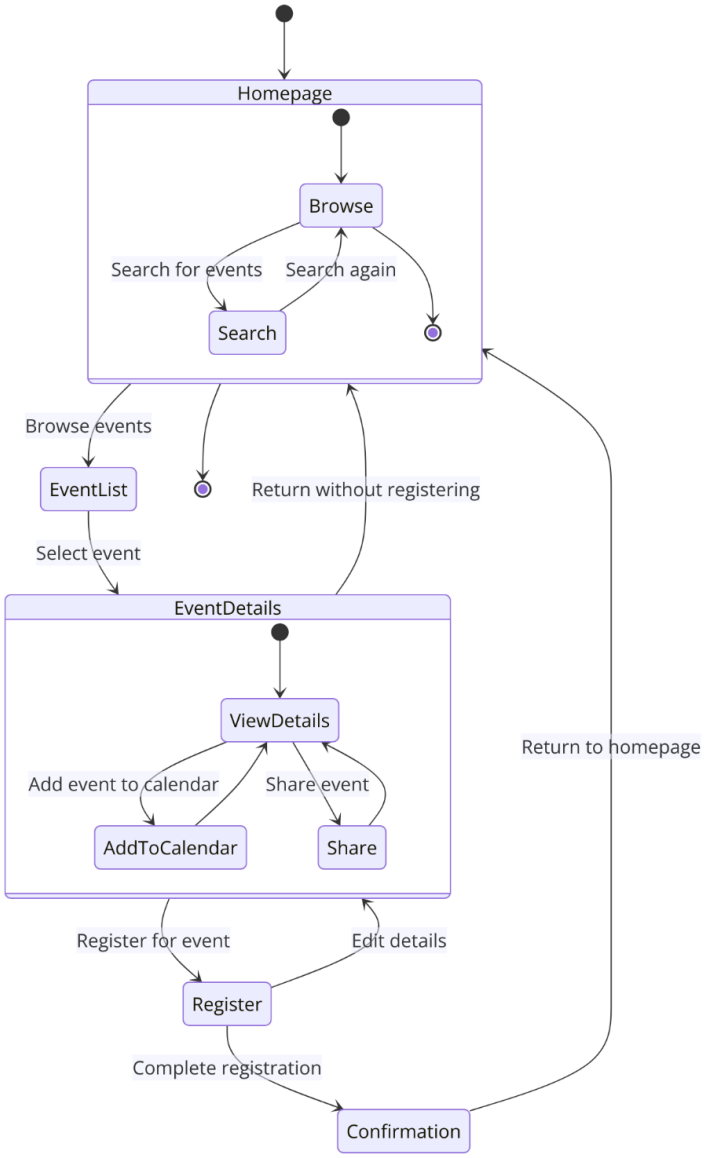

Рисунок 2.8 – Діаграма станів та переходів системи

Основні компоненти діаграми:

1) Homepage: стан, де користувач може вибрати «Browse» для перегляду подій або «Search» для пошуку подій за параметрами;

2) EventList: відображається після вибору «Browse» або завершення пошуку, звідси користувач може обрати подію для детального перегляду або повернутися на головну сторінку без реєстрації;

3) EventDetails: стан, який активується після вибору конкретної події. Користувач може переглянути деталі події, поділитися подією, додати подію до календаря, зареєструватися на подію;

4) Register: процес реєстрації на подію, що включає введення необхідних даних та підтвердження реєстрації;

5) Confirmation: фінальний стан, де користувач отримує підтвердження про успішну реєстрацію на подію.

2.4 Моделювання діаграм діяльності

Діаграм діяльності ілюструють послідовність операцій та потоків управління в системі для визначених бізнес-процесів [6]. Діаграми діяльності відображають не лише дії, що виконуються в рамках окремих процесів, але й умови, які впливають на переходи між різними станами або етапами цих процесів.

Перша діаграма (див. рис. 2.9) демонструє поведінку користувача при відвідуванні вебсайту подій. Вона починається з доступу користувача до сайту та пошуку подій. У випадку, якщо подія знайдена, користувач може її обрати і зареєструватися на неї, перед цим перевіривши наявність авторизації. Якщо користувач не авторизований, йому пропонується увійти в систему або зареєструватися. Після авторизації користувач отримує екран-підтвердження на реєстрацію події.

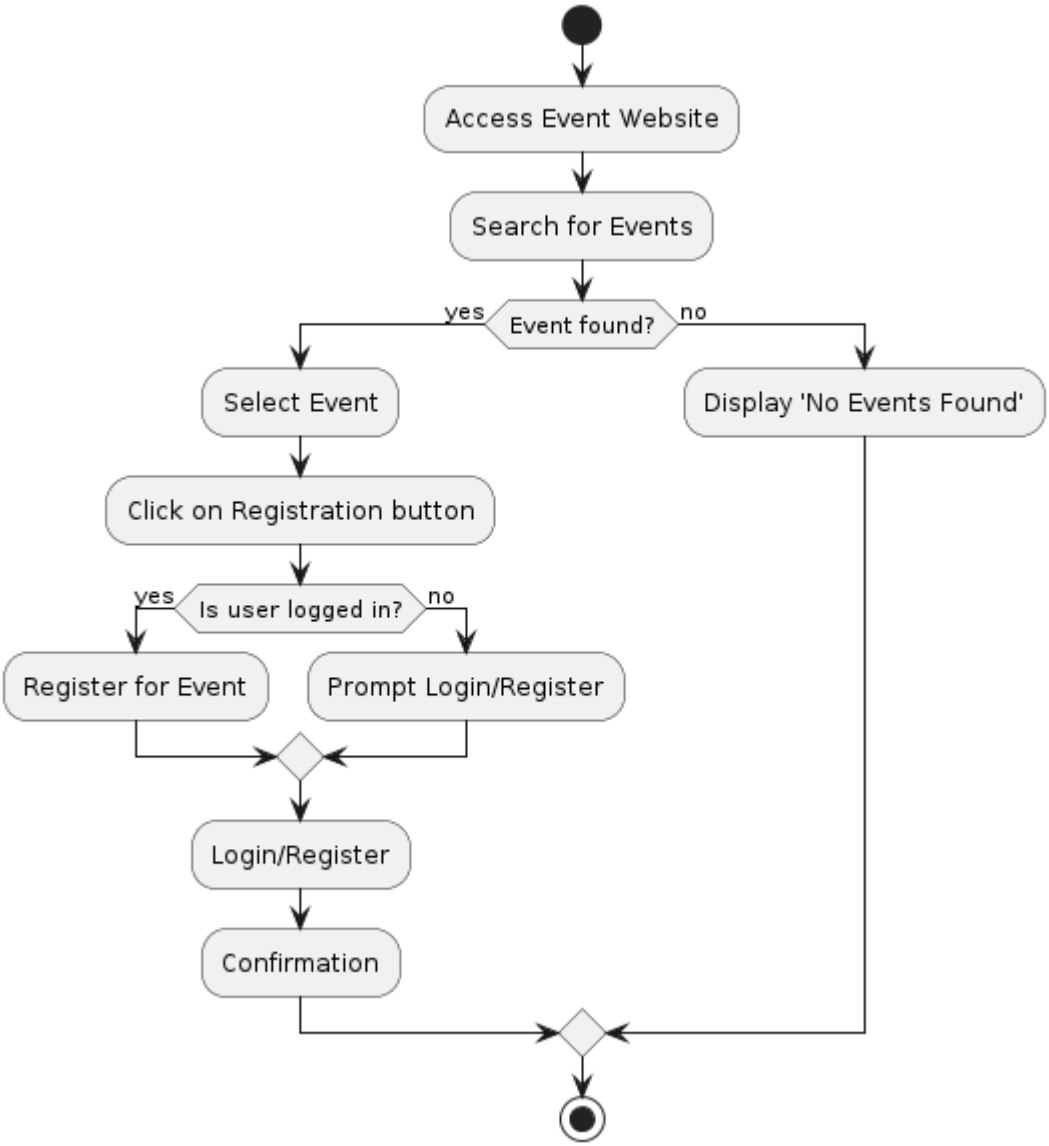

Рисунок 2.9 – Діаграма діяльності «Пошук та реєстрація на подію»

Наступна діаграма управління подіями (див. рис. 2.10) відображає процес взаємодії організатора з панеллю управління подіями. Починаючи з доступу до панелі управління, організатору необхідно пройти авторизацію. Після успішного входу, організатор може створювати нові події, вводячи їх деталі, додаткові питання та публікуючи їх. Якщо деталі події введені неповністю, система відображає відповідне повідомлення.

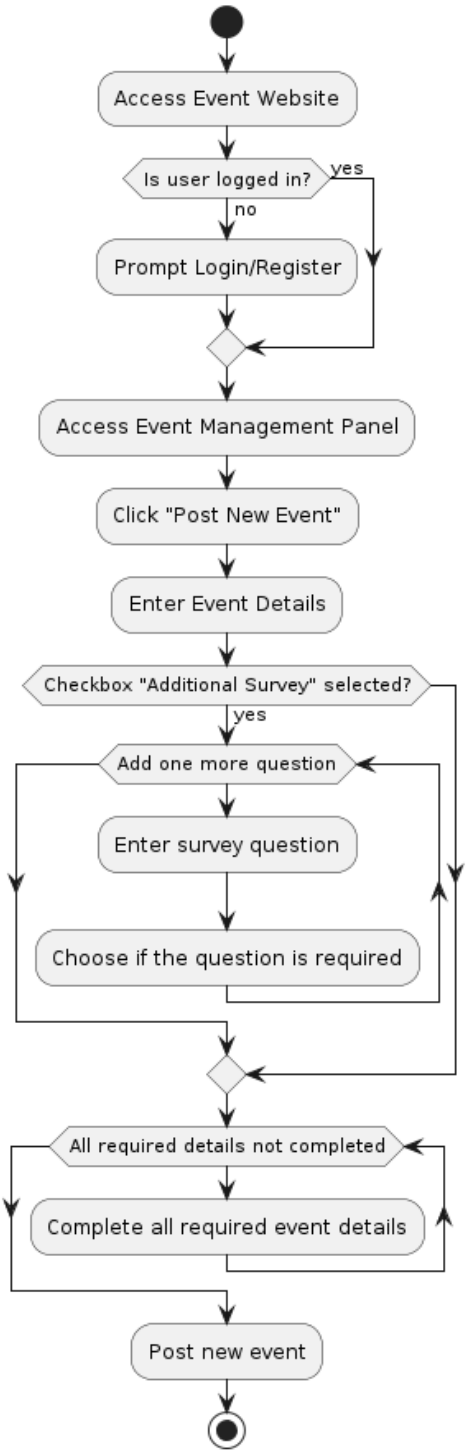

Рисунок 2.10 – Діаграма діяльності «Створення нової події»

Змоделювавши подані діаграми взаємодії з системою, можна уникнути помилок у реалізації бізнес-логіки і забезпечити кращий користувацький досвід.

2.5 Дизайн мокапів мобільної версії вебзастосунку

Сучасні тенденції вебдизайну відображають ріст важливості візуальної атрактивності та користувацького досвіду. Адаптивний дизайн продовжує бути ключовим, особливо з урахуванням зростання мобільного використання інтернету, що вимагає вебсайтів, оптимізованих для всіх типів пристроїв. Інтерактивний сторітеллінг та емоційне залучення користувачів через дизайн є ще однією тенденцією, яка допомагає брендам створювати більш запам'ятовувані вебдосвіди. Ці тенденції спонукають дизайнерів інтегрувати більш складні візуальні ефекти та анімацію, що робить вебсайти не просто інформативними, але й тими, які надихають на дії [7].

Також спостерігається тенденція до персоналізації вебдизайну, де вебсайти стають більш адаптованими до індивідуальних потреб і вподобань користувачів. Використання індивідуально створених фотографій, ілюстрацій, іконок та інших елементів, що відображають унікальність бренду, стає нормою. Це відходження від уніфікованих шаблонів до більш оригінального та особистого підходу в дизайні, що відповідає прагненню до більшої індивідуалізації у всіх аспектах споживчого досвіду [7].

З урахуванням трендів у вебдизайні створено мокапи Mobile First дизайну [8], які мають на меті сформувати UX-частину вебзастосунку, яка відповідатиме вимогам користувачів. На головній сторінці розташовані два розділи, на які може перейти користувач: «Усі події» та «Мої події» (див. рис. 2.11). Також користувачу буде доступна функція додавання нової події через кнопку. Нижче – каталог подій, які потенційно можуть бути цікаві користувачу. В кінці сторінки відображатиметься статистика вебсайту.

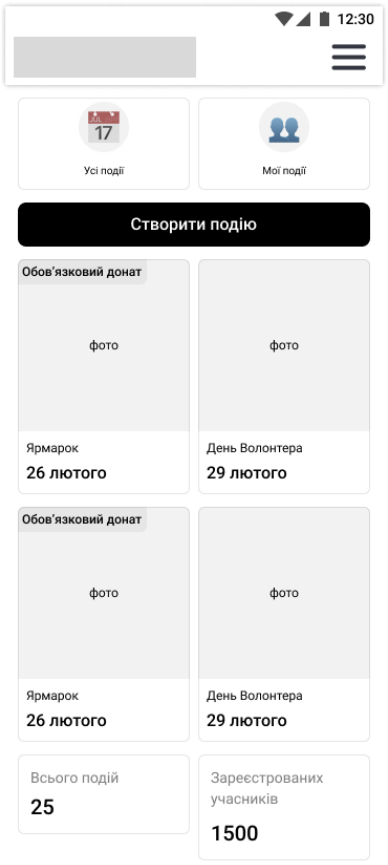

Рисунок 2.11– Мокап головної сторінки

Для реєстрації користувач може обрати кілька варіантів, швидкий – через обліковий запис Google, а також присутня можливість реєстрації через електронну пошту та пароль (див. рис. 2.12).

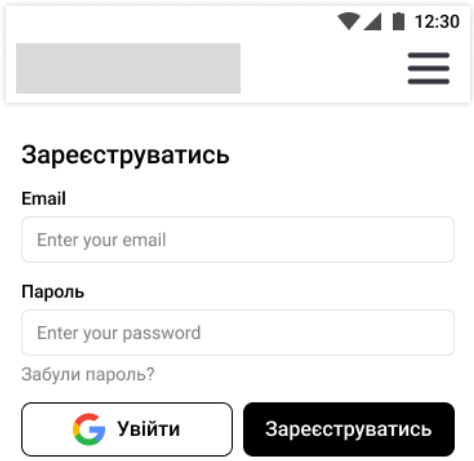

Рисунок 2.12 – Мокап сторінки реєстрації

На сторінці профілю (див. рис. 2.13) користувач матиме можливість змінити дані, які передаватимуться при реєстрації. Також для організаторів подій доступний розділ керування подіями. У розділі налаштувань клієнт має можливість налаштувати сповіщення, а також видалити всі свої дані з платформи, включно з обліковим записом.

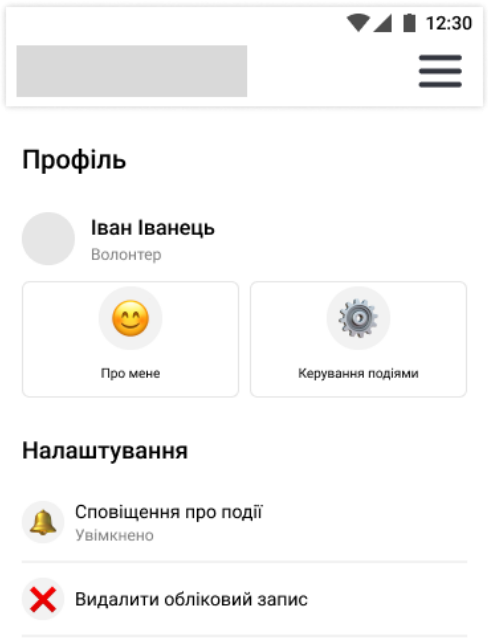

Рисунок 2.13 – Мокап сторінки профілю

На сторінці події (див. рис. 2.14) головний акцент спрямовується на локацію події, після неї – головна інформація про подію: назва, дата, контакти організатора, можливість зареєструватись на подію, а також додати її у календар.

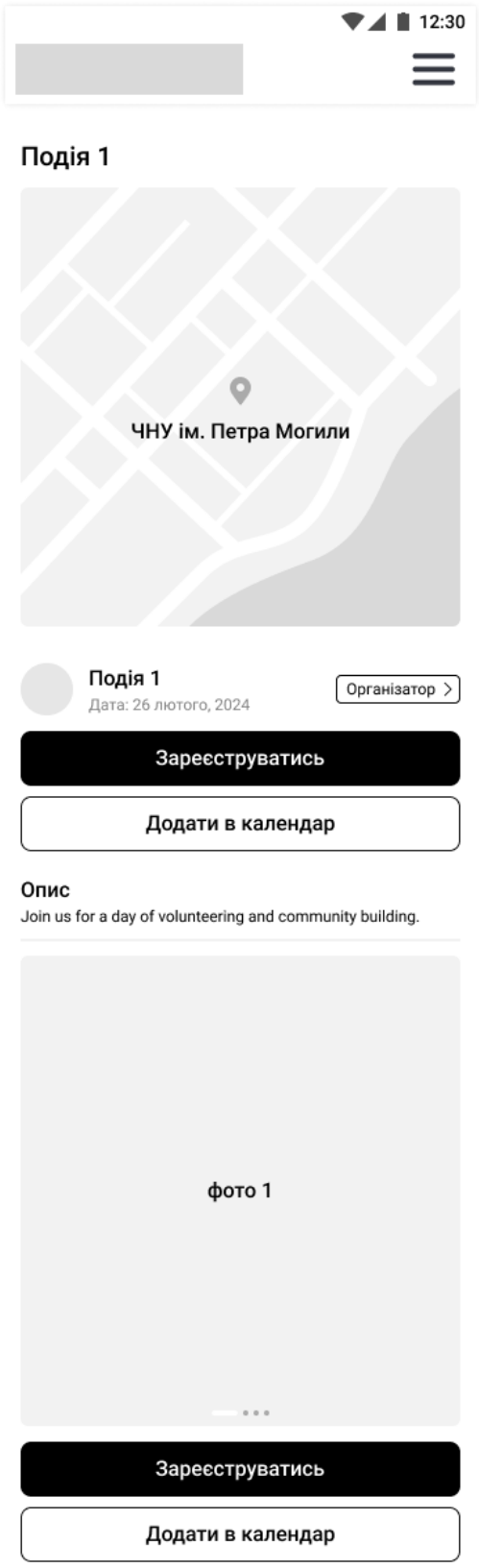

Рисунок 2.14 – Мокап сторінки події

Після основної інформації є розділ з детальним описом події, а також фотографіями.

Висновки до розділу 2

У другому розділі кваліфікаційної роботи бакалавра розглянуто моделювання вебзастосунку реєстрації подій, детально описуючи структуру системи, сценарії використання, процеси розробки діаграм розгортання, взаємодії, станів та переходів, а також діяльності. Деталізоване викладення архітектури системи, включаючи як фронтенд, так і бекенд, виявило здатність системи ефективно обробляти запити користувачів і взаємодіяти з базою даних через добре структуровану логіку на стороні сервера та клієнта. Така структура сприяє не тільки високій продуктивності, але й легкості супроводження та масштабування системи.

Аналіз варіантів використання системи підкреслює важливість чіткої дефініції користувацьких сценаріїв, зокрема реєстрації користувачів, створення та управління подіями, що забезпечує розуміння взаємодії між користувачами та системою. Розробка діаграм взаємодій та станів дозволяє візуалізувати процеси в системі, забезпечуючи ясність у взаємодіях і потоках даних, що є критично важливим для забезпечення надійності та інтуїтивно зрозумілої поведінки вебзастосунку.

Крім того, детальне моделювання діяльності і дизайн мокапів мобільної версії вебзастосунку вказують на високий рівень уваги до користувацького досвіду, що підтверджує зосередженість на зручності кінцевих користувачів. Це, у свою чергу, відкриває шлях для створення ефективного та привабливого вебзастосунку, орієнтованого на забезпечення високої взаємодії і задоволення потреб користувачів.

3 ПРОЄКТУВАННЯ ТА КОНСТРУЮВАННЯ ВЕБЗАСТОСУНКУ

Структура системи для вебзастосунку реєстрації подій розподілена на дві основні частини: фронтенд (Front-end) і бекенд (Back-end).

Запити користувача обробляються компонентами React, які взаємодіють з Redux для управління станом. Redux відправляє і отримує дані через RESTful API, яке перенаправляє запити до бекенд сервісів. Сервіси обробляють бізнес-логіку та здійснюють взаємодію з базою даних через DbContext.

Всі дані зберігаються в централізованій базі даних, що дозволяє системі бути консистентною та надійною.

RESTful API (Representational State Transfer Application Programming Interface) – це стиль архітектури програмних інтерфейсів, який використовується для розробки вебсервісів. REST використовує стандартні HTTP-методи, такі як GET, POST, PUT та DELETE для маніпулювання даними. Цей підхід до розробки API дозволяє легко інтегрувати різноманітні вебсервіси та є популярним через свою простоту та легкість у використанні. RESTful API зазвичай працює з ресурсами, кожен з яких ідентифікується через URI (Uniform Resource Identifier). Клієнт взаємодіє з цими ресурсами, відправляючи запити до сервера, який відповідає представленнями ресурсу, зазвичай у форматі JSON або XML. Це дозволяє розробникам легко обмінюватися даними між різними платформами та застосунками без необхідності узгодження складних інтерфейсів. Один з основних принципів REST – становість, тобто кожен запит від клієнта містить всю необхідну інформацію для його обробки. Сервер не зберігає стан сесії, що забезпечує вищу масштабованість і надійність системи. Відповіді на запити можуть кешуватися для підвищення продуктивності [9].

Таблиця 3.1 – Опис структури Front-end частини

Назва	Опис
React Component	Це базові елементи інтерфейсу користувача, розроблені з використанням бібліотеки React. Компоненти React забезпечують динамічне відображення вебсторінок і взаємодію з користувачем [10]. Компоненти реагують на дії користувача та відправляють дані в Redux для подальшої обробки [10].
Redux	Використовується для управління станом застосунка. Redux зберігає стан всієї аплікації у вигляді єдиного об’єкта, який допомагає управляти станом реактивних компонентів і взаємодією між ними [11]. Redux відправляє запити до API на бекенді та обробляє відповіді, що дозволяє синхронізувати дані між користувачем і сервером.

Таблиця 3.2 – Опис структури Back-end частини

Назва	Опис
API (Controllers)	API слугує мостом для комунікації між фронтендом і бекендом. Він приймає запити від фронтенду, передає їх у відповідні сервіси для обробки, і повертає результати назад у фронтенд. API використовує REST архітектуру для обробки HTTP запитів, що дозволяє легко інтегрувати різноманітні типи клієнтських додатків [9].
Service	Сервісний шар містить бізнес-логіку застосунка і відповідає за обробку даних, отриманих від API. Сервіси виконують всі необхідні операції, які потрібні для роботи системи, такі як валідація даних, обробка запитів до бази

Кінець таблиці 3.2

Назва	Опис
	даних тощо. Сервіси також забезпечують зв'язок з DbContext для взаємодії з базою даних.
DbContext	DbContext відповідає за взаємодію з базою даних. Це компонент, що дозволяє запитувати і зберігати дані в базі даних за допомогою об'єктно-реляційного відображення. Використовується для виконання CRUD операцій на даних, які зберігаються у базі даних.
Database	Центральне сховище даних, де зберігаються всі дані, необхідні для роботи системи. БД може включати таблиці для зберігання інформації про користувачів, події, реєстрації, відгуки тощо. Забезпечує високий рівень збереження і доступності даних для системи.

Описана структура відображена на схемі (див. рис. 3.1).

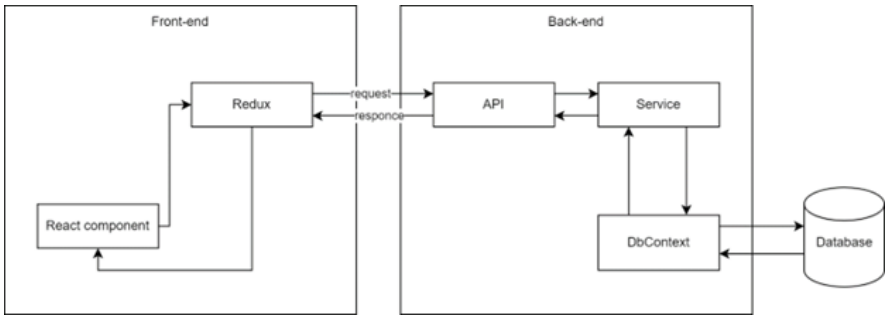

Рисунок 3.1 – Схема структури системи

Ця структура дозволяє чітко розділити відповідальності між різними частинами системи, що сприяє легшій підтримці та масштабуванню системи.

3.1 Розробка діаграми розгортання

Діаграма розгортання відіграє ключову роль у візуалізації архітектури фізичної розгортки компонентів системи та їх взаємозв'язків (див. рис. 3.2) [6]. На діаграмі зображені три основні компоненти системи: Front-end, Back-end та БД.

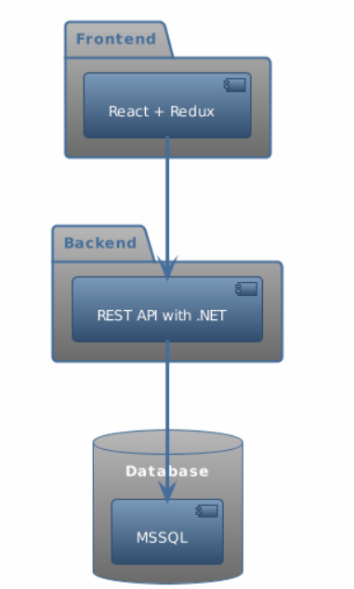

Рисунок 3.2 – Діаграма розгортання системи

Фронтенд виконує роль клієнтського інтерфейсу, де користувачі взаємодіють з вебзастосунком. Використання React дозволяє створювати інтерактивні UI компоненти, тоді як Redux забезпечує управління станом застосунку для забезпечення послідовної взаємодії.

Бекенд обробляє запити від фронтенду, виконує бізнес-логіку та взаємодіє з базою даних. .NET використовується як API [12], котре бере за основу REST архітектуру, забезпечуючи легку інтеграцію та стандартизований обмін даними.

Microsoft SQL Server використовується як центральна БД для зберігання і управління всіма даними вебзастосунку.

3.2 Діаграма класів

Діаграма класів є ключовим компонентом в процесі проєктування та конструювання вебзастосунку для реєстрації подій. Ця діаграма забезпечує структуроване візуальне представлення класів, які формують систему, їхніх атрибутів, методів та взаємозв'язків між ними. Вона допомагає розробникам і архітекторам системи зрозуміти бізнес-логіку та технічні аспекти реалізації вебзастосунку. Основна мета діаграми класів – забезпечити чітке зображення об'єктно-орієнтованої структури системи.

Кожен клас на діаграмі представляє собою окремий компонент чи модуль вебзастосунку, який виконує визначені функції. Наприклад, клас «Event» має атрибути, такі як title, description, та start date, та методи для управління реєстрацією та автентифікацією.

Атрибути класу детально описують характеристики об'єктів, які будуть створені з цього класу. Вони включають дані, які клас зберігатиме, наприклад, ім'я, адресу електронної пошти, контактні дані користувачів.

Методи класу визначають функції або дії, які можуть бути виконані об'єктами класу. Це можуть бути операції як створення нового події, реєстрація на подію, або отримання інформації про подію.

Зв'язки на діаграмі показують, як класи взаємодіють між собою. Наприклад, асоціації, агрегації чи композиції. Зв'язки можуть вказувати на те, що один клас використовує або є частиною іншого. Зв'язки допомагають зрозуміти залежності та взаємодії в системі.

Ця діаграма класів (див. рис. 3.3) є основою для подальшої розробки вебзастосунку, оскільки вона допомагає визначити, як дані будуть організовані та як будуть взаємодіяти різні частини системи.

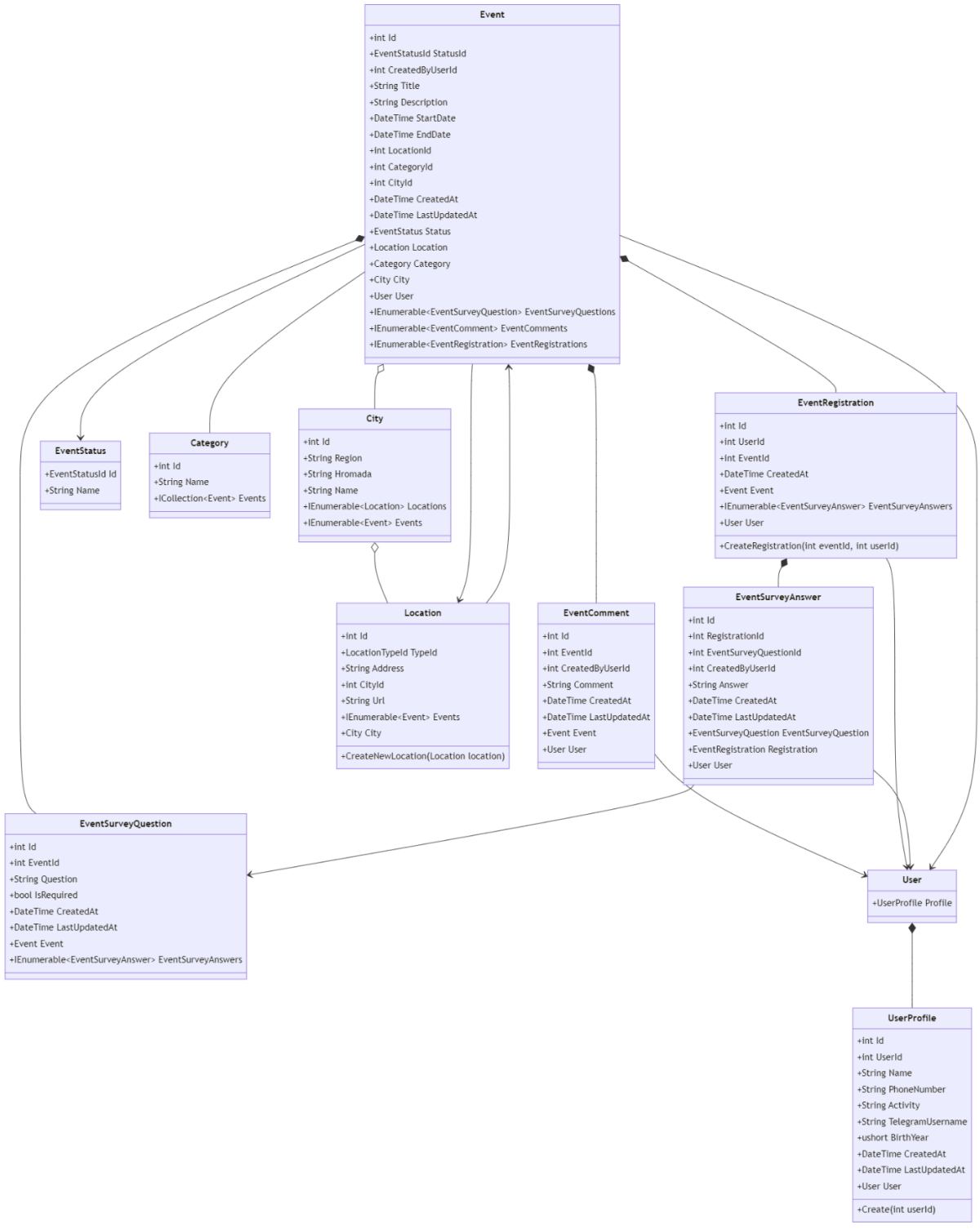

Рисунок 3.3 – Діаграма класів

Вона також забезпечує основу для кодування об'єктно-орієнтованих класів у мові програмування, забезпечуючи що вебзастосунок буде масштабованим, ефективним та легко супроводжуваним.

3.3 Діаграма компонентів

Діаграма компонентів надає чітке візуальне зображення структурних взаємозв'язків у вебзастосунку для реєстрації подій. Діаграма поділена на три основні секції: Front-end, Back-end і База даних, ілюструючи взаємодію цих компонентів через різні інтерфейси та потоки даних.

Front-end:

React Component: Цей компонент відповідальний за рендеринг користувацького інтерфейсу. Він динамічно взаємодіє з Redux Store для відображення даних та обробки взаємодій користувачів.

Redux Store: Діє як центральний контейнер для управління станом, який керує станом по всьому застосунку. Він отримує дії від React Components, які використовує для оновлення стану і забезпечення реактивного відображення даних компонентам.

Back-end:

API Controller: Слугує точкою входу для запитів від front-end. Він обробляє вхідні API-виклики і направляє їх до відповідних служб у структурі back-end.

Service Layer: Впроваджує бізнес-логіку застосунку. Він обробляє логіку, необхідну для виконання операцій, запитуваних API-викликами, забезпечуючи правильне застосування бізнес-правил та валідацій.

Repository: Відповідальний за операції доступу до даних. Він спілкується з базою даних для отримання, оновлення чи зберігання даних, діючи як посередник між Service Layer та базою даних.

База даних:

MSSQL Database: Система управління реляційними базами даних, яка використовується для безпечного зберігання та управління всіма даними застосунку. Repository взаємодіє з цим компонентом для виконання SQL-запитів, забезпечуючи збереження та отримання даних.

Front-end спілкується з Back-end через HTTP API виклики, ініційовані React Components і управляються Redux Store для підтримки консистентності стану застосунку.

Back-end обробляє ці виклики за допомогою API Controller, який делегує завдання Service Layer. Service Layer використовує Repository для взаємодії з Базою даних, виконуючи необхідні CRUD (створення, читання, оновлення, видалення) операції.

Потік даних назад до front-end слідує у зворотньому напрямку, де результати запитів передаються від Бази даних до Repository, потім до Service Layer і, нарешті, назад до API Controller, який відповідає на запит front-end.

Діаграма компонентів (див. рис. 3.4) ефективно демонструє модульну архітектуру системи.

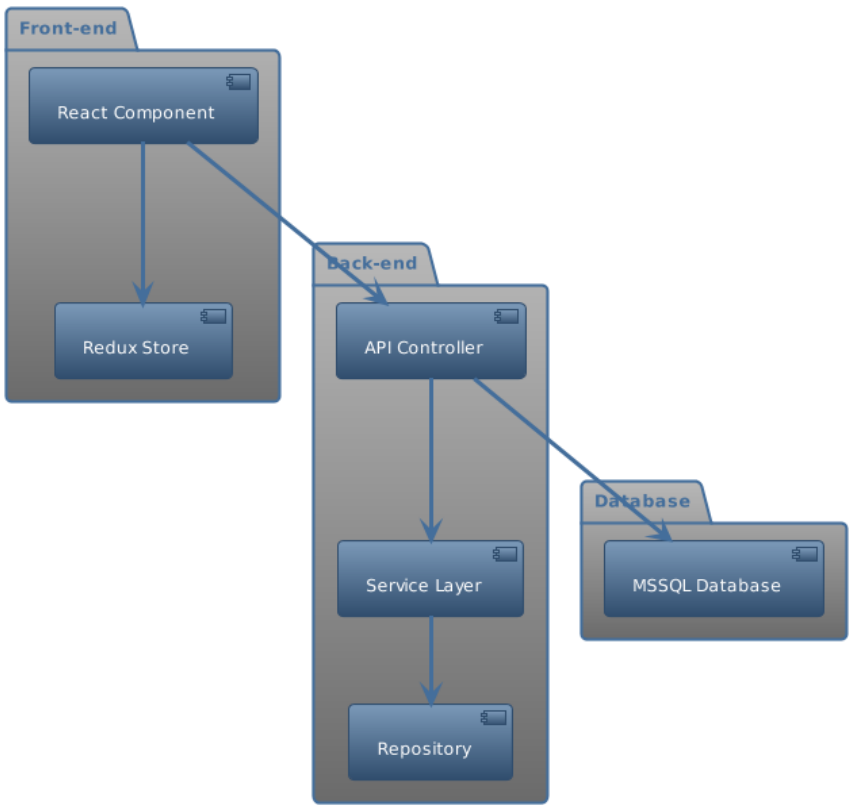

Рисунок 3.4 – Діаграма компонентів системи

Діаграма підкреслює чітке розділення відповідальностей, що сприяє легкості обслуговування та масштабуванню застосунку.

3.4 Діаграма пакетів

На діаграмі пакетів представлено структуру програмної системи (див. рис. 3.5), що включає різні компоненти вебзастосунку, які взаємодіють між собою для оптимізації процесу розробки та забезпечення чіткого розділення відповідальностей.

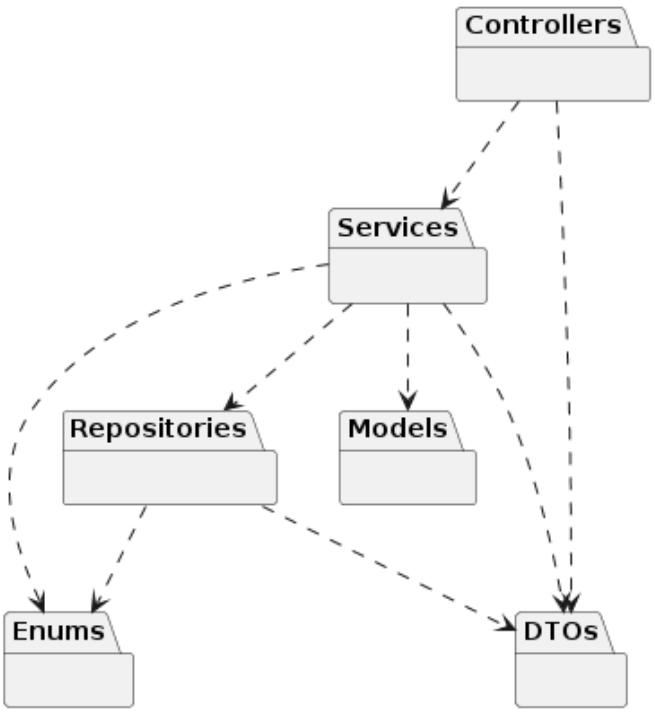

Рисунок 3.5 – Діаграма пакетів система

Основні елементи діаграми:

– Controllers: контролери відповідають за прийом запитів від користувачів, їх обробку та відправку відповідних відповідей. Вони діють як посередники між користувацьким інтерфейсом і логікою обробки даних, делегуючи більш специфічну обробку даних службам;

– Services: сервіси містять бізнес-логіку застосунку та використовуються для обробки бізнес-операцій, правил і обчислень. Вони викликаються контролерами та можуть взаємодіяти з репозиторіями для отримання або зміни даних;

– Repositories: репозиторії відповідають за пряму взаємодію з базою даних. Вони ізолюють логіку доступу до даних від бізнес-логіки, що дозволяє зменшити залежності та спростити тестування компонентів;

– Models: моделі визначають структуру даних у системі. Вони використовуються для відображення даних у базі, а також передачі даних між різними частинами архітектури;

– DTOs: DTOs використовуються для передачі даних між процесами, зокрема між backend-ом і frontend-ом. Вони допомагають забезпечити безпеку даних, обмежуючи об'єм переданої інформації до того, що дійсно потрібно для конкретної операції;

– Enums: використовуються для визначення наборів констант або обмежених значень, які можуть використовуватися у всьому застосунку, що додає читабельності та знижує ймовірність помилок.

Діаграма пакетів наглядно показує структуру програми, зорієнтовану на легке масштабування та поділ застосунку на логічно відокремлені шари, що спрощує розробку, супровід та оновлення програмного забезпечення.

3.5 ER-діаграма бази даних

ER-діаграма (Entity-Relationship diagram) – це візуальне представлення структури бази даних. Вона використовується для ілюстрації зв'язків між різними сутностями в базі даних. Кожна сутність у діаграмі представлена у формі прямокутника, який включає назву сутності та її атрибути, а зв'язки між сутностями зображаються лініями, що можуть мати різні символи на кінцях для позначення типу зв'язку (один до одного, один до багатьох, багато до багатьох). ER-діаграма допомагає аналітикам та розробникам розуміти логіку бази даних і спрощує процес її проєктування і розвитку.

На представленій ER-діаграмі відображені різні таблиці та їхні зв'язки, які є складовими бази даних вебзастосунку для реєстрації подій (див. рис. 3.6).

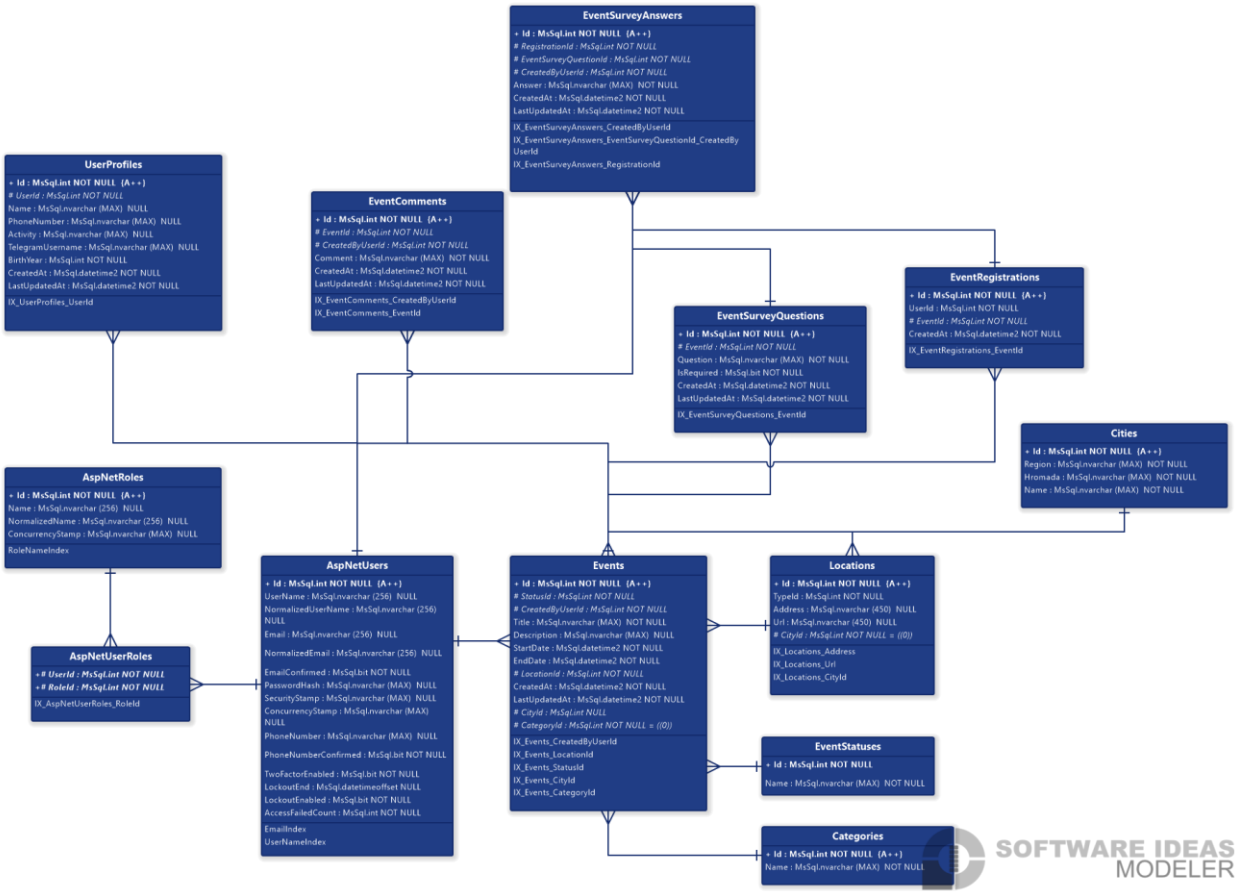

Рисунок 3.6 – ER-діаграма

Діаграма включає наступні основні компоненти:

UserProfiles: Ця таблиця зберігає особисті профілі користувачів, включаючи інформацію, таку як ID користувача, ім'я, номер телефону, дату створення профілю тощо.

AspNetUsers: Ця таблиця управляє реєстрацією користувачів та їхніми обліковими даними, зокрема, логіном, паролем та електронною адресою. Вона інтегрована з системою ідентифікації ASP.NET.

AspNetRoles: Управління ролями користувачів в системі, вказуючи на специфічні ролі, які можуть бути призначені користувачам.

Events: Таблиця для управління подіями, яка містить інформацію про кожну подію, таку як ID події, заголовок, опис, стан події, час початку та завершення.

Locations: Управління локаціями, де можуть відбуватися події. Включає адресу, назву місця та зв'язок із таблицею міст.

Cities: Інформація про міста, включаючи назву міста та регіон, забезпечуючи можливість локалізації подій.

EventComments: Коментарі, залишені користувачами до подій, що містять відомості про ID події та текст коментаря.

EventRegistrations: Управління реєстрацією користувачів на події, зберігаючи зв'язок між користувачами та подіями.

EventSurveyQuestions та EventSurveyAnswers: Управління опитуваннями та відповідями на них для подій, дозволяючи збирати зворотний зв'язок від учасників.

Categories: Категорії, до яких можуть належати події, наприклад, музика, освіта тощо.

EventStatuses: Перелік можливих статусів подій, як-от «Планується», «Відбувається», «Завершено».

На представленій ER-діаграмі визначені наступні зв'язки між таблицями бази даних:

UserProfiles і AspNetUsers:

Зв'язок між цими таблицями відображає відношення між особистими даними користувачів та їхніми аутентифікаційними даними. Ймовірно, кожен профіль користувача прив'язаний до облікового запису в системі аутентифікації.

AspNetUsers і AspNetRoles:

Зв'язок між користувачами та їхніми ролями дозволяє визначати рівні доступу та дозволи в системі, використовуючи рольову модель доступу.

Events і Locations:

Події прив'язані до локацій, що вказує на місце проведення кожної події. Цей зв'язок дозволяє управляти даними про місцезнаходження подій.

Locations і Cities:

Кожна локація асоційована з певним містом, що дозволяє деталізувати інформацію про місцеположення подій у межах певних географічних регіонів.

Events і Categories:

Події категоризовані для кращої організації та пошуку в системі. Кожна подія має одну або декілька категорій, які визначають її тематику.

EventRegistrations і Events:

Цей зв'язок відображає реєстрації користувачів на події. Кожен запис у таблиці реєстрацій асоційований з конкретною подією.

EventRegistrations і AspNetUsers:

Кожна реєстрація пов'язана з певним користувачем, що дозволяє відстежувати, які події вибирають користувачі.

EventComments і Events:

Коментарі, залишені користувачами до подій, прив'язані безпосередньо до цих подій, що дозволяє управляти зворотнім зв'язком від користувачів.

EventComments і AspNetUsers:

Коментарі пов'язані з користувачами, які їх залишили, забезпечуючи відстеження авторства коментарів.

EventSurveyQuestions і Events:

Опитування асоційовані з конкретними подіями, дозволяючи збирати специфічну інформацію для кожної події.

EventSurveyAnswers і EventSurveyQuestions:

Відповіді на питання опитувань прив'язані до конкретних питань, що дозволяє аналізувати зворотний зв'язок користувачів у контексті поставлених питань.

Ці зв'язки дозволяють забезпечити комплексне управління даними в системі, оптимізуючи взаємодію між різними компонентами вебзастосунку.

Під час створення бази даних використовувалась нормалізація, процес організації даних у базі даних. Цей процес включає розділення великих

таблиць на менші (менше зв'язані) таблиці та визначення зв'язків між ними, щоб зменшити дублювання даних та покращити інтегрітет даних [13].

Нормалізація часто використовується для оптимізації структури бази даних, роблячи її більш організованою та ефективною для зберігання, обробки запитів та забезпечення цілісності даних [13].

Основні форми нормалізації:

Перша нормальна форма (1NF): Вимагає, щоб усі значення в таблиці були атомарними (кожен стовпчик містить лише одне значення для кожного рядка) [13].

Друга нормальна форма (2NF): Знаходиться у 1NF і всі атрибути, що не є ключовими, повинні залежати від усього первинного ключа [13].

Третя нормальна форма (3NF): Знаходиться у 2NF і всі його атрибути, що не є ключовими, не мають транзитивних залежностей (залежностей від інших неключових атрибутів) [13].

Приклади з діаграми:

AspNetUsers і AspNetRoles:

1NF: Кожен стовпець (Id, UserName, Email та ін.) містить лише одне значення для кожного рядка. Це демонструє дотримання першої нормальної форми.

2NF та 3NF: Ролі користувачів виокремлені у власну таблицю AspNetRoles з унікальним Id, що забезпечує відсутність часткових залежностей та транзитивних залежностей в таблиці AspNetUsers, демонструючи дотримання другої та третьої нормальної форми.

Events, Locations, Cities:

Розділення локацій та міст: Локації та міста мають власні таблиці, з'єднані через зовнішні ключі (CityId у таблиці Locations та Id у таблиці Cities). Це зменшує дублювання інформації про міста в таблиці локацій і вказує на 3NF, де кожен атрибут залежить лише від первинного ключа, а не від інших атрибутів.

EventRegistrations і Events:

Залежність від первинного ключа: У таблиці EventRegistrations використовується зовнішній ключ EventId, що зв'язує реєстрації з конкретними подіями в таблиці Events. Це приклад 2NF, де атрибути залежать від усього первинного ключа (в даному випадку, EventId є частиною первинного ключа реєстрації).

Ці приклади демонструють, як нормалізація застосовується для організації та збереження даних у базі даних, мінімізуючи повторення та покращуючи ефективність управління даними.

3.6 Використані технології та мови програмування

React

React – це бібліотека JavaScript, розроблена Facebook, для створення інтерактивних та динамічних користувацьких інтерфейсів, особливо для односторінкових застосунків [10].

React використовує декларативний підхід до програмування, що спрощує процес проєктування комплексних інтерфейсів за допомогою компонентного підходу. Він також підтримує віртуальний DOM, що оптимізує оновлення інтерфейсу, підвищуючи продуктивність [10].

Redux

Redux – це відкритий інструмент управління станом для JavaScript-застосунків, часто використовуваний разом з React [11].

Redux дозволяє централізувати управління станом застосунків, що спрощує роботу з даними на великих масштабах і забезпечує легкість у відстеженні змін стану, полегшуючи тестування і налагодження [11].

Tailwind CSS

Tailwind – це утилітно-орієнтований CSS фреймворк, який надає розробникам гнучкість стилізації за допомогою класів, які можна застосовувати безпосередньо у HTML [14].

Tailwind мінімізує потребу в написанні CSS, дозволяючи розробникам швидше створювати інтерфейси з високою адаптивністю та налаштуванням, використовуючи утиліти без необхідності виходу з HTML [14]

ASP.NET Core

ASP.NET Core – це високопродуктивний фреймворк від Microsoft для створення сучасних інтернет-застосунків та API [12].

Фреймворк підтримує розробку на мультиплатформеному рівні, має вбудовану підтримку залежностей (dependency injection) і можливості для масштабування, що робить його ідеальним для створення застосунків [12].

Цей фреймворк від Microsoft спроєктований таким чином, щоб надати розробникам повний контроль над їхніми вебзастосунками, при цьому забезпечуючи високу продуктивність, гнучкість у виборі технологій та простоту в обслуговуванні[15].

ASP.NET Core пропонує багато функцій, таких як вбудована підтримка залежностей (dependency injection), легке тестування, вбудовані інструменти для безпеки та можливість працювати на багатьох платформах, включаючи Windows, Linux та macOS [15].

Swagger (OpenAPI)

Swagger або OpenAPI – це інструмент для опису, створення документації і взаємодії з RESTful вебсервісами [16].

Swagger дозволяє автоматично генерувати документацію для API, забезпечуючи точність і актуальність документації, а також надає інтерактивний інтерфейс для тестування API [16].

Bearer token у HttpOnly cookies

Використання Bearer токенів у httponly cookies забезпечує безпечне зберігання та передачу аутентифікаційних даних між клієнтом і сервером.

Httponly cookies не доступні через JavaScript на клієнтській стороні, що зменшує ризик XSS атак, тоді як Bearer токени забезпечують простий, але ефективний спосіб управління сеансами користувачів.

Dependency Injection

Dependency Injection (DI) – це техніка програмування, що використовується для зменшення жорсткої залежності між компонентами програмного забезпечення.

DI дозволяє більшу гнучкість і легшу тестову підтримку, а також сприяє слабкому зв'язку між компонентами, підвищуючи повторне використання коду та легкість обслуговування.

EntityFramework

Entity Framework (EF) є об'єктно-реляційним відображувачем (ORM), розробленим Microsoft для .NET платформи. Він дозволяє розробникам працювати з базою даних за допомогою вищого рівня абстракції, представляючи базу даних як колекцію об'єктів і властивостей, замість традиційних SQL-запитів.

Entity Framework зменшує кількість написання коду для доступу до даних, оскільки автоматизує більшість завдань по створенню коду доступу до даних і обробки транзакцій. Він також підтримує різні підходи до моделювання даних, включаючи «Code First», «Database First» та «Model First», що надає гнучкість у виборі підходу в залежності від специфіки проєкту [17].

Однією з ключових переваг EF є його здатність автоматично синхронізувати структуру класів .NET зі схемою бази даних, що значно спрощує процеси розробки та супроводу. Entity Framework включає потужний запитувальний API, LINQ (Language Integrated Query), який дозволяє розробникам виконувати запити до бази даних високого рівня абстракції. LINQ інтегрується безпосередньо з C# або іншими мовами .NET, що дозволяє писати запити, що забезпечують типову безпеку та компіляційну перевірку.

Entity Framework також оптимізує взаємодію з базою даних, використовуючи стратегії кешування та ліниве завантаження для підвищення продуктивності застосунків [17].

C#

C# – це мова програмування від Microsoft, створена як частина .NET фреймворку, яка використовується для розробки широкого спектру застосунків [18].

C# є об'єктно-орієнтованою мовою, що підтримує строгу типізацію, автоматичне управління пам'яттю та багато інших сучасних функцій програмування, забезпечуючи швидкість, безпеку та продуктивність [18].

T-SQL

T-SQL (Transact-SQL) – це розширення SQL, що використовується в Microsoft SQL Server для включення бізнес-логіки в запити до бази даних [19]

T-SQL дозволяє виконувати складні запити, забезпечуючи контроль над транзакціями, обробку помилок та інші програмні можливості, які покращують безпеку та ефективність взаємодії з даними [19].

Висновки до розділу 3

У третьому розділі кваліфікаційної роботи бакалавра представлено комплексний підхід до проєктування та конструювання вебзастосунку для реєстрації подій, що об'єднує як фронтенд, так і бекенд компоненти системи.

Фронтенд реалізовано за допомогою бібліотеки React, яка забезпечує динамічну взаємодію з користувачем та оптимізує процеси рендерингу інтерфейсу. Redux використано для управління станом додатка, що дозволяє ефективно управляти даними і станом UI на клієнтській стороні. Це забезпечує швидку відповідь застосунку на дії користувача і підвищує зручність взаємодії.

Бекенд розроблено з використанням ASP.NET Core, що надає потужні інструменти для створення RESTful API. Серверна частина забезпечує обробку бізнес-логіки, взаємодію з базою даних і виконання функцій захисту даних. Використання Entity Framework для взаємодії з базою даних спрощує реалізацію CRUD операцій і підвищує ефективність обробки запитів.

4 РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ ЗАСТОСУНКУ

4.1 Огляд back-end сервісів застосунку

Сервіс для керування подіями, включаючи створення, видалення, отримання інформації про події та їх оновлення.

Таблиця 4.1 – EventService

Функція	Опис
CreateEventAsync	Створює нову подію з перевіркою авторизації та додаванням місця проведення.
DeleteEventAsync	Видаляє подію з бази даних після перевірки на наявність та авторизації. Включає в себе видалення залежних даних.
GetEventsAsync	Отримує список подій, створених або з участю користувача.
SearchEventsAsync	Пошук подій за ключовими словами, категоріями, містами.
GetEventByIdAsync	Повертає деталі події за ID.
UpdateEventAsync	Оновлює інформацію про подію з перевіркою авторизації та наявності події.

Сервіс EventCommentService управляє коментарями до подій, забезпечуючи можливість додавання та видалення коментарів.

Таблиця 4.2 – EventCommentService

Функція	Опис
AddNewCommentAsync	Додає новий коментар до події після перевірки існування події та авторизації користувача.
DeleteCommentAsync	Видаляє коментар, перевіряючи наявність

Кінець таблиці 4.2

Функція	Опис
	коментаря та авторизацію користувача або адміністративні права для видалення.

Розглянемо сервіс для управління відповідями на анкетні питання подій.

Таблиця 4.3 – EventSurveyAnswerService

Функція	Опис
AddAnswersAsync	Додає відповіді на питання події після перевірки наявності всіх обов'язкових питань.

Сервіс для реєстрації та відписки користувачів від подій - RegistrationService.

Таблиця 4.4 – RegistrationService

Функція	Опис
Register	Реєструє користувача на подію, перевіряючи наявність події та повноту профілю користувача.
Unregister	Відміняє реєстрацію користувача на подію з відповідними перевірками.

Нижче наведено функції сервісу для управління профілями користувачів, включаючи отримання, оновлення та видалення профілів.

Таблиця 4.5 – UserService

Функція	Опис
DeleteUserAndProfile	Видаляє користувача та його профіль після відповідних перевірок.
GetUserProfile	Повертає профіль користувача за ID.

Кінець таблиці 4.5

Функція	Опис
UpdateUserProfile	Оновлює профіль користувача після перевірки авторизації.

4.2 Створення checklist для тестування

Для забезпечення ретельного тестування основного функціоналу створено наступні чеклисти, які охоплюють різні сценарії та перевірки [20].

Таблиця 4.6 – Чеклист створення події

ID	Опис кейсу	Кроки виконання	Очікуваний результат
1	Створення події з усіма обов'язковими полями	1) авторизуйтесь у системі; 2) введіть усі обов'язкові дані; 3) натисніть кнопку «Створити подію».	Подія створюється успішно, користувач отримує підтвердження
2	Створення події без обов'язкових полів	1) авторизуйтесь у систем; 2) пропустіть деякі обов'язкові поля; 3) натисніть «Створити подію».	Виникає помилка з інформацією про необхідність заповнення усіх обов'язкових полів
3	Створення події незареєстрованим користувачем	1) спробуйте створити подію без авторизації.	Виникає помилка доступу з проханням авторизуватися

Продовження таблиці 4.6

ID	Опис кейсу	Кроки виконання	Очікуваний результат
4	Введення некоректного формату дати	1) авторизуйтесь у системі; 2) введіть некоректну дату; 3) натисніть «Створити подію».	Виникає помилка з вимогою введення дати у коректному форматі
5	Введення дуже великого тексту в опис події	1) авторизуйтесь у системі; 2) введіть надмірно великий текст у поле опису; 3) натисніть "Створити подію".	Виникає помилка про перевищення допустимої кількості символів
6	Створення події із зазначенням існуючої локації	1) авторизуйтесь у системі; 2) введіть дані події, включаючи адресу існуючої локації; 3) натисніть «Створити подію».	Подія використовує існуючу локацію, користувач отримує підтвердження створення.
7	Створення події в майбутньому на 5 років	1) авторизуйтесь у системі; 2) встановіть дату події на 5 років вперед; 3) натисніть «Створити подію».	Подія успішно створюється на вказану дату, отримується підтвердження
8	Відміна створення події під час вводу даних	1) авторизуйтесь у системі; 2) розпочніть вводити дані; 3) натисніть «Відміна».	Всі дані очищаються, і користувач

Кінець таблиці 4.6

			залишає форму без створення події
9	Створення події з некоректним URL локації	1) авторизуйтесь у системі; 2) введіть некоректний URL; 3) натисніть «Створити подію».	Виникає помилка з вимогою введення коректного URL
10	Вибір неіснуючої категорії для події	1) авторизуйтесь у системі; 2) виберіть неіснуючу категорію; 3) натисніть «Створити подію».	Виникає помилка про вибір неіснуючої категорії
11	Створення події з некоректними символами у назві	1) авторизуйтесь у системі; 2) введіть назву з спецсимволами; 3) натисніть «Створити події».	Виникає помилка про некоректні символи у назві

Таблиця 4.7 – Чеклист реєстрації на подію з обов’язковим опитуванням

ID	**Опис кейсу**	**Кроки виконання**	**Очікуваний результат**
1	Реєстрація на подію з відповідями на всі обов'язкові питання	1) авторизуйтесь у системі; 2) оберіть подію для реєстрації; 3) відповісти на всі обов'язкові питання; 4) натисніть	Реєстрація на подію успішна, користувач отримує підтвердження

Продовження таблиці 4.7

ID	Опис кейсу	Кроки виконання	Очікуваний результат
		«Зареєструватися».	
2	Реєстрація на подію без відповідей на обов'язкові питання	1) авторизуйтесь у системі; 2) оберіть подію; 3) пропустіть деякі обов'язкові питання; 4) натисніть «Зареєструватися».	Виникає помилка з вимогою відповісти на всі обов'язкові питання
3	Реєстрація на подію незареєстрованим користувачем	1) спробуйте зареєструватися на подію без авторизації.	Виникає помилка доступу з проханням авторизуватися
4	Введення некоректних даних у відповіді	1) авторизуйтесь у системі; 2) оберіть подію. 3) введіть некоректні дані у відповіді на питання; 4) натисніть «Зареєструватися».	Виникає помилка з вимогою введення коректних даних
5	Реєстрація на подію з відповідями на необов'язкові питання	1) авторизуйтесь у системі; 2) оберіть подію; 3) відповісти тільки на необов'язкові питання; 4) натисніть «Зареєструватися».	Реєстрація на подію успішна, навіть якщо не відповідали на обов'язкові питання
6	Реєстрація на подію, яка вже повністю	1) авторизуйтесь у системі; 2) оберіть подію, яка вже не	Виникає повідомлення,

Кінець таблиці 4.7

ID	Опис кейсу	Кроки виконання	Очікуваний результат
	заповнена	має вільних місць; 3) спробуйте зареєструватися; 4) натисніть «Зареєструватися».	що місць немає та реєстрація неможлива
7	Реєстрація на подію з надвеликим текстом у відповідях	1) авторизуйтесь у системі; 2) оберіть подію; 3) введіть у відповіді на питання дуже довгий текст; 4) натисніть «Зареєструватися».	Перевірка на максимальну довжину тексту відповіді, виведення помилки якщо текст завеликий
8	Спроба реєстрації після закінчення терміну реєстрації	1) авторизуйтесь у системі. 2) оберіть подію, термін реєстрації на яку вже закінчився; 4) натисніть «Зареєструватися».	Виникає помилка про закінчення терміну реєстрації на подію

4.3 Демонстрація роботи розробленого ПЗ

Скористаємось застосунком для реєстрації на подію. Наведені далі скриншоти та їх опис актуальні для незареєстрованого користувача, який має спершу зареєструватись та заповнити профіль, після чого він успішно буде зареєстрованим на обрану подію, яка не має попереднього опитування.

У випадку якщо користувач зареєстрований, але його профіль не заповнено, користувачу буде запропоновано заповнити профіль спершу.

Найпростіший сценарій реєстрації на подію відбуватиметься, якщо користувач авторизований та має заповнений профіль, у такому випадку реєстрація зводитися до одного кліку на кнопку «Зареєструватись» у обраній користувачем події.

Розпочнемо з головної сторінки (див. рис. 4.1).

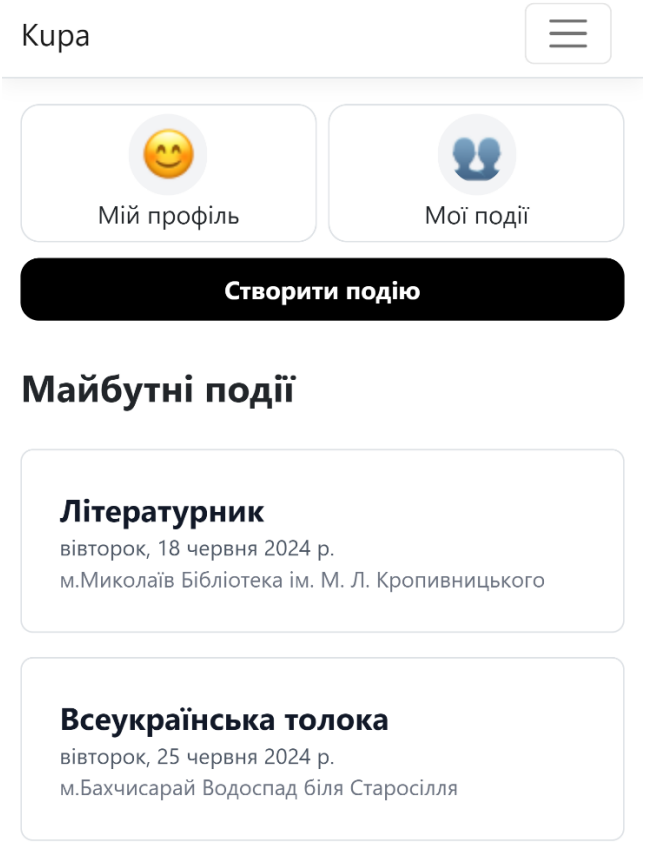

Рисунок 4.1 – Головна сторінка застосунку

Стартова сторінка додатку, яка показує опції для користувача («Мій профіль», «Мої події»), а також кнопку для створення події. Нижче видно список майбутніх подій, зокрема «Літературник» та «Всеукраїнська толока», які відсортовані за датою початку події. З таких блоків подій можна дізнатись основну інформацію про подію, яка може зацікавити учасників: назва, дата проведення, та місце.

Детальніше про подію можна дізнатись перейшовши на сторінку події (див. рис. 4.2), натиснувши на неї зі списку.

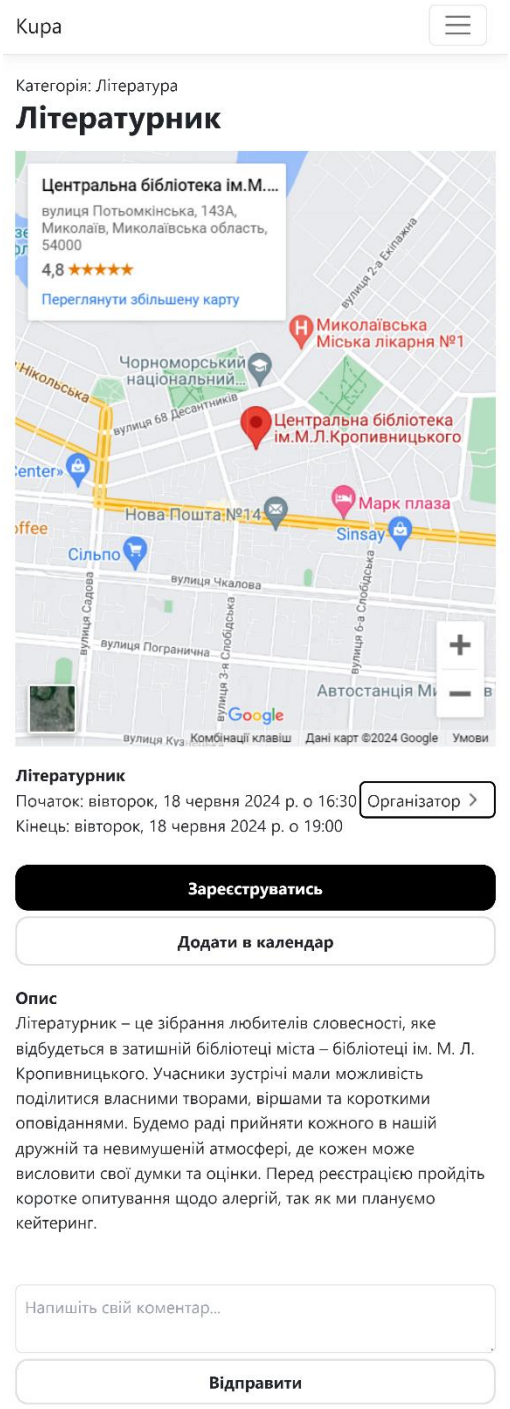

Рисунок 4.2 – Сторінка події «Літературник»

Сторінка з інформацією про подію «Літературник», включає опис, місце проведення на мапі, час, інформацію про організатора та можливість реєстрації. Також є кнопки для додавання події у календар. Так як користувач незареєстрований пропонуємо йому це (див. рис. 4.3).

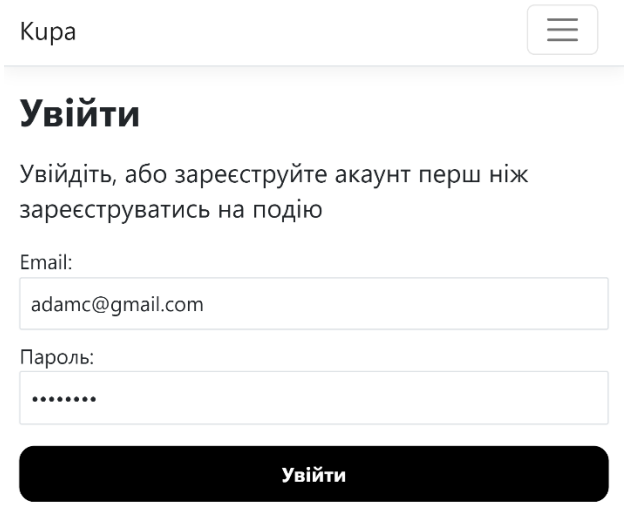

Рисунок 4.3 – Сторінка авторизації

Користувачу пропонується ввести email і пароль для входу в систему. Після цього лишається лише заповнити профіль (див. рис. 4.4), інформація з якого братиметься для потреб організаторів.

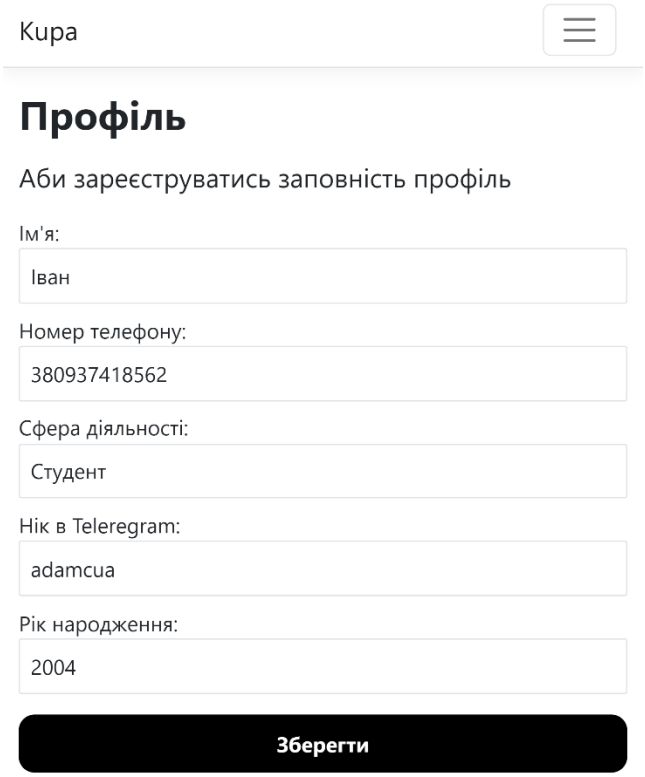

Рисунок 4.4 – Сторінка налаштування профілю користувача

Проста форма для заповнення детальної інформації профілю нового користувача після реєстрації, містить в собі ім'я, номер телефону, сферу діяльності, ім’я користувача у месенджері та дату народження. Зберігаємо інформацію в базі даних та переадресовуємо користувача до сторінки події, де його чекає повідомлення про успіх (див. рис. 4.5).

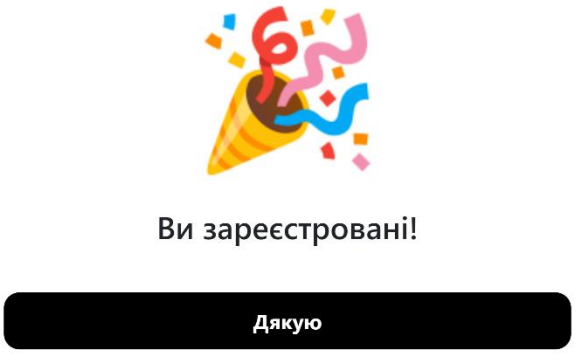

Рисунок 4.5 – Повідомлення про успішну реєстрацію

Сторінка з повідомленням про успішну реєстрацію на подію, що містить вітальне зображення і кнопку «Дякую», що підтверджує завершення процесу.

4.4 Впровадження системи

Розроблена версія програмного забезпечення для молодіжної волонтерської організації стане значним кроком у покращенні процесу реєстрації на події.

Етапи впровадження:

1) підготовка та налаштування: на цьому етапі здійснюється підготовка інфраструктури організації під вимоги нової системи. Це включає в себе створення хмарного серверного програмного забезпечення, налаштування баз даних та конфігурацію мережевих налаштувань між front-end та back-end;

2) тестування: перед повноцінним запуском програмне забезпечення пройде ретельне тестування, щоб забезпечити його стабільність та надійність.

Тестування включатиме як функціональні, так і нефункціональні аспекти системи;

3) навчання користувачів: ключові користувачі та волонтери пройдуть навчання щодо використання нового програмного забезпечення, що забезпечить гладкий перехід від старих процесів до нової системи;

4) запуск: після завершення підготовчих заходів та навчання, система буде офіційно запущена та стане доступною для використання всіма членами організації;

5) моніторинг та оптимізація: впровадження системи супроводитиметься постійним моніторингом її ефективності та внесенням необхідних змін для оптимізації процесів.

На даному етапі система на етапі узгодження між членами організації, але теперішня версія вже відзначилась наступними результатами:

– покращена ефективність: завдяки автоматизації процесу реєстрації знижено часові затрати на обробку заявок та підвищено загальну продуктивність роботи організації;

– збільшена точність даних: мінімізація людських помилок при введенні даних завдяки використанню форм із заповненими полями;

– покращений користувацький досвід: інтерфейс системи став інтуїтивно зрозумілим і зручним для користувачів, що сприяло залученню більшої кількості учасників до подій організації.

Висновки до розділу 4

У цьому розділі кваліфікаційної роботи бакалавра розглядалися кроки реалізації, тестування та впровадження вебзастосунку для реєстрації на події. Реалізація включала детальний опис back-end сервісів, які управляють всіма аспектами подій, коментарями, анкетними питаннями та реєстрацією учасників. Основна увага приділялася модулям, які обробляють створення,

оновлення, видалення та пошук подій, а також управління коментарями та анкетними відповідями.

Тестування програмного забезпечення відіграло ключову роль у забезпеченні його надійності та ефективності. Було розроблено чеклисти для систематичного тестування кожної функціональності, зокрема перевірка правильності реалізації бізнес-логіки і взаємодій із базою даних.

Впровадження розробленого програмного забезпечення стане важливим кроком для організації, де воно почне покращувати процес реєстрації на події. Воно забезпечить поліпшення в обробці даних та надасть користувачам кращий і більш інтуїтивно зрозумілий інтерфейс.

ВИСНОВКИ

Виконання кваліфікаційної роботи бакалавра зосередилось на розробці вебзастосунку для реєстрації подій, що відповідає сучасним потребам громадської активності та волонтерства. Для досягнення мети вирішено наступні завдання: проведено аналіз наявних систем реєстрації на заходи та визначено їхні недоліки; обрано технологічний стек для розробки вебзастосунку; розроблено мінімально життєздатний продукт концепції «одноразової реєстрації» для спрощення процесу участі в заходах; створено простий і легко запам’ятовуваний дизайн інтерфейсу; реалізовано необхідний функціонал для зберігання та обробки даних користувачів; протестовано вебзастосунок на відповідність бізнес-вимогам користувачів та безпеки даних.

В умовах зростаючої потреби в ефективних інструментах для організації заходів, особливо на фоні війни росії проти України, розроблений вебзастосунок вносить значний вклад у підтримку волонтерської діяльності та громадської участі.

Проєкт вирішує проблему повторного введення даних при реєстрації на події, пропонуючи зрозумілий та простий механізм «одноразової реєстрації». Це сприяє збільшенню участі в заходах, зменшує часові витрати на реєстрацію та покращує користувацький досвід.

Аналіз існуючих систем та визначення потреб користувачів дозволили ідентифікувати ключові напрямки для покращення вебзастосунку. Враховано вимоги до інтуїтивності інтерфейсу та інтеграції з календарем, що забезпечує високий рівень зручності для користувачів. Детальне проєктування архітектури системи з діаграмами класів, компонентів та пакетів забезпечує чітке розуміння структури та зв'язків компонентів. Це сприяє ефективному розподілу ресурсів і легкості супроводження. Ретельне тестування вебзастосунку підтвердило його надійність і відповідність бізнес-вимогам. Реалізація проєкту сприяє збільшенню громадської участі, що є важливим в

контексті підтримки демократичних цінностей і зміцнення громадянського суспільства в Україні.

Завершуючи, можна констатувати, що мету кваліфікаційної роботи досягнуто, розроблений вебзастосунок відповідає актуальним потребам і стане значним кроком у напрямку технологічної підтримки волонтерської діяльності та громадських ініціатив, забезпечуючи потенціал для подальшого розвитку та вдосконалення.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. SoftServe. SoftServe Helps Launch National Volunteer Platform in Ukraine - News | SoftServe. SoftServe | Software Development & Digital Services Company. URL: https://www.softserveinc.com/uk-ua/news/softserve-helps-launch-national-volunteer-platform (date of access: 28.01.2024).

2. Top 10 Must Have Features Of Event Management Software - Eventleaf. Event Management Software and Mobile Event Apps - Eventleaf. URL: https://www.eventleaf.com/event-app/ (date of access: 28.01.2024).

3. Календар DOU. DOU. URL: https://dou.ua/calendar/ (дата звернення: 28.01.2024).

4. Product: Event Management Software. Bizzabo. URL: https://www.bizzabo.com/event-management-software (date of access: 28.01.2024).

5. Горбачов Є. І., Константинова Л. В. Огляд SQL Server Management Studio для роботи з БД. Інформаційна безпека та комп'ютерні технології, м. Кропивницький. Кропивницький, 2023. С. 60–61. URL: https://www.kntu.kr.ua/doc/science/zahody/vikl/2023/6-tez.pdf#page=60.

6. Sundaramoorthy S. UML Diagramming: A Case Study Approach. Auerbach Publications, 2022. 416 p.

7. Тарасенко В. В. Сучасні тренди та тенденції розвитку веб-дизайну. *Proceedings of III International Scientific and Practical Conference*, м. Львів, 14 черв. 2022 р. 2022. С. 1236–1239.

8. Mobile First Design: What it is & How to implement it | BrowserStack. BrowserStack. URL: https://www.browserstack.com/guide/how-to-implement-mobile-first-design (date of access: 02.05.2024).

9. RESTful API Testing Methodologies: Rationale, Challenges, and Solution Directions / A. Ehsan et al. *Applied Sciences*. 2022. Vol. 12, no. 9.4369. 16 p. URL: https://doi.org/10.3390/app12094369 (date of access: 28.03.2024).

10. React Reference Overview. *React*. URL: https://react.dev/reference/react (date of access: 28.03.2024).

11. A JS library for predictable and maintainable global state management. *Redux*. URL: https://redux.js.org/ (date of access: 02.06.2024).

12. ASP.NET documentation. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-8.0 (date of access: 28.03.2024).

13. Єрмолаєв О. Д., Суліма С. В. Оптимізація виконання запитів у реляційній системі управління базами даних. Збірник матеріалів Міжнародної науково-технічної конференції «ПЕРСПЕКТИВИ ТЕЛЕКОМУНІКАЦІЙ», м. Київ, 8–21 квіт. 2023 р. Київ, 2023. URL: http://conferenc.its.kpi.ua/2023/paper/view/27548/15731.

14. Documentation. *Tailwind CSS - Rapidly build modern websites without ever leaving your HTML*. URL: https://v2.tailwindcss.com/docs (date of access: 28.03.2024).

15. Lock A. ASP. NET Core in Action, Third Edition. Manning Publications Co. LLC, 2023. 984 p.

16. Documentation. *Swagger*. URL: https://swagger.io/docs/ (date of access: 02.06.2024).

17. Entity Framework documentation hub. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/ef/ (date of access: 28.03.2024).

18. C# docs - get started, tutorials, reference. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/uk-ua/dotnet/csharp/ (date of access: 28.03.2024).

19. Transact-SQL reference (Database Engine) - SQL Server. *Microsoft Learn: Build skills that open doors in your career.* URL: https://learn.microsoft.com/en-us/sql/t-sql/language-reference?view=sql-server-ver16 (date of access: 28.03.2024).

20. Parsa S. Software Testing Automation. Cham : Springer International Publishing, 2023. URL: https://doi.org/10.1007/978-3-031-22057-9 (date of access: 28.03.2024).