

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Комп'ютерних наук і технологій
(повне найменування факультету)

Комп'ютерні системи та мережі
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалаврський
(ступінь вищої освіти)

на тему: РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ ОРГАНІЗАЦІЇ ПОДІЙ ТА
ВЗАЄМОДІЇ КОРИСТУВАЧІВ “EVENTIFY APP”

Виконав(ла): студент(ка) 4 курсу,
групи КНТ-512сп

Спеціальності
123 Комп'ютерна інженерія
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерна інженерія
(назва освітньої програми (спеціалізації))

ІВАНЧЕНКО Д.Р.
(ПРИЗВИЩЕ та ініціали)

Керівник КУДЕРМЕТОВ Р.К.
(ПРИЗВИЩЕ та ініціали)

Рецензент МАЛИЙ О.Ю.
(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій
Кафедра Комп'ютерних систем та мереж
Ступінь вищої освіти бакалаврський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри КУДЕРМЕТОВ Р.К.

«14» квітня 2025 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ІВАНЧЕНКО Данііла Романовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка вебзастосунку для організації подій та взаємодії користувачів “Eventify App”

керівник проєкту (роботи) к.т.н., доцент, КУДЕРМЕТОВ Р.К.

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від «08» квітня 2025 року № 151

2. Строк подання студентом проєкту (роботи) 01.06.2025 р.

3. Вихідні дані до проєкту (роботи) опис предметної області, технології розробки клієнтської та серверної частини

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Аналіз предметної області та визначення функцій системи;

2) Моделювання програмного забезпечення;

3) Проєктування вебсервісу;

4) Розробка вебзастосунку.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	КУДЕРМЕТОВ Р.К.		
Нормоконтроль	ЩЕРБАК Н.В.		

7. Дата видачі завдання «14» квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз предметної області	до 18.04.2025	
2	Визначення та аналіз вимог до програмного забезпечення	до 22.04.2025	
3	Розробка проєктних рішень	до 24.04.2025	
4	Моделювання та конструювання програмного забезпечення	до 01.05.2025	
5	Розробка описів прецедентів	до 05.05.2025	
6	Проектування архітектури програмного забезпечення	до 12.05.2025	
7	Проектування інтерфейсу користувача	до 22.05.2025	
8	Оформлення пояснювальної записки	до 25.05.2025	
9	Проходження нормоконтролю	до 01.06.2025	
10	Перевірка на наявність академічного плагіату	до 03.06.2025	
11	Проходження рецензування	до 10.06.2025	

Студент(ка)

(підпис)

Даніїл ІВАНЧЕНКО

(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис)

Равіль КУДЕРМЕТОВ

(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
86 с., 3 табл., 17 рис., 14 джерел.

WEB-ДОДАТОК, БАЗА ДАНИХ, ІНФОРМАЦІЙНА СИСТЕМА, ОРГАНІЗАЦІЯ СВЯТ, ПОДІЯ, ПРЕЦЕДЕНТ

Об'єкт розробки: інформаційна система для управління подіями та забезпечення користувацької взаємодії в рамках вебзастосунку.

Мета кваліфікаційної роботи: розробка вебзастосунку "Eventify App", який надає користувачам комплексну платформу для організації подій та ефективної взаємодії між учасниками.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, проведено структурно-функціональне моделювання, визначено варіанти використання web-додатку та спроектовано базу даних.

У третьому розділі описано розробку web-додатку, яка є результатом проектування. Наведено архітектуру web-додатку.

У четвертому розділі обрано вибір платформи для розробки, виконано проектування і розробку програми, наведено опис алгоритму і структури функціонування застосунку, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів.

ABSTRACT

Explanatory Note to the Bachelor's Qualification Thesis: 86 pages, 3 tables, 17 figures, 14 sources.

WEB APPLICATION, DATABASE, INFORMATION SYSTEM, EVENT ORGANIZATION, EVENT, USE CASE

Object of Development: An information system for event management and ensuring user interaction within a web application.

Objective of the Qualification Thesis: Development of the web application "Eventify App", which provides users with a comprehensive platform for organizing events and effective interaction among participants.

The introduction analyzes the current state of the issue, specifies the objective of the qualification thesis and its application field, provides the rationale for the relevance of the topic, and clarifies the problem statement.

The first chapter presents an analysis of the subject area, identifies the relevance of the task and the purpose of the development, formulates the task statement, and sets out the requirements for software implementation, technologies, and software tools.

The second chapter analyzes existing solutions, conducts structural and functional modeling, defines the use cases of the web application, and designs the database.

The third chapter describes the development of the web application as the result of the design process. The architecture of the web application is presented.

The fourth chapter outlines the platform choice for development, the design and implementation of the application, provides a description of the algorithm and functioning structure of the application, defines input and output data, and presents the specifications of the technical parameters used.

ЗМІСТ

Скорочення та умовні позначки.....	7
Вступ.....	8
1 Аналіз предметної області та визначення функцій системи	10
1.1 Огляд предметної області.....	10
1.2 Дослідження аналогів застосунку для організації подій	12
1.3 Специфікація вимог до програмного забезпечення	17
1.4 Постановка задачі.....	24
2 Моделювання програмного забезпечення	26
2.1 Сценарії використання системи.....	27
2.2 Прецедент "Створення події".....	29
2.3 Прецеденти "Перегляд подій", "Видалення події" та "Редагування події".....	32
2.4 Прецеденти "Підписка на подію" та "Перегляд підписаних подій"	36
3 Проєктування вебсервісу.....	39
3.1 Архітектура вебсервісу.....	39
3.2 Проєктування Cloud Firestore.....	43
3.3 Обґрунтування вибору СКБД	50
3.4 Розробка інтерфейсу користувача	53
3.5 Аналіз та вибір засобів розробки.....	59
4 Розробка вебзастосунку.....	62
4.1 Розробка бази даних.....	62
4.2 Налаштування індексів та оптимізація доступу до даних у Firestore	64
4.3 Розробка клієнтської частини	66
4.4 Розробка інтерфейсу вебзастосунку.....	69
4.5 Опис функцій API	74
Висновки	81
Перелік джерел посилання	83
Додаток А Лістинги програм	85

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

- API - application programming interface
- СКБД - система керування базами даних
- UI - user interface
- UID - unique identifier
- URL - uniform resource locator
- UX - user experience

ВСТУП

У сучасному світі цифрових технологій усе більше сфер людської діяльності переходять у віртуальний простір. Особливо це стосується організації подій - від невеликих зустрічей і майстер-класів до масштабних конференцій та фестивалів. Швидкий обмін інформацією, зручна реєстрація, ефективна комунікація між учасниками та організаторами стали ключовими чинниками успішного проведення заходів. У цьому контексті зростає потреба у зручних, функціональних і доступних інструментах, що забезпечують повний цикл організації подій онлайн [1].

Дипломна робота присвячена розробці вебзастосунку “Eventify App”, основним призначенням якого є створення, управління та просування подій, а також забезпечення інтерактивної взаємодії між користувачами. Цей застосунок покликаний стати ефективною платформою як для організаторів, так і для відвідувачів заходів, надаючи інтуїтивно зрозумілий інтерфейс, систему повідомлень, гнучкі налаштування подій та механізми соціальної взаємодії.

Актуальність теми обумовлена зростаючим попитом на інструменти, які дозволяють спростити процес організації подій, підвищити рівень залученості учасників та автоматизувати основні задачі. Більшість існуючих рішень або є надто складними у використанні, або мають обмежений функціонал. Тому створення нового веб-застосунку з акцентом на зручність, масштабованість та інтеграцію сучасних технологій є своєчасним та доцільним. Таким чином, розробка даного веб-застосунку відповідає сучасним вимогам цифрового суспільства та ринку подієвих послуг, а також сприяє покращенню сервісу в галузі івент-менеджменту, що робить тему дипломного дослідження актуальною та практично значущою.

Мета даної роботи - спроектувати та реалізувати вебзастосунок, який би відповідав вимогам користувачів щодо зручності організації подій, взаємодії між учасниками та ефективної модерації контенту.

Об’єктом дослідження є інформаційна система для управління подіями та забезпечення користувацької взаємодії в рамках вебзастосунку.

Предмет дослідження - технології та засоби реалізації функціоналу вебзастосунку для організації подій та взаємодії користувачів.

Для досягнення мети необхідно виконати наступні завдання:

- проаналізувати предметну область і визначити актуальність розробки;
- дослідити наявні аналоги вебсайтів у цій галузі, визначити їхні переваги та недоліки;
- обрати оптимальні веб-технології для реалізації функціоналу;
- спроектувати структуру та модель веб-додатку;
- реалізувати структуру у вигляді повноцінного веб-ресурсу;
- розробити функціонал, що забезпечить повноцінну роботу івент-агентства;
- провести тестування готового продукту.

Практичне значення створеного додатку полягає у спрощенні процесу організації подій та покращенні взаємодії між учасниками, скороченні часу, що витрачається користувачами на організацію події, підвищенні якості сервісу та автоматизації ключових процесів у роботі івент-агентства.

Структура роботи охоплює теоретичну частину, що описує концептуальні основи веб-розробки та огляд аналогічних рішень, а також практичну частину, присвячену реалізації, впровадженню та оцінці ефективності розробленого застосунку “Eventify App”.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИЗНАЧЕННЯ ФУНКЦІЙ СИСТЕМИ

1.1 Огляд предметної області

Організація подій (івент-менеджмент) є багатогранною сферою діяльності, що охоплює планування, координацію, логістику та комунікацію з учасниками заходів. Сучасні події можуть бути різного формату: конференції, семінари, концерти, весілля, корпоративи тощо. Усі вони вимагають ретельного управління ресурсами, часу та персоналом. З огляду на це, виникає потреба в ефективних цифрових інструментах, які автоматизують ці процеси.

У галузі організації подій постійно з'являються нові технологічні інструменти, які спрямовані на вдосконалення комунікації між організаторами та учасниками. Вебзастосунки для реєстрації подій відіграють ключову роль у цьому процесі, надаючи користувачам можливість швидко знаходити заходи, реєструватися на них і взаємодіяти з іншими учасниками. Такі системи включають широкий спектр функціональних можливостей: календарі подій, інтеграцію з платіжними системами, опції для зворотного зв'язку тощо [2].

Центральним елементом розробки є створення інтуїтивно зрозумілого і зручного інтерфейсу, що позитивно впливає на рівень залучення та задоволеності користувачів. Сучасні системи також включають аналітичні інструменти, що дозволяють організаторам відстежувати поведінку користувачів та оптимізовувати процеси.

Швидкий розвиток цифрових технологій стимулює інтеграцію таких вебзастосунків із соціальними мережами, сервісами електронної пошти та іншими онлайн-інструментами. Це відкриває нові можливості для маркетингу та реклами, але водночас вимагає високого рівня захисту персональних даних та відповідності вимогам інформаційної безпеки.

Таким чином, вебплатформи для реєстрації подій знаходяться на перетині технологічного прогресу та змінних потреб споживачів, що змушує розробників не

лише відповідати сучасним вимогам, а й передбачати майбутні тренди задля збереження конкурентоздатності.

З розвитком інформаційних технологій і, зокрема, веб-технологій, на ринку з'явилися численні веб-сервіси та застосунки, які спрощують організацію заходів.

До найпопулярніших функцій таких систем належать:

- створення та адміністрування подій;
- онлайн-реєстрація та продаж квитків;
- управління списками учасників;
- надсилання сповіщень та нагадувань;
- збір зворотного зв'язку після події;
- інтеграція з соціальними мережами.

Серед існуючих рішень можна відзначити такі платформи, як Eventbrite, Meetup, Cvent, Bizzabo та інші. Вони забезпечують базовий функціонал для управління подіями, проте мають низку обмежень: складність у використанні, відсутність гнучкого налаштування під локальні потреби, або платну основу з обмеженим безкоштовним функціоналом.

Крім того, ці сервіси орієнтовані переважно на великі події або корпоративний сегмент. Водночас на ринку спостерігається попит на універсальний, доступний інструмент, який би поєднував у собі простоту використання, можливість організації як публічних, так і приватних заходів, а також розширений набір інструментів для комунікації між учасниками.

Саме в цьому контексті актуальним є створення веб-застосунку “Eventify App”, який би об'єднував ключові переваги наявних рішень, долаючи їхні недоліки. Запропонований застосунок орієнтований на інтуїтивний інтерфейс, гнучке налаштування подій, інтеграцію з популярними сервісами, а також забезпечення ефективної взаємодії користувачів у межах платформи.

1.2 Дослідження аналогів застосунку для організації подій

У процесі вивчення наявних рішень на ринку вебзастосунків, призначених для організації подій, акцент робиться на кількох ключових складових. Ці складові дозволяють краще зрозуміти ринкову ситуацію та виявити можливості для удосконалення власного продукту. До основних аспектів дослідження належать:

- функціональні можливості;
- технологічна основа;
- досвід користувачів та дизайн (UX/UI);
- цільовий сегмент ринку;
- відгуки користувачів та репутація.

Функціональні можливості – здійснюється аналіз типових та унікальних функцій, які пропонуються в аналогічних сервісах. Зокрема, розглядаються можливості з реєстрації учасників, адміністрування подій, створення та їхнє просування, а також специфічні опції, як-от інтеграція з календарями;

Технологічна основа – вивчаються інструменти та технології, що застосовуються конкурентами: мови програмування, хмарні сервіси, архітектурні рішення, а також підходи до реалізації інтерфейсу. Це дозволяє обрати ефективну технічну базу для створення якісного, зручного та стабільного застосунку;

Досвід користувачів та дизайн (UX/UI) – аналізується зовнішній вигляд інтерфейсів, логіка навігації, зручність користування на різних пристроях, а також загальне враження користувача від взаємодії із застосунком [3];

Цільовий сегмент ринку – визначення основної аудиторії існуючих продуктів допомагає зрозуміти, на кого орієнтовані ці рішення, та як адаптувати свій вебзастосунок до потреб певних категорій користувачів [2];

Відгуки користувачів та репутація – аналіз думок та оцінок користувачів дає змогу виявити недоліки в конкурентних продуктах, які можна врахувати для вдосконалення власного рішення.

Такий детальний підхід до аналізу аналогів забезпечує глибоке розуміння сучасних трендів, ринкових викликів і очікувань користувачів. Це, у свою чергу, сприятиме розробці інноваційного продукту з високим рівнем якості та користувацької задоволеності.

1.2.1 Meetup

Meetup - це глобальна онлайн-платформа, яка надає користувачам змогу створювати та знаходити події або спільноти за інтересами. Вона охоплює широкий спектр тем: від IT-зустрічей, освітніх курсів і мовних клубів до творчих майстерень і заходів для дозвілля. Платформа була створена з метою заохочення живої взаємодії між людьми через офлайн- і онлайн-зустрічі, що особливо актуально в умовах цифрового світу.

Meetup - це потужна платформа для організації подій, яка забезпечує глобальне охоплення та формування локальних спільнот за інтересами. Завдяки великій базі користувачів з усього світу, сервіс дозволяє створювати локальні осередки, які зберігають стійкий інтерес до певної тематики - наприклад, стартапи, програмування чи викладання. Це створює ідеальне середовище для професійного та соціального нетворкінгу.

Інтерфейс Meetup інтуїтивно зрозумілий, що дає змогу користувачам легко знаходити події як за географічним розташуванням, так і за тематикою. Платформа підтримує персоналізовані рекомендації та ефективну фільтрацію, що значно спрощує пошук і підвищує релевантність запропонованих заходів.

Мобільний застосунок для iOS і Android дозволяє швидко переглядати події, реєструватися на них та отримувати сповіщення в реальному часі. Інтеграція з Google Calendar або Apple Calendar дає змогу зручно планувати свою участь, не виходячи за межі звичних інструментів керування часом.

Ще однією перевагою платформи є підтримка постійних спільнот, які організовують регулярні зустрічі. Це сприяє формуванню стабільної аудиторії та забезпечує тяглість у взаємодії між учасниками. Таким чином, Meetup не лише спрощує організацію заходів, а й підтримує розвиток активних, самопідтримуваних спільнот.

Недоліки Meetup:

- наявність платної підписки для організаторів, що створює фінансові бар'єри для неприбуткових чи студентських ініціатив;
- відсутність україномовного інтерфейсу: платформа не підтримує українську мову, що ускладнює її використання для широкої частини аудиторії в Україні;
- обмежена популярність в Україні, що знижує активність локальних спільнот.

Meetup - це ефективний інструмент для організації подій та створення тематичних спільнот, особливо в англomовному середовищі або великих містах із розвиненою подієвою культурою. Проте для українських користувачів застосунок має низку бар'єрів - мовних, фінансових і культурних - що обмежують його використання як основного інструменту в локальних умовах. Це створює можливість для розробки українського аналога, адаптованого до потреб місцевої спільноти.

1.2.2 Bizzabo

Bizzabo - це SaaS-платформа для гібридних, віртуальних і офлайн-подій, що забезпечує організаторів інструментами для створення, просування та аналітики подій. Вона орієнтована переважно на бізнес-сегмент та великі компанії [4].

Переваги:

- широка функціональність: забезпечує повний цикл організації подій;
- інтуїтивний інтерфейс: для користувача та адміністратора;
- розширені можливості аналітики: детальні метрики для оцінки ефективності заходів;
- фокус на взаємодію: інструменти для нетворкінгу між учасниками.

Недоліки:

- складність інтеграції: великі об'єми коду, закриті API;
- висока вартість: орієнтація на корпоративний сегмент, що не підходить для малого/середнього ринку.

Bizzabo демонструє потужну реалізацію вебзастосунку для організації подій з розвиненою системою взаємодії між користувачами. Проте його складність та комерційна орієнтація роблять його менш придатним для гнучких кастомних рішень. Аналіз Bizzabo дозволяє визначити критичні функціональні елементи, які варто реалізувати у власному проєкті (реєстрація, профілі учасників, чат, інтерактивна програма подій), залишаючи при цьому простір для адаптації до цільової аудиторії, економічної ефективності та простоти використання.

1.2.3 Party.Space

Party.Space - український вебзастосунок для віртуальних святкових подій.

Party.Space - це інноваційний український вебзастосунок, який спеціалізується на організації віртуальних святкових подій, зокрема корпоративів, днів народження, тематичних вечірок та неформальних онлайн-зустрічей. На відміну від класичних платформ для організації подій, він фокусується на створенні соціального середовища та емоційної присутності в онлайн-форматі.

Основною метою Party.Space є забезпечення живої та інтерактивної взаємодії між учасниками події в умовах онлайн. До основних функціональних завдань належать [10]:

- створення віртуального простору для заходу, в якому кожен гість може “пересуватися” між зонами (наприклад, сцена, лаунж, бар), що імітує фізичну присутність;
- забезпечення відео- та голосового спілкування між учасниками, яке активується автоматично при “наближенні” до іншого гостя;
- інтеграція розважального контенту - міні-ігри, вікторини, музичні виступи, презентації;
- можливість налаштування дизайну та тематики події, що дозволяє персоналізувати простір відповідно до типу заходу (корпоратив, вечірка, день народження);
- управління гостями та аналітика участі, включно з контролем активності, відвідуваності та інтерактивної взаємодії користувачів.

Платформа має низку переваг, які вигідно вирізняють її на фоні традиційних систем:

- соціальна взаємодія в реальному часі: користувачі відчують ефект присутності завдяки відео-аватарам і змінним зонам взаємодії, що імітує справжні події;
- сильний емоційний ефект: оформлення простору, музика, тематика та інтерактиви створюють атмосферу свята, що особливо важливо для вечірок і неформальних заходів;
- універсальність застосування: система підходить як для корпоративних івентів, так і для приватних свят - днів народження, ювілеїв, зустрічей друзів;
- гейміфікація та розваги: міні-ігри, тести, конкурси інтегруються у віртуальне середовище без потреби в сторонніх сервісах.

Українське походження: платформа є конкурентоспроможним продуктом на міжнародному рівні, розробленим в Україні, що робить її релевантною для вітчизняного ринку.

Разом з тим, у Party.Space є певні недоліки, які варто враховувати під час проєктування власного вебзастосунку:

- технічна складність: платформа вимагає потужних браузерів, підтримки WebGL і стабільного інтернет-з'єднання, що обмежує її доступність для частини користувачів із слабшими пристроями;
- складність розробки: для реалізації подібного функціоналу потрібен значний ресурс - як технічний (відео-стрімінг, 3D-графіка), так і фінансовий;
- фокус виключно на онлайн: Party.Space орієнтований лише на віртуальні заходи, тому не надає інструментів для підтримки гібридного формату або організації офлайн-подій;
- відсутність традиційних елементів планування: немає таких стандартних функцій, як списки гостей, завдання для організатора тощо - фокус зроблений на «моменті події», а не її плануванні.

Аналіз Party.Space демонструє новий підхід до організації святкових подій у цифровому просторі, де головною цінністю є живе спілкування, індивідуальний

досвід і емоційне залучення гостей. Такий підхід вартий уваги при створенні власного вебзастосунку для організації подій. І хоча реалізація повного функціоналу Party.Space виходить за рамки типового дипломного проєкту, ідеї, закладені в платформу, можуть стати натхненням для більш простого, але емоційно насиченого інструменту для організації приватних свят.

1.3 Специфікація вимог до програмного забезпечення

Розроблюване програмне забезпечення є вебзастосунком, призначеним для організації та реєстрації на події різного формату - від невеликих зустрічей і тематичних семінарів до масштабних конференцій, фестивалів чи корпоративів. Основною метою створення цієї системи є спрощення процесу керування подіями для організаторів, а також забезпечення зручного й інтуїтивно зрозумілого способу реєстрації для учасників.

Програмний продукт буде орієнтований на максимальну інтеграцію з сучасними вебтехнологіями, без підтримки десктопних застосунків чи мобільних нативних клієнтів. Це дозволить зосередитися на універсальності доступу через веббраузер, адаптивності інтерфейсу та зменшенні загальної складності розгортання.

1.3.1 Межі проєкту

Розробка обмежується створенням веборієнтованого застосунку, який буде функціонувати у середовищі браузера та створенням мобільної версії сайту. У межах даного проєкту не передбачається створення мобільних додатків або десктопних програм. Основна увага зосереджена на адаптації функціональності до потреб сучасного вебкористувача.

1.3.2 Область застосування

Система призначена для використання під час організації й відвідування широкого спектру подій, включаючи: дні народження, корпоративні зустрічі;

професійні конференції; розважальні заходи та фестивалі; приватні події за запрошенням (за потреби).

1.3.3 Категорії користувачів та їх потреби

У межах системи передбачено три основні типи користувачів, кожен з яких має власні функціональні потреби та модель взаємодії з вебзастосунком: звичайні користувачі (учасники), організатори подій та адміністратори системи.

Звичайні користувачі (учасники) - це особи, які зацікавлені у пошуку та відвідуванні подій. Основна взаємодія для них полягає в перегляді афіші заходів, фільтрації за критеріями, реєстрації на події.

Ключові потреби: простий та адаптивний інтерфейс; ефективна система пошуку й фільтрації; швидка реєстрація, зокрема через акаунти соціальних мереж; доступ до актуальної інформації щодо подій.

Організатори подій - користувачі, які створюють та керують подіями в системі. Вони мають можливість додавати нові заходи, редагувати їх опис, контролювати кількість зареєстрованих учасників, використовувати аналітичні інструменти та комунікувати з аудиторією.

Ключові потреби: розширені можливості керування подіями; інструменти комунікації з учасниками.

Адміністратори системи - це користувачі, відповідальні за загальне управління платформою, включно з її безпекою, стабільністю, структурою прав доступу та оновленням системних модулів.

Ключові потреби: повний доступ до функціоналу системи; управління ролями користувачів; контроль за даними, правами доступу, технічним станом сервера; можливість модерації контенту та логування активності.

Розуміння ролей та потреб кожної категорії користувачів є ключовим чинником для побудови надійної, масштабованої та інтуїтивно зрозумілої системи. Вебзастосунок повинен враховувати ці аспекти на всіх етапах - від інтерфейсу до архітектури бази даних, забезпечуючи баланс між функціональністю, безпекою та зручністю використання для кожного типу користувачів.

1.3.4 Основні функції системи

У межах розроблюваного вебзастосунку передбачено реалізацію ключових функціональних можливостей, які забезпечують повний цикл взаємодії користувача з платформою - від реєстрації до управління подіями.

1.3.4.1 Авторизація користувачів

Функція авторизації дозволяє зареєстрованим користувачам входити в систему, використовуючи облікові дані.

Вхідні дані: логін (електронна пошта або інший ідентифікатор), пароль.

Вихідні дані: повідомлення про успішний вхід або помилка автентифікації.

Функціональні вимоги: перевірка коректності введених даних; повідомлення про неправильний логін чи пароль; можливість відновлення доступу (відновлення паролю).

1.3.4.2 Реєстрація нових користувачів

Ця функція забезпечує створення нового облікового запису для фізичної особи з внесенням базових контактних даних.

Вхідні дані: ім'я, електронна пошта, номер телефону, пароль.

Вихідні дані: підтвердження успішної реєстрації або повідомлення про помилки.

Функціональні вимоги: валідація введених даних; перевірка унікальності електронної пошти; повідомлення про успішну реєстрацію або наявність уже зареєстрованого користувача[5].

1.3.4.3 Особистий кабінет з переглядом подій

Особистий кабінет надає зареєстрованому користувачеві доступ до інформації про події, на які він зареєстрований, а також можливість керувати своїми реєстраціями.

Вхідні дані: ідентифікатор користувача (через сесію або токен доступу).

Вихідні дані: список зареєстрованих подій з короткими описами й можливістю перегляду деталей.

Функціональні вимоги: дотримання вимог конфіденційності; можливість скасування реєстрації на подію; відображення лише актуальних даних.

1.3.4.4 Управління персональними даними

Система забезпечує зберігання, перегляд та оновлення основної особистої інформації користувача.

Вхідні дані: ім'я, електронна пошта, номер телефону.

Вихідні дані: підтвердження успішного збереження або повідомлення про помилку.

Функціональні вимоги: можливість редагування персональних даних у кабінеті користувача; автоматична перевірка валідності введених значень; збереження змін у базі даних із захистом інформації.

1.3.4.5 Перегляд списку доступних подій

Користувачі, незалежно від авторизації, можуть переглядати всі майбутні події, доступні для реєстрації або участі.

Вхідні дані: не вимагаються.

Вихідні дані: перелік подій із відображенням короткої інформації про кожну (назва, дата, місце, опис).

Функціональні вимоги: динамічне оновлення інформації про події; можливість фільтрації за категоріями, датою, містом тощо; адаптивний інтерфейс для зручного перегляду на мобільних пристроях.

1.3.4.6 Швидка реєстрація на події в один клік

Ця функція дає змогу користувачам миттєво реєструватися на заходи, використовуючи збережені персональні дані, що знижує кількість необхідних дій.

Вхідні дані: ідентифікатори користувача та події.

Результат: підтвердження успішної реєстрації.

Функціональні особливості: автоматичне заповнення даних на основі раніше збереженої інформації користувача та можливість підтвердити дію перед остаточною реєстрацією.

1.3.4.7 Пошук подій за заданими критеріями

Ця функція забезпечує користувачам зручний інструмент для фільтрації та пошуку заходів за різними параметрами, зокрема за датою, місцем проведення та типом події.

Вхідні дані: вибрані критерії пошуку (дата, локація, категорія тощо).

Результат: перелік подій, що відповідають заданим умовам.

1.3.4.8 Створення та публікація подій (для організаторів)

Функція дозволяє організаторам легко створювати нові події та розміщувати їх на платформі для доступу користувачам.

Вхідні дані: назва, дата, час, місце проведення, опис події.

Результат: підтвердження успішного створення та публікації події.

Функціональні можливості: зручний інтерфейс для внесення інформації про подію, перевірка правильності введених даних (валідація), можливість інтеграції з календарем для ефективного планування.

1.3.4.9 Управління створеними подіями (для організаторів)

Функція надає організаторам повний контроль над подіями: редагування даних, контроль реєстрацій, оновлення інформації.

Вхідні дані: ідентифікатор події, нові або змінені дані.

Результат: підтвердження успішного оновлення інформації.

Функціональні можливості: інтерфейс для редагування деталей події, моніторинг змін у реєстраціях, доступ до базових аналітичних даних по події.

1.3.4.10 Відстеження реєстрацій (для організаторів)

Ця функція дозволяє організаторам у режимі реального часу переглядати статистику реєстрацій на події та аналізувати динаміку.

Вхідні дані: унікальний ідентифікатор події.

Результат: аналітична інформація щодо кількості та структури реєстрацій.

Функціональні можливості: розробка зручної аналітичної панелі (дашборду), інтеграція з базою даних для збору, зберігання та обробки інформації, забезпечення актуальності та точності відображених даних.

1.3.5 Нагадування про подію за 24 години до її початку

Ця функція автоматично інформує користувача про наближення події, на яку він зареєструвався, надсилаючи повідомлення за добу до її початку. Метою є зменшення кількості пропущених заходів та покращення користувацького досвіду.

Зареєстровані учасники подій (користувачі, які підтвердили участь).

Вхідні дані: ідентифікатор користувача; ідентифікатор події; дата та час початку події; канали сповіщення (електронна пошта, push-повідомлення, SMS - залежно від налаштувань користувача).

Вихідні дані: надіслане повідомлення (сповіщення) із деталями події.

Функціональні вимоги: автоматичне планування: після реєстрації користувача на подію система фіксує дату і час початку заходу. Автоматично створюється запланована задача на сповіщення за 24 години до події.

Налаштування користувача: можливість увімкнути/вимкнути нагадування; вибір каналу сповіщення; технічна реалізація. Для кожного відповідного користувача генерується повідомлення згідно з їхніми налаштуваннями.

Нефункціональні вимоги: висока надійність доставлення повідомлень; можливість масштабування (при великій кількості користувачів); безпечне зберігання контактних даних користувача.

1.3.6 Мобільна версія сайту

Платформа має адаптивну версію для мобільних пристроїв, яка забезпечує комфортне використання функціоналу сайту зі смартфонів та планшетів.

Вхідні дані: розмір екрану пристрою.

Результат: інтерфейс, оптимізований під мобільні екрани.

Ключові вимоги: реалізація адаптивного дизайну, збереження повної функціональності основного сайту, забезпечення функціональності всіх основних функцій сайту в мобільній версії, оптимізація швидкодії та продуктивності для мобільних пристроїв.

1.3.7 Архітектура та розробка програмної системи

Система побудована за принципом монолітної архітектури, де вебзастосунок складається з взаємопов'язаних сервісів, що спільно використовують єдину базу даних. Така структура спрощує розгортання та супровід, особливо на етапі ранньої розробки.

Для розробки використовуються такі середовища:

- Visual Studio Code – для фронтенду та роботи з інтерфейсами[5];
- Firebase - для роботи з базами даних.

Комунікація між компонентами системи забезпечується через сучасні мережні протоколи, зокрема HTTPS [6], що гарантує безпечний обмін даними.

Основним інструментом адміністрування та роботи з базами даних є - Firebase. Він забезпечує: зручне створення та редагування структур бази даних, управління безпекою доступу, модифікацію, налаштування параметрів серверів, графічний інтерфейс для роботи з даними.

Для написання Front-end використовувались наступні технології: React, TypeScript, RTK query, Zustand. Для бази даних: Firebase [7].

1.3.8 Інтерфейс користувача (UI/UX)

Інтерфейс має бути максимально зручним та інтуїтивно зрозумілим для користувачів з різним рівнем технічної підготовки. Застосування сучасних принципів UI/UX дизайну забезпечує легку навігацію, доступ до основних функцій і приємний користувацький досвід.

1.3.9 Нефункціональні вимоги

Система повинна бути постійно доступною - 24 години на добу, 7 днів на тиждень, без тривалих збоїв чи технічних перерв, забезпечуючи надійний сервіс для всіх користувачів.

Програмне забезпечення має бути легко підтримуваним: із добре структурованим кодом, повною технічною документацією та налагодженим механізмом зворотного зв'язку для швидкого реагування на помилки та пропозиції користувачів.

Архітектура системи повинна дозволити її розгортання на різних платформах і в різних середовищах з мінімальними змінами. Цього можна досягти шляхом компонентного підходу для фронтенду. [8]

Система повинна швидко обробляти великі обсяги інформації та одночасні запити, забезпечуючи час відгуку не більше ніж 3 секунди навіть при високому навантаженні.

Функціонування системи має бути стабільним і безперервним. Необхідно мінімізувати кількість збоїв, помилок і відмов, особливо в критичних компонентах, що впливають на роботу користувача.

Система повинна гарантувати високий рівень захисту персональних даних. Передбачено використання сучасних методів шифрування, безпечної автентифікації на основі токенів, які зберігаються в середовищах, недоступних для клієнтської сторони.

1.4 Постановка задачі

Даний проєкт створюється задля забезпечення ефективної організації різноманітних подій та взаємодії користувачів для того, щоб об'єднати учасників навколо спільних інтересів і забезпечити зручну платформу для планування заходів. Він дає змогу швидко поширювати інформацію, управляти списками гостей, збирати зворотний зв'язок і підтримувати активну комунікацію між організаторами та учасниками. Такий інструмент сприяє кращій координації й підвищенню якості проведення подій. Отже, додаток повинен об'єднувати функції різних рішень та надавати їх в зручному вигляді.

Основною функцією є власне створення події та її афішування. Організатор повинен мати можливість редагувати інформацію про подію, наприклад опис, назва чи постер. Щодо саме афішування, додаток матиме окреме меню, де можна подивитись загальний список відкритих подій, відфільтрувати їх за датою проведення чи популярністю, а також доєднатись до них в якості гостя. Важливо, щоб це меню не було перенавантажено інформацією, адже такий дизайн UI може не лише відволікати користувача від перегляду подій, а й змусити його відмовитись від додатку в цілому.

Як було зазначено, дизайн інтерфейсу додатку повинен бути досить простим, не перевантаженим інформацією, та мати інтуїтивно зрозумілу навігацію.

Для досягнення поставленої мети передбачено виконання наступних завдань:

- провести аналіз предметної області: вивчити існуючі рішення (наприклад, Meetup, Bizzabo, Party.Space), оцінити їхні функції, переваги та обмеження;
- визначити основні потреби користувачів: сформулювати цільові ролі - організатор, учасник, адміністратор — та визначити їх функціональні очікування від системи;
- сформулювати вимоги до системи: описати функціональні та нефункціональні характеристики майбутнього застосунку;
- розробити структуру вебзастосунку: спроектувати архітектуру, обрати відповідний стек технологій, створити моделі даних і взаємодії;
- реалізувати застосунок: створити інтерфейс користувача, реалізувати основний функціонал (створення/перегляд подій, комунікація, модерація), забезпечити зручну навігацію та взаємодію;
- провести тестування: здійснити перевірку працездатності, стабільності та відповідності застосунку технічним і функціональним вимогам;
- сформулювати висновки: оцінити результати проєкту, обґрунтувати ефективність розробленого рішення та визначити перспективи його вдосконалення або масштабування.

У результаті виконання проєкту буде створено повнофункціональний вебзастосунок, орієнтований на користувача, який забезпечує ефективну організацію подій та зручну взаємодію між учасниками.

У першому розділі бакалаврської кваліфікаційної роботи проведено всебічний аналіз наявних вебзастосунків для організації подій. Особливу увагу приділено їх функціональним можливостям, технологічній реалізації, рівню зручності для користувачів, позиціонуванню на ринку та відгукам цільової аудиторії.

Проведене дослідження дало змогу виділити ключові напрями розвитку, серед яких - потреба в інтеграції з календарями, важливість створення простих та інтуїтивно зрозумілих інтерфейсів, а також необхідність адаптації до мобільних

пристроїв. Це дозволило окреслити можливості для впровадження інновацій, здатних забезпечити розроблюваній системі конкурентні переваги [4].

У процесі аналізу також сформульовано базовий перелік функціональних компонентів майбутнього застосунку: швидка реєстрація, персоналізовані форми опитування, а також інструменти аналітики для організаторів подій.

Окрім того, визначено необхідні технології та середовище розробки, що створює основу для побудови ефективної, надійної та безпечної системи, здатної задовольнити потреби як організаторів, так і учасників подій, полегшуючи процес управління та участі в заходах.

2 МОДЕЛЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У сучасному програмному інжинірингу досягнення високої якості, надійності та масштабованості системи неможливе без чіткого попереднього проектування її архітектури та поведінки. Для веб-додатка Eventify, який призначений для створення, планування та керування подіями, моделювання програмного забезпечення відіграє ключову роль на всіх етапах розробки.

По-перше, моделі допомагають формалізувати вимоги до Eventify: ми будуємо діаграми прецедентів (use-case), щоб відобразити взаємодію користувачів (організаторів та учасників) із системою, і на їхній основі уточнюємо функціональні сценарії (створення події, вибір дати/часу, завантаження зображень, управління підписками тощо).

По-друге, моделювання дозволяє вибрати оптимальну архітектуру фронтенду. Для мого проекту це означає:

- структурне моделювання - діаграми класів та компонентів, що описують сутності події, користувача, підписки, геопозиції та взаємозв'язки між ними;

- поведінкове моделювання - діаграми послідовностей відображають процес реєстрації нового користувача, створення івенту, підтвердження участі та надсилання сповіщень.

По-третє, за допомогою моделей легше проводити верифікацію й тестування: використовуючи діаграми активності ми з'ясовуємо потенційні «вузькі місця» у логіці обробки даних (наприклад, фільтрація доступних часових слотів) та запобігаємо помилкам ще до написання коду [9].

Нарешті, моделі слугують комунікаційним інструментом між усіма учасниками процесу – від аналітиків і дизайнерів до розробників та тестувальників. Вони забезпечують прозорість і зрозумілість внутрішньої структури Eventify, що особливо важливо в умовах командної розробки й майбутньої підтримки проєкту.

Таким чином, моделювання програмного забезпечення для Eventify не лише формалізує вимоги та проєктні рішення, а й значно підвищує гнучкість, якість і надійність створюваної системи. У наступних розділах буде детально описано побудову UML-діаграм, вибір нотацій та підходів, а також інструменти, які використовувалися для моделювання архітектури та бізнес-процесів нашого додатка.

2.1 Сценарії використання системи

2.1.1 Опис діаграми прецедентів додатку Eventify (вертикальна версія)

Діаграма прецедентів додатку Eventify, зображена на рисунку 2.1, відображає взаємодію двох акторів - користувача та адміністратора сайту - із системою Eventify. Вона демонструє функціональні можливості, доступні кожному актору, та чітко визначає межі системи. Діаграма створена у вертикальному форматі з акторами зліва для максимальної читабельності та зручності сприйняття.

Діаграма прецедентів додатку Eventify (вертикальна версія)

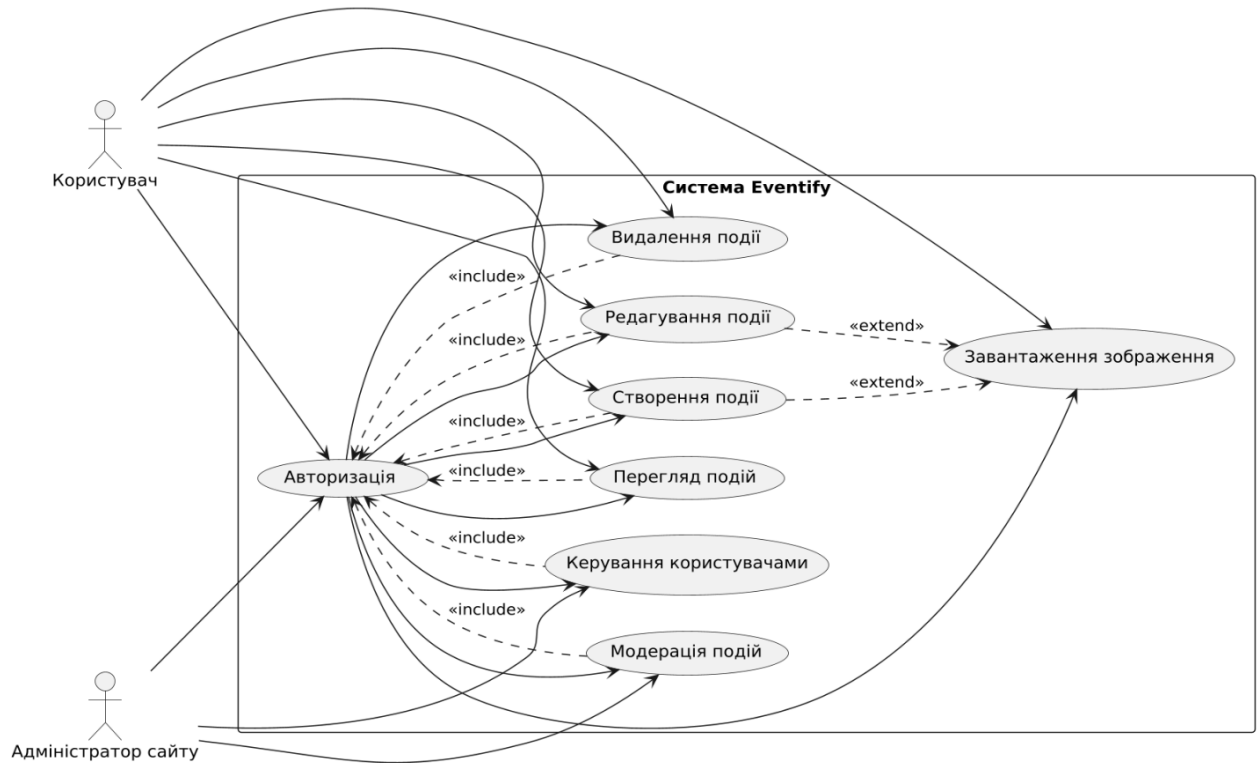


Рисунок 2.1 – Діаграма прецедентів застосунку

У системі «Eventify» визначено двох основних акторів - користувача та адміністратора сайту. Користувач взаємодіє з додатком з метою створення, редагування, перегляду або видалення подій, а також має змогу додавати до подій зображення. Адміністратор, у свою чергу, виконує функції модерації, які включають перевірку подій, схвалення або відхилення їх відповідно до встановлених вимог платформи, а також здійснює управління обліковими записами користувачів.

Ключовою передумовою для здійснення більшості дій у системі є процес авторизації. Вона є обов'язковою як для користувача, так і для адміністратора, оскільки забезпечує доступ до функцій, які потребують автентифікації. Саме тому прецеденти, що передбачають роботу з подіями або акаунтами, логічно включають авторизацію як необхідну дію. Це відображається у вигляді зв'язку типу <<include>>.

Окремі функції системи є умовно необов'язковими, що відображено через використання зв'язків типу <<extend>>. Зокрема, процеси створення та

редагування подій можуть бути доповнені можливістю завантаження зображення. Це дозволяє забезпечити більшу гнучкість функціоналу без ускладнення основного процесу взаємодії з подіями.

Діаграма прецедентів реалізована у вертикальному форматі, де всі прецеденти розташовані у межах прямокутника, що символізує межі системи «Eventify». Така організація забезпечує структурованість і зменшує кількість перетинів зв'язків між елементами. Актори системи розташовані зліва від меж системи, що відповідає загальноприйнятим принципам побудови діаграм типу use case. Зв'язки між акторами та відповідними прецедентами реалізовані у вигляді спрямованих стрілок, що вказують напрямок взаємодії.

Усі зв'язки типу <<include>> та <<extend>> виконані у формі пунктирних стрілок, розміщених всередині меж системи, що сприяє логічному групуванню та полегшує сприйняття структури діаграми. Візуальна реалізація діаграми витримана в монохромному стилі, без використання кольорів, з метою акцентування уваги на змістовній частині взаємодії між елементами системи[10].

Таким чином, діаграма прецедентів слугує засобом графічного представлення функціональності додатку «Eventify» та ілюструє типову поведінку користувачів і адміністраторів під час взаємодії з ключовими модулями системи.

2.2 Прецедент "Створення події"

Прецедент «Створення події», представлений на рисунку 2.2, описує процес, за допомогою якого користувач може створити нову подію в системі Eventify. Це одна з ключових функцій додатку, яка забезпечує основне призначення системи - надання можливості організації подій та їх представлення іншим користувачам. У межах даного прецеденту користувач має змогу ввести всю необхідну інформацію про подію, зокрема її назву, опис, дату, час початку та завершення, координати місця проведення (широту і довготу), час можливого чек-іну, обмеження за віком для

учасників, а також умови паркування. Крім цього, передбачена додаткова опція - завантаження зображення для події, яка реалізується як розширення до основного сценарію.

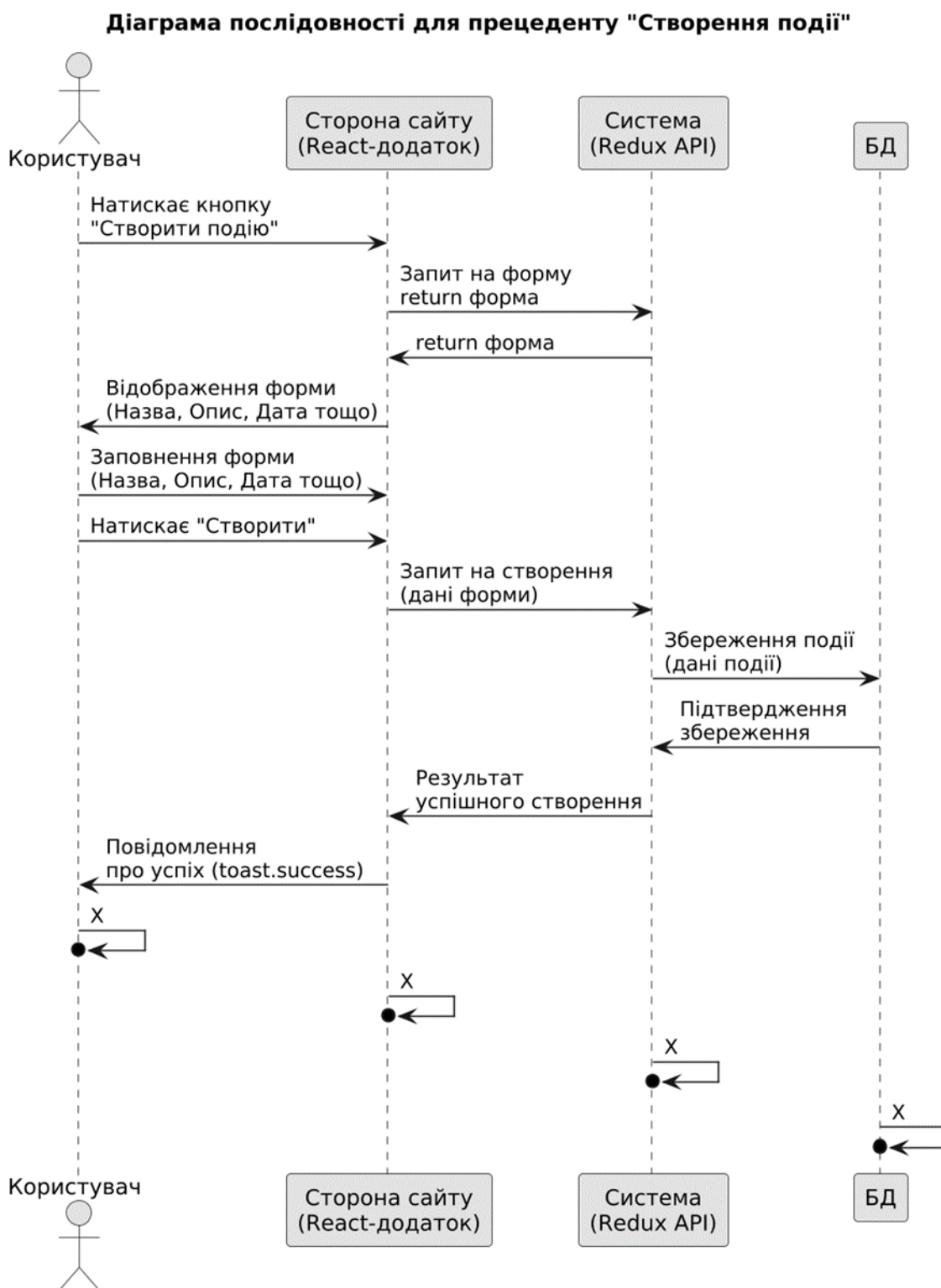


Рисунок 2.2 – Створення події

Функціональність створення події доступна лише авторизованим користувачам. Відтак, перед початком взаємодії з інтерфейсом створення події, користувач повинен пройти процедуру авторизації, яка виступає необхідною передумовою виконання дії. Авторизація реалізована як включений прецедент, що забезпечує автентифікацію користувача та надає йому доступ до інтерфейсу створення.

Після авторизації користувач відкриває інтерфейс створення події, який реалізовано у вигляді модального вікна. У цьому вікні відображається форма, що містить текстові поля для введення основної інформації, елементи вибору дати та часу, опції для налаштування вікових обмежень, а також параметри, пов'язані з паркуванням. Якщо користувач бажає додати зображення до події, він має можливість скористатися функціоналом завантаження, що реалізується через компонент `CldUploadWidget`. Після вибору файлу зображення система здійснює його завантаження до хмарного сховища `Cloudinary` та додає відповідне посилання до метаданих події.

2.2.1 Основний сценарій

Після завершення введення усієї інформації користувач підтверджує створення події натисканням відповідної кнопки. Система перевіряє коректність заповнення обов'язкових полів, таких як назва події, дата, час і координати. У випадку успішної валідації, дані передаються до серверної частини через відповідний інтерфейс API, реалізований із використанням `Redux`, після чого подія зберігається в базі даних [11]. У разі успішного завершення процесу користувач отримує повідомлення про успішне створення події, і ця подія стає доступною для перегляду в особистому кабінеті.

2.2.2 Альтернативні сценарії

Система передбачає обробку виключних ситуацій. Зокрема, якщо користувач не авторизований, йому буде повернуто повідомлення про необхідність входу в систему. Якщо під час заповнення форми були пропущені обов'язкові поля або допущено помилки, система повідомить про це та не допустить завершення створення до усунення недоліків. У разі помилки завантаження зображення подія

все одно може бути створена, але без візуального супроводу, або користувач має можливість повторити спробу завантаження.

Також можуть виникати ситуації, пов'язані з проблемами під час збереження події (наприклад, через збій сервера або нестабільне інтернет-з'єднання). У такому разі система відображає відповідне повідомлення про помилку, після чого користувач може повторити операцію.

Прецедент «Створення події» має два основних логічні зв'язки з іншими прецедентами: включення до нього процесу авторизації користувача, що забезпечує безпеку та контроль доступу, а також можливість розширення через додатковий прецедент «Завантаження зображення». Така модель дозволяє забезпечити як базовий сценарій використання, так і підтримку розширеного функціоналу для покращення візуального представлення подій.

Загалом, даний прецедент є центральною складовою системи Eventify, оскільки реалізує базову функцію створення подій. Його реалізація демонструє гнучкість у введенні користувацьких даних, дотримання принципів інтерфейсної доступності та підтримку інтеграції з зовнішніми сервісами (Cloudinary) для покращення користувацького досвіду.

2.3 Прецеденти "Перегляд подій", "Видалення події" та "Редагування події"

У цьому підрозділі розглядаються функціональні можливості додатку Eventify, що дозволяють авторизованому користувачу переглядати, редагувати або видаляти свої події. Всі три дії передбачають наявність авторизації в системі та базуються на основному прецеденті - «Перегляд подій». Додатково кожна з дій може включати обробку виключних ситуацій, пов'язаних із помилками доступу, обробки або взаємодії з сервером.

2.3.1 Прецедент «Перегляд подій»

Прецедент «Перегляд подій» полягає в наданні користувачеві можливості переглянути список своїх створених подій разом із їхніми атрибутами, такими як назва, опис, дата, час і місце проведення. Після проходження авторизації користувач переходить до інтерфейсу перегляду подій, де система, використовуючи Redux API, запитує дані з бази. У відповідь база даних повертає відповідний набір подій, який відображається на фронтенді за допомогою React-компонентів. Кожна подія представлена у вигляді картки або запису в таблиці з відповідними даними.

Основний сценарій прецеденту передбачає, що після авторизації користувач переходить до розділу «Мої події». Система надсилає запит до серверної частини, яка звертається до бази даних і повертає актуальний список подій. Дані передаються у React-додаток, де виводяться на екран.

Альтернативний сценарій передбачає, що у разі виникнення помилки завантаження (наприклад, через перебої з мережею або збої API), система генерує повідомлення про помилку, наприклад, через `toast.error`. Користувач може повторити спробу після усунення проблеми.

2.3.2 Прецедент «Видалення події»

Прецедент «Видалення події» активується тоді, коли користувач бажає видалити певну подію зі свого списку. Дія доступна лише після авторизації та перегляду списку подій. Користувач обирає конкретну подію та натискає кнопку «Видалити», після чого система відкриває модальне вікно для підтвердження дії. Якщо користувач підтверджує намір, система надсилає запит на видалення до серверної частини, яка видаляє відповідний запис із бази даних і підтверджує успішне виконання. React-додаток у відповідь оновлює список подій та виводить повідомлення про успіх.

Основний сценарій передбачає наступну послідовність: авторизований користувач переглядає список подій, обирає подію для видалення, підтверджує дію в модальному вікні, після чого відбувається обробка запиту, оновлення бази даних і повернення нового списку подій, який відображається в інтерфейсі.

Альтернативні сценарії включають ситуацію, коли користувач скасовує підтвердження видалення, або коли відбувається помилка видалення - у такому разі система інформує користувача про невдачу, дозволяючи повторити дію пізніше.

2.3.3 Прецедент «Редагування події»

Прецедент «Редагування події» реалізує можливість оновлення атрибутів існуючої події. Після авторизації та перегляду списку подій користувач натискає кнопку «Редагувати» на відповідній картці. Система отримує поточні дані події з бази даних і передає їх до React-додатку, який відображає форму редагування. Користувач змінює необхідні поля - наприклад, дату, опис або місце проведення - і натискає кнопку «Зберегти». Після цього система оновлює відповідний запис у базі даних, і в разі успішного виконання повертає оновлений список подій. У разі, якщо користувач додає зображення до події, до цього прецеденту може бути динамічно доданий додатковий функціонал - «Завантаження зображення».

Основний сценарій передбачає послідовність: авторизація → перегляд подій → відкриття форми редагування → редагування даних → збереження → оновлення бази → підтвердження змін → оновлений інтерфейс.

Альтернативні сценарії включають ситуації, коли користувач залишає обов'язкові поля порожніми або неправильно заповнює дані - у такому разі система повертає повідомлення про помилку й не дозволяє зберегти зміни. Також можлива помилка оновлення через технічні збої, яка супроводжується відповідним сповіщенням.

Усі три прецеденти мають спільні риси - зокрема, вони потребують авторизованого доступу та базуються на одному і тому ж джерелі даних. Інтерфейс реалізований за допомогою React-компонентів, які тісно інтегровані з Redux-архітектурою, що забезпечує ефективну обробку запитів до серверної частини [11].

Діаграма послідовності для цих прецедентів зображена на рисунку 2.3

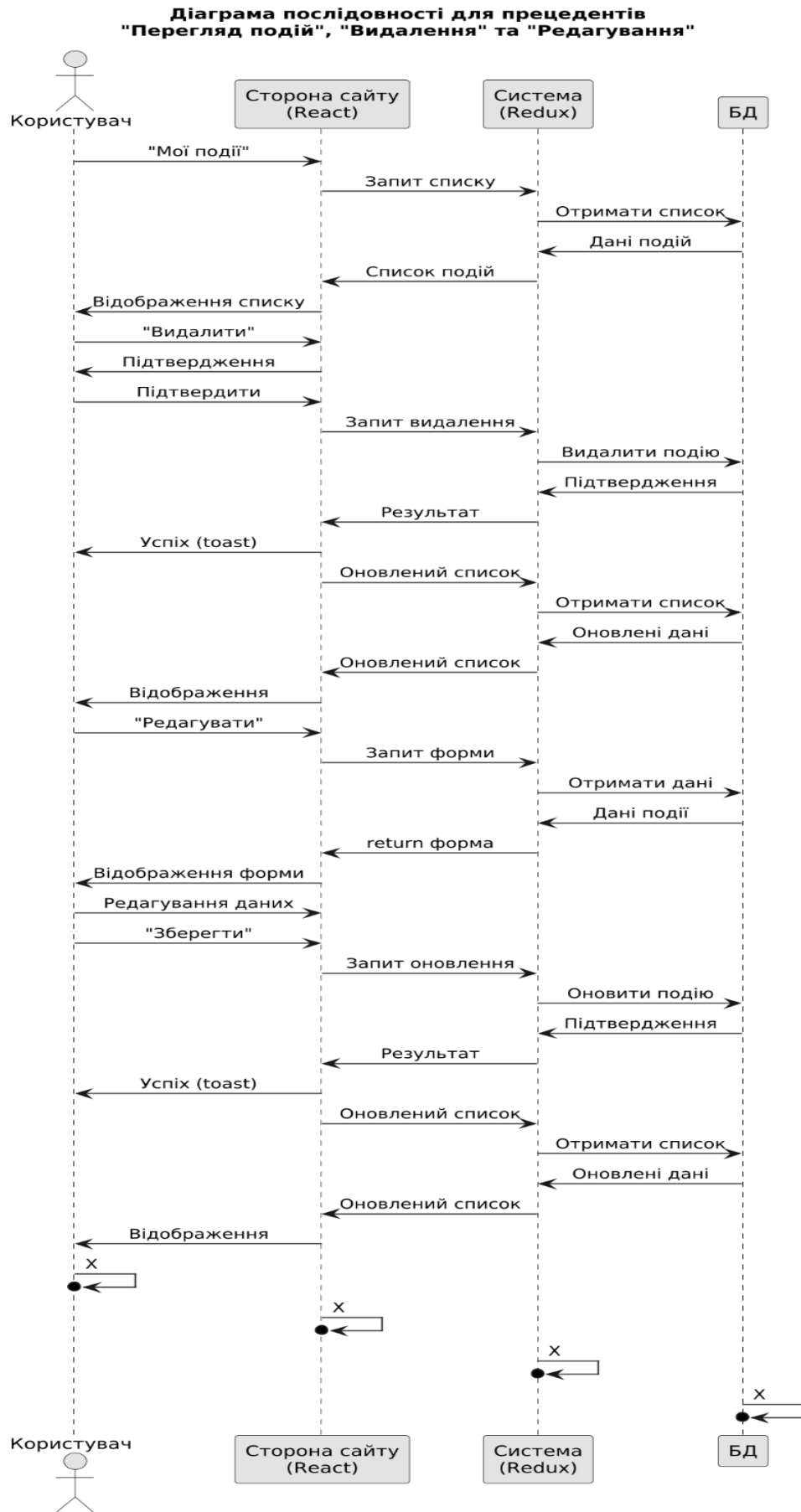


Рисунок 2.3 – Діаграма прецедентів: "Перегляд подій", "Видалення події" та "Редагування події"

2.4 Прецеденти "Підписка на подію" та "Перегляд підписаних подій"

Розглянуто два ключові прецеденти взаємодії користувача з додатком Eventify: підписка на події, створені іншими користувачами та перегляд списку подій, на які вже оформлено підписку. Обидва прецеденти передбачають обов'язкову авторизацію користувача в системі, що забезпечує індивідуалізовану роботу з даними та персоналізований досвід.

2.4.1 Прецедент "Підписка на подію"

Прецедент "Підписка на подію" описує процес, за якого авторизований користувач, переглядаючи доступні події у відповідному інтерфейсі, може обрати певну подію та ініціювати підписку на неї шляхом натискання кнопки «Підписатися». Ця дія запускає запит, який надсилається з React-додатку до серверної частини через Redux API. Після надходження запиту система здійснює додавання відповідного запису до бази даних, який встановлює зв'язок між ідентифікатором користувача та ідентифікатором події. У разі успішного виконання операції користувач отримує підтвердження у вигляді сповіщення про успішну підписку. Відповідно, інтерфейс змінює стан кнопки, наприклад, на «Відписатися» або відображає статус участі.

Однак існують і альтернативні сценарії реалізації цього прецеденту. Перший полягає в тому, що користувач уже підписаний на подію: у такому випадку система розпізнає дублювання і повертає повідомлення про це, після чого React-додаток інформує користувача про неможливість повторної підписки. Інший альтернативний розвиток подій пов'язаний з технічною помилкою при виконанні операції, наприклад, через проблеми з мережею або збої бази даних. У такому разі користувачу пропонується повторити спробу пізніше, а інтерфейс відображає відповідне повідомлення про помилку.

Ілюстрацією логіки реалізації взаємодій виступає діаграма послідовності, зображена на рисунку 2.4.

**Діаграма послідовності для прецедентів
"Підписка на подію" та "Перегляд підписаних"**

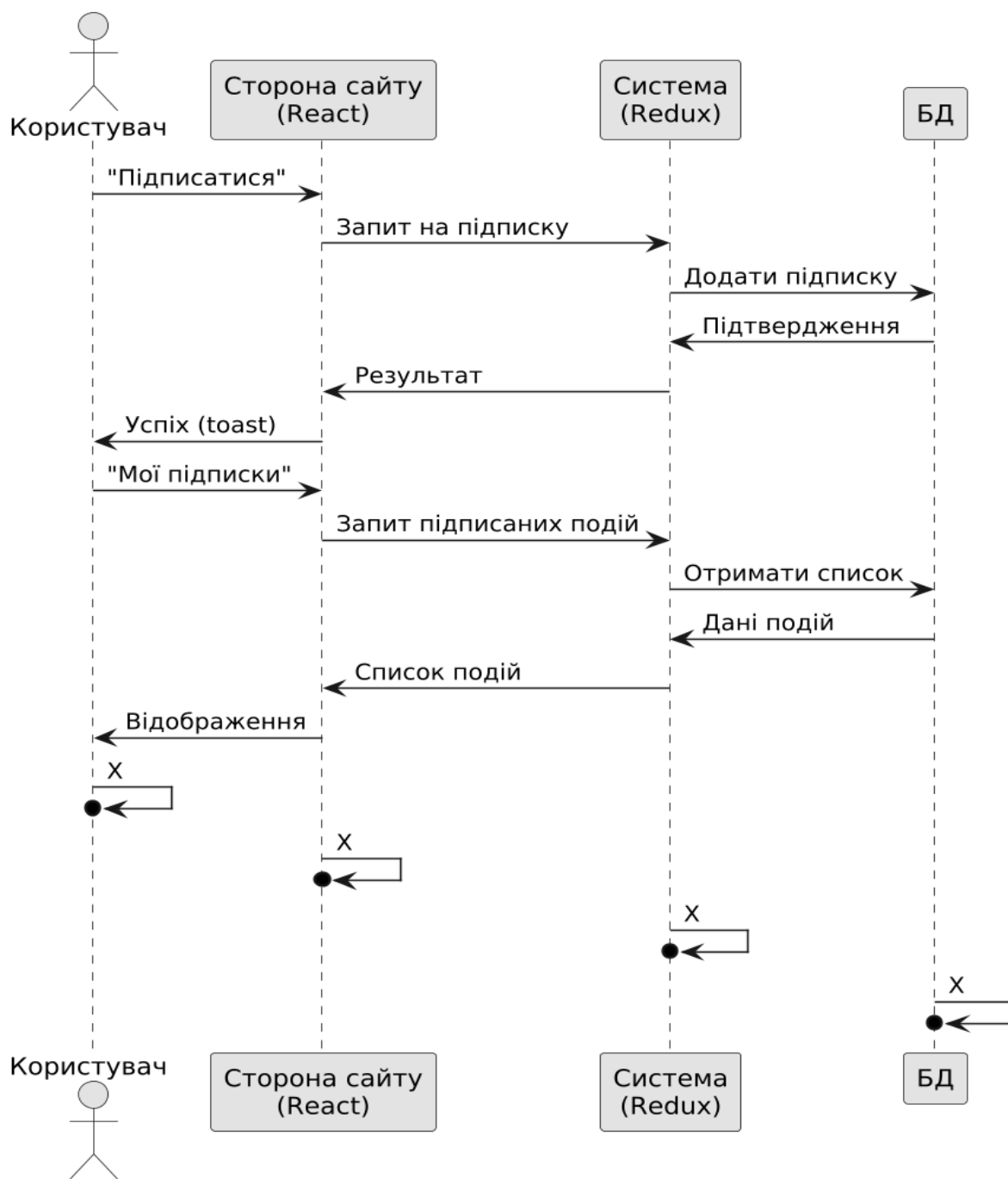


Рисунок 2.4 - Діаграма прецедентів "Підписка та перегляд подій"

Прецедент "Перегляд підписаних подій" спрямований на надання користувачу можливості ознайомитися з переліком подій, на які він раніше підписався. Після авторизації користувач отримує доступ до інтерфейсу, де реалізовано відповідний функціонал. Натискання на пункт «Мої підписки» ініціює запит до серверної частини для отримання актуального списку підписаних подій.

Система звертається до бази даних, використовуючи ідентифікатор поточного користувача, та повертає перелік відповідних подій. Отримані дані обробляються фронтенд - частиною додатку, і результати відображаються у вигляді карток або таблиці з деталізованою інформацією (назва, опис, дата, час, місце проведення тощо).

Альтернативний сценарій реалізації цього прецеденту може включати помилки при завантаженні даних, наприклад, у випадку порушення з'єднання з сервером або збою при зверненні до бази даних. У такій ситуації інтерфейс повідомляє користувача про невдачу та пропонує повторити спробу. Інший можливий варіант розвитку подій передбачає, що користувач ще не підписаний на жодну подію. У такому разі система повертає порожній список, а інтерфейс відповідно повідомляє про відсутність підписок.

Обидва прецеденти є логічним продовженням базового прецеденту авторизації та розширюють можливості користувача щодо взаємодії з платформою, забезпечуючи персоналізацію, збереження участі в заходах та зручне планування відвідин подій.

У другому розділі кваліфікаційної роботи бакалавра розглянуто моделювання вебзастосунку організації подій, детально описуючи структуру системи, сценарії використання, процеси розробки діаграм розгортання, взаємодії, а також діяльності. Деталізоване викладення архітектури системи, включаючи фронтенд, 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. Це, у свою чергу, відкриває шлях для створення ефективного та привабливого вебзастосунку, орієнтованого на забезпечення високої взаємодії і задоволення потреб користувачів.

3 ПРОЄКТУВАННЯ ВЕБСЕРВІСУ

3.1 Архітектура вебсервісу

Архітектура вебсервісу Eventify побудована за принципом серверлес-архітектури, де клієнтська частина на пряму взаємодіє з Firebase як основним сервісом для обробки даних та автентифікації. Такий підхід забезпечує простоту розробки, масштабованість та зручність інтеграції із зовнішніми сервісами, такими як Cloudinary для завантаження зображень.

Структура системи для вебзастосунку організації подій розподілена на дві основні частини: фронтенд (Front-end) і Firebase.

Клієнтська частина (Frontend):

- реалізована з використанням React.js для створення інтерактивного інтерфейсу користувача;
- використовує Redux Toolkit для управління станом додатку (наприклад, списки подій, статус авторизації);
- інтегрується з Cloudinary через CldUploadWidget для завантаження зображень подій [9];
- на пряму взаємодіє з Firebase для обробки запитів, авторизації та збереження даних.

Firebase використовується як серверлес-платформа, що замінює традиційний бекенд. Його використання дозволило відмовитися від окремого сервера, оскільки Firebase забезпечує повний набір функцій, необхідних для повноцінної роботи застосунку - зокрема автентифікацію користувачів, зберігання даних у реальному часі та обробку запитів. Це рішення дозволяє значно пришвидшити розгортання

проєкту, зменшити витрати на інфраструктуру та зосередитися на розробці функціоналу.

Основні функції Firebase у додатку:

- Firebase Authentication забезпечує авторизацію користувачів (наприклад, через email/пароль або соціальні мережі);
- Firestore реляційна NoSQL база даних для зберігання даних про користувачів, події та підписки;
- Firebase Functions (опціонально) для виконання серверлес-логіки, якщо потрібна додаткова обробка (наприклад, модерація подій) [9].

Основні колекції у Firestore:

- Users - інформація про користувачів (ID, ім'я, email);
- Events - інформація про події (ID, назва, опис, дата, час, місце, організатор тощо);
- Subscriptions - зв'язок між користувачами та подіями, на які вони підписані.

Окремо варто виділити інтеграцію із зовнішнім сервісом Cloudinary, який використовується для завантаження, зберігання та обробки зображень, що додаються до подій. Завдяки інтеграції через компонент CldUploadWidget, користувачі можуть легко завантажувати фото безпосередньо з клієнтської частини застосунку. У відповідь сервіс генерує URL зображення, який згодом зберігається у відповідному документі колекції Events у Firestore. Це дозволяє уникнути перенавантаження бази великими файлами та водночас гарантує швидке й оптимізоване завантаження зображень на стороні користувача [10].

Загалом, обрана структура бази даних та зовнішні інтеграції відображають прагнення до створення гнучкої, масштабованої та зручно підтримуваної архітектури застосунку, яка відповідає сучасним вимогам до веброботи.

Взаємодія компонентів у запропонованій архітектурі побудована на принципі тісної інтеграції між клієнтською частиною (frontend) та хмарними сервісами, що забезпечують зберігання, обробку й захист даних. Цей підхід дозволяє досягти високої швидкодії застосунку, гнучкості в масштабуванні та спрощеної логіки керування даними.

У центрі взаємодії знаходиться користувач, який працює із застосунком через браузер. Його дії - такі як реєстрація, авторизація, створення нової події або підписка на вже існуючу - ініціюють запити до системи. Усі ці запити обробляються фронтом, розробленим на основі React. Саме клієнтська частина відповідає за візуалізацію інтерфейсу, обробку подій користувача та комунікацію з хмарними сервісами.

Після взаємодії користувача з інтерфейсом застосунок надсилає запити до Firebase. Якщо йдеться про автентифікацію, то запит передається до Firebase Authentication, де перевіряється правильність введених даних (наприклад, email і пароль або авторизація через Google). У разі успішної перевірки користувач отримує доступ до персоналізованих функцій системи.

Дані, пов'язані з подіями або підписками, зберігаються у Firestore - гнучкій NoSQL базі даних, де реалізована підтримка роботи в реальному часі. Запити на створення, оновлення чи видалення інформації про події надсилаються з фронту безпосередньо до Firestore, який обробляє їх згідно з заданими правилами безпеки. Наприклад, коли користувач створює подію, система формує новий документ у відповідній колекції; якщо він підписується на подію - створюється запис у колекції Subscriptions, що зв'язує конкретного користувача з подією.

Паралельно з обробкою текстових даних, застосунок підтримує роботу з візуальним контентом. Завантаження зображень здійснюється через сервіс Cloudinary, з яким фронтенд взаємодіє напряму через CldUploadWidget. Користувач обирає зображення, воно надсилається до Cloudinary, який у відповідь генерує URL-адресу зображення. Цей URL далі зберігається у відповідному документі події у Firestore, що дозволяє ефективно завантажувати зображення без потреби зберігати їх у самій базі [11].

На завершальному етапі дані, що надійшли з Firebase (як текстові, так і візуальні), відображаються користувачу через фронтенд. Наприклад, формується список подій з урахуванням підписок, фільтрів і вподобань користувача.

Окрему роль у системі виконує адміністратор, який так само працює через клієнтський інтерфейс, однак має розширені права доступу. Йому надається

можливість модерації контенту, редагування чи видалення подій, а також керування користувачами. Усі його дії також проходять через Firebase, що дозволяє централізовано контролювати процеси управління.

Вся система працює під управлінням програмного забезпечення, яке забезпечує реалізацію основних функцій системи. Загальна структура програмного забезпечення представлена на рис. 3.1.

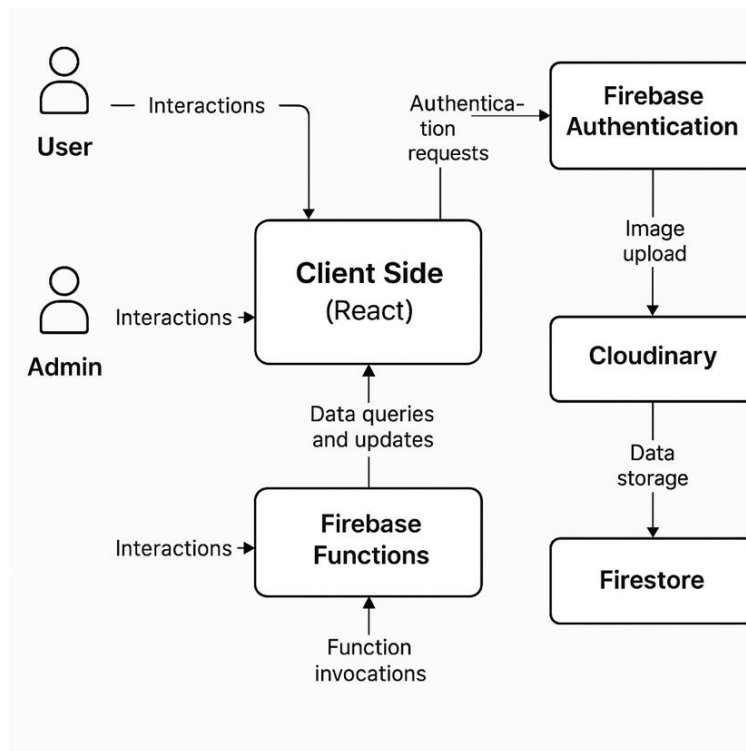


Рисунок 3.1 – Архітектура програмного забезпечення системи вебзастосунку «Eventify App»

Таким чином, взаємодія всіх компонентів - користувача, клієнтської частини, Firebase, Firestore, Cloudinary - створює цілісну, узгоджену архітектуру, яка дозволяє застосунку працювати стабільно, швидко й безпечно, навіть за великої кількості одночасних користувачів.

Основні особливості архітектури вебзастосунку для організації подій Eventify, які формують її переваги та ефективність у реальному використанні:

- використання Firebase усуває необхідність у традиційному бекенді, спрощуючи розробку та розгортання;

- Firebase Authentication забезпечує безпечну авторизацію, а правила безпеки Firestore контролюють доступ до даних;
- Firebase автоматично масштабується при зростанні кількості користувачів, забезпечуючи стабільну роботу;
- Cloudinary ефективно управляє зображеннями, зменшуючи навантаження на систему;
- поєднання React, Firebase та Cloudinary дає змогу зосередитися на логіці й дизайні продукту, зменшуючи технічну складність інфраструктури;
- система підтримує як звичайних користувачів (учасників подій), так і адміністраторів, які мають доступ до додаткових функцій, таких як модерація подій, керування контентом тощо.

Ці характеристики роблять архітектуру застосунку ефективною, адаптивною до змін і придатною для швидкого розгортання у хмарному середовищі.

3.2 Проектування Cloud Firestore

3.2.1 Сутності та їхня структура

Cloud Firestore використовується у додатку Eventify як основна база даних для зберігання інформації про події, користувачів та сповіщення. Firestore є NoSQL базою даних, що дозволяє гнучко структурувати дані у вигляді колекцій та документів. Описано структуру основних сутностей - Event (Подія), User (Користувач) та Notification (Сповіщення), а також їхні взаємозв'язки.

3.2.1.1 Event (Подія)

Структура сутності Event (Подія) в базі даних Cloud Firestore є важливим елементом для розуміння організації зберігання даних у застосунку Eventify. Оскільки Firestore є документно-орієнтованою NoSQL базою даних, усі сутності реалізовано у вигляді колекцій, що містять документи з вкладеними полями та

мапами. Такий підхід забезпечує гнучкість, масштабованість та зручність доступу до структурованої інформації.

Сутність Event представлена у Firestore як документ у колекції Events, який зберігає детальну інформацію про кожну створену подію. Унікальний ідентифікатор `uid` використовується як ключ до документа, забезпечуючи однозначне посилання на кожну подію в базі.

Кожен документ містить атрибути, що дозволяють повноцінно описати подію. Зокрема, визначено вікові обмеження (`ageRestriction`, `allowedAge`), інформацію про організатора (`createdby`), а також геолокаційні координати місця проведення (`location.lat`, `location.lon`). Це дає змогу фільтрувати події за релевантністю для певної категорії користувачів або за місцезнаходженням.

Часові параметри зберігаються у вкладеному об'єкті `dateinfo`, який містить дату події у форматі `timestamp`, а також час її початку та завершення. Такий підхід дозволяє легко впроваджувати календарні фільтри, сортування подій за датами, а також нагадування.

Візуальна складова реалізована через два об'єкти - `createdImg` та `image`, що містять дані про зображення події, включаючи URL та ідентифікатор файлу у сервісі Cloudinary. Це дає змогу зберігати як початкове, так і оновлене зображення, не перевантажуючи Firestore файлами, а використовуючи лише метадані.

Інші важливі поля включають:

- `description` - текстовий опис події;
- `parkingOption` - інформація про наявність паркування;
- `subscribedCount` - кількість користувачів, які підписалися на подію;
- `timebefore` та `timeBeforeType` - налаштування нагадування;
- `timestamp` - мітка часу створення або оновлення документа;
- `title` - назва події.

Ключовою особливістю такої структури є її гнучкість і розширюваність: у разі потреби до документа можна легко додати нові поля без зміни загальної логіки застосунку або структури бази. Структура документа Event зображена на рисунку 3.2.

Event	
uid	string
title	string
description	string
dateinfo_date	timestamp
dateinfo_start	string
dateinfo_end	string
timeBeforeType	string
timebefore	number
ageRestriction	string
allowedAge	number
parkingOption	string
subscribedCount	number
createdby	string
createdImg_file	string
createdImg_url	string
image_file	string
image_url	string
location_lat	number
location_lon	number
timestamp	timestamp

Риснок 3.2 – Діаграма структури сутності Event у Firestore

3.2.1.2 User (Користувач)

Опис структури сутності User (Користувач) у Firestore базі даних додатку Eventify відіграє ключову роль у моделюванні взаємодії користувача з подіями та системою загалом. Оскільки Firestore є NoSQL - документною базою, кожен користувач представлений як окремий документ у колекції Users, ідентифікований унікальним ключем uid.

Ця структура дозволяє:

- швидко отримати всю інформацію про користувача;
- легко підключити або змінити аватар через інтеграцію з Cloudinary;
- зберігати зв'язки між користувачем і подіями (через subscribedEvents);
- адаптувати функціональність профілю під індивідуальні потреби завдяки полю description.

Структура сутності User представлена у таблиці 3.1.

Таблиця 3.1 – Основні поля системи User

Поле	Тип даних	Опис
uid	string	Унікальний ідентифікатор користувача, що використовується також для авторизації через Firebase Authentication.
name	string	Ім'я користувача.
surname	string	Прізвище користувача.
username	string	Ім'я користувача у системі (може бути нікнеймом).
email	string	Email користувача, пов'язаний із його Firebase-акаунтом.
age	string	Вік користувача, може бути використано для вікових обмежень подій.
address	string	Адреса проживання або локація користувача.
description	string	Короткий опис профілю користувача.
avatar	map	Об'єкт, що містить інформацію про аватар:
→ file	string	Ідентифікатор зображення у Cloudinary.
→ url	string	URL до аватару в Cloudinary, який використовується у фронтенді.
subscribedEvents	array	Масив з uid подій, на які підписався користувач (зовнішнє посилання на колекцію Events).

Завдяки динамічній природі Firestore, до цієї структури можна безперешкодно додавати нові поля в майбутньому (наприклад, роль користувача, соціальні лінки, тощо), не змінюючи логіку застосунку.

User	
address	string
age	string
description	string
email	string
name	string
surname	string
username	string
uid	string
subscribedEvents	string[]
avatar_file	string
avatar_url	string

Рисунок 3.3 - Структура документа User

3.2.1.2 Notification (Сповіщення)

Колекція Notifications містить документи, що представляють сповіщення, які надсилаються користувачам (наприклад, нагадування про події). Кожен документ має унікальний ідентифікатор (id) та набір полів, які описують сповіщення. Структура документа Notification подана у таблиці 3.2.

Таблиця 3.2 – Основні поля системи Notification

Поле	Тип даних	Опис
id	string	Унікальний ідентифікатор сповіщення (використовується як ID документа у Firestore).
createdAt	timestamp	Час створення сповіщення. Дозволяє сортувати сповіщення хронологічно.
eventId	string	Ідентифікатор події, до якої належить сповіщення. Посилання на документ у колекції Events.
eventTitle	string	Назва події, що стосується сповіщення. Використовується для зручного відображення в інтерфейсі.
message	string	Текст сповіщення. Наприклад: “Нагадуємо, що подія розпочнеться за 30 хвилин”.
receiverId	string	Ідентифікатор користувача, якому надіслано сповіщення. Посилання на документ у колекції Users.
read	boolean	Статус прочитання: true – сповіщення переглянуто, false – ще ні.
type	string	Тип сповіщення. Наприклад: reminder, update, invitation, тощо. Це дозволяє фільтрувати сповіщення за категоріями.

Така структура вебзастосунку забезпечує:

- персоналізоване спілкування: кожне сповіщення прив’язане до конкретного користувача (receiverId);
- контекстуальність: прив’язка до події (eventId, eventTitle) дозволяє легко дізнатися, чого саме стосується повідомлення;
- гнучкість: завдяки полю type, система може адаптувати поведінку інтерфейсу - наприклад, по-іншому візуалізувати нагадування і системні сповіщення;
- моніторинг: поле read дає змогу реалізувати індикатори непрочитаних сповіщень.

Усі поля адаптовані до типового сценарію використання в мобільних додатках, і зберігають ефективність, простоту та масштабованість, притаманні NoSQL-системам.

Структура документа Notification зображена на рисунку 3.4.

Notification	
createdAt	timestamp
eventId	string
eventTitle	string
id	string
message	string
read	boolean
receiverId	string
type	string

Рисунок 3.4 – Структура Notification

3.2.2 Взаємозв'язки між сутностями

У застосунку Eventify структура бази даних, реалізована у Firestore, побудована таким чином, щоб забезпечити зручну взаємодію між ключовими сутностями: подіями (Event), користувачами (User) та сповіщеннями (Notification). Взаємозв'язки між цими сутностями формують логіку функціонування платформи, зокрема механізми створення, підписки на події, надсилання нагадувань і побудову інтерфейсів користувача.

Кожна подія створюється конкретним користувачем, і це відображається через поле createdBy, яке містить uid користувача. Таким чином, між Event і User існує прямий зв'язок, що дозволяє визначити організатора кожної події. Крім того,

користувачі можуть підписуватися на події - ця інформація зберігається у вигляді масиву `subscribedEvents` у документі користувача, що містить ідентифікатори подій. Завдяки цьому підхід дозволяє ефективно реалізувати функціональність персоналізованих підписок, адже система може легко визначити, які події цікавлять конкретного користувача. Зі свого боку, сама подія містить лічильник підписників (`subscribedCount`), який відображає кількість зацікавлених користувачів. Схема взаємозв'язків між сутностями зображена на рисунку 3.5.

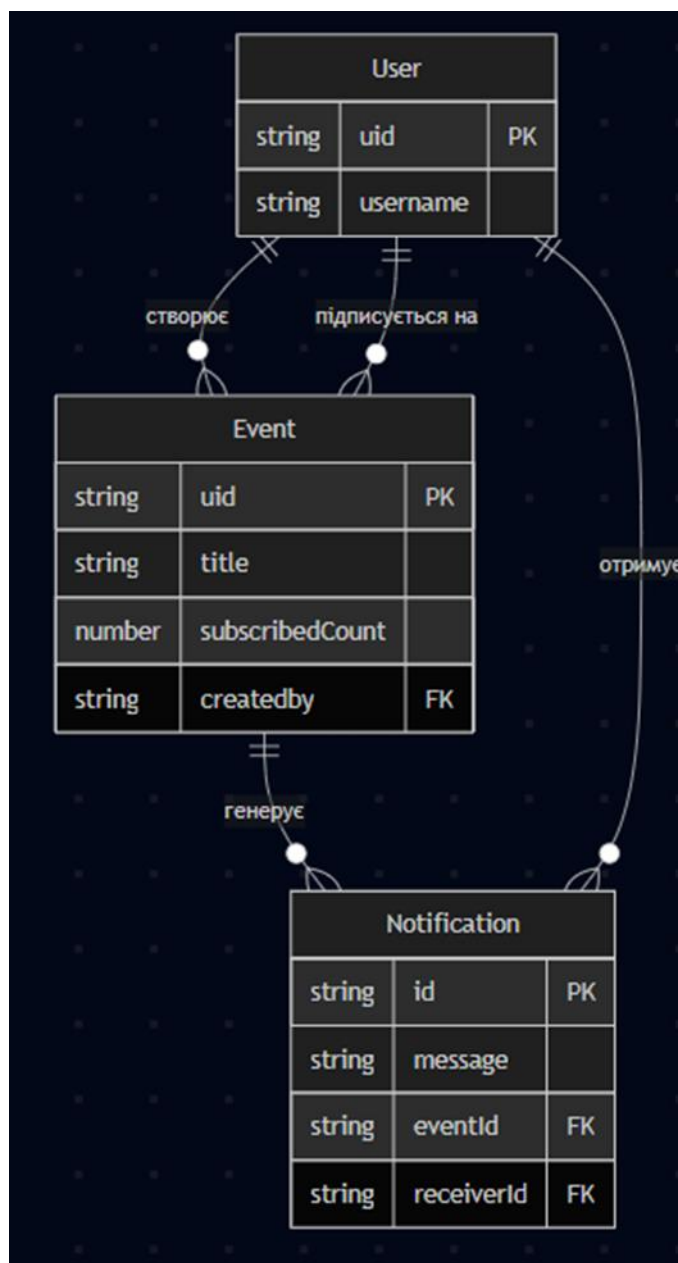


Рисунок 3.5 – Взаємодія сутностей

Сповіщення у базі даних пов'язані як із подіями, так і з користувачами. Кожне сповіщення (Notification) має поле `eventId`, що вказує на подію, до якої воно належить, та `receiverId`, що вказує на користувача, якому воно адресоване. Це дає змогу системі надсилати персоналізовані повідомлення, наприклад нагадування про майбутні події. Для зручності відображення в інтерфейсі мобільного або вебзастосунку, у Notification зберігається також назва події (`eventTitle`). Хоча це є денормалізацією даних (дублюванням), такий підхід дозволяє зменшити кількість додаткових запитів до бази під час формування списку сповіщень, що оптимізує продуктивність і затрати на читання.

Гнучка документно-орієнтована структура Firestore дозволяє легко масштабувати систему - додавати нові поля, нові типи сповіщень чи взаємодії між користувачами без необхідності змінювати загальну схему бази. Водночас, такий підхід зберігає логічну зв'язність між сутностями, що сприяє зручній навігації в даних і ефективному виконанню запитів на стороні клієнта.

3.3 Обґрунтування вибору СКБД

Для реалізації бази даних додатку Eventify було обрано Cloud Firestore від Firebase. Вибір СКБД є важливим етапом проєктування, оскільки від нього залежить продуктивність, масштабованість та зручність розробки.

Додаток Eventify має такі основні вимоги до системи керування базами даних:

- гнучка структура даних - додаток працює з подіями, користувачами та сповіщеннями, які можуть мати змінну кількість полів (наприклад, зображення, геолокація, часові налаштування);
- масштабованість - система повинна витримувати зростання кількості користувачів та подій без значних змін у архітектурі;

- реальний час - для сповіщень та оновлення списків подій важлива підтримка оновлень у реальному часі;
- простота інтеграції - оскільки бекенд відсутній, СКБД має легко інтегруватися з фронтендом на React;
- безпека - необхідна вбудована підтримка авторизації та правил доступу до даних;
- управління витратами - для стартапу важливо мінімізувати витрати на інфраструктуру, особливо на ранніх етапах.

Таблиця 3.3 - Порівняння СКБД

Характеристика	Cloud Firestore (Firebase)	MongoDB	PostgreSQL
Тип БД	NoSQL (документно-орієнтована, хмарна)	NoSQL (документно-орієнтована)	Реляційна (SQL)
Гнучкість структури даних	Висока, дозволяє додавати поля на ходу	Висока, аналогічно Firestore	Низька, строго задані схеми
Масштабованість	Автоматична, без серверів	Хороша, але потребує налаштування кластерів	Можлива, через реплікацію/шардінг вручну
Оновлення в реальному часі	Вбудовано	Потрібні сторонні інструменти (наприклад, Socket.IO)	Не підтримується нативно
Інтеграція з фронтендом (React)	Дуже проста, через Firebase SDK	Потребує бекенду або сторонніх бібліотек	Потребує бекенду
Підтримка авторизації	Через Firebase Authentication	Потрібно впроваджувати окремо	Потрібно налаштовувати зовнішньо
Безпека доступу до даних	Правила Firestore Security	Можна реалізувати через middleware	Використовує контроль доступу через SQL-ролі
Підтримка складних запитів	Обмежена, прості фільтрації	Завдяки агрегаціям	Повноцінний SQL
Інфраструктура	Serverless (непотрібен власний сервер)	Потребує окремого хостингу або серверу	Потребує окремого сервера
Залежність від екосистеми	Висока (Firebase/Google)	Низька, відкритий стандарт	Низька, відкритий стандарт

З аналізу таблиці 3.3. видно, що Cloud Firestore - оптимальне рішення для проєктів, орієнтованих на швидку розробку, інтеграцію з фронтендом та мобільними застосунками, де критичними є масштабованість і оновлення в

реальному часі. У свою чергу, MongoDB більше підходить для кастомних бекендів зі складною логікою агрегації, а PostgreSQL - для застосунків, де потрібні строгі зв'язки між таблицями та складні SQL-запити [12].

Для розробки застосунку Eventify було прийнято обґрунтоване рішення на користь використання Cloud Firestore, яке ґрунтується на кількох ключових перевагах цієї бази даних у контексті сучасного фронтенд-орієнтованого середовища.

По-перше, Firestore підтримує серверлес-архітектуру, що дозволяє уникнути створення окремого бекенду - фронтенд застосунок безпосередньо взаємодіє з базою даних через Firebase SDK. Це значно спрощує розробку, зменшує технічну складність проєкту та знижує витрати на інфраструктуру, що критично важливо на ранніх етапах запуску стартапу.

Другою важливою причиною стала гнучкість NoSQL-моделі, яка дозволяє легко працювати зі змінною структурою документів. У системі Eventify події можуть мати різні властивості - зображення, геолокацію, додаткові метадані - і Firestore без труднощів підтримує такі варіації без необхідності оновлення схеми бази.

Firestore також забезпечує оновлення в реальному часі, що дозволяє миттєво відображати зміни у списках подій та надсилати сповіщення без додаткових інструментів. Це особливо важливо для подієвого застосунку, де користувачі очікують актуальної інформації в режимі онлайн.

Інтеграція з екосистемою Firebase, зокрема з Firebase Authentication, значно полегшує реалізацію авторизації та ідентифікації користувачів. У поєднанні з потужними правилами безпеки Firestore можна точно контролювати, хто має доступ до яких документів - наприклад, гарантувати, що користувач може редагувати лише власні події.

Ще однією важливою причиною стала автоматична масштабованість, яку надає Firestore: база даних може легко обслуговувати як десятки, так і тисячі користувачів без втрати продуктивності, що ідеально підходить для проєктів з потенційним швидким ростом.

Firestore також зручно інтегрується з Cloudinary - сервісом для зберігання зображень. Замість збереження самих файлів, у Firestore зберігаються лише URL, що спрощує обробку медіа та економить ресурси.

Нарешті, економічна модель Firebase, яка базується на оплаті за фактичне використання, робить Firestore дуже привабливим вибором для стартапів: на початкових етапах розробки з невеликою кількістю активних користувачів витрати мінімальні, але система легко адаптується до зростання навантаження в майбутньому.

Cloud Firestore є оптимальним вибором для додатку Eventify завдяки своїй гнучкості, підтримці реального часу, простоті інтеграції та серверлес-підходу. У порівнянні з MongoDB та PostgreSQL, Firestore краще відповідає вимогам проєкту, особливо з урахуванням відсутності бекенду та потреби в швидкій розробці. Хоча Firestore має обмеження на складні запити, вони компенсуються денормалізацією даних (наприклад, дублюванням назви події у сповіщеннях), що дозволяє ефективно обробляти запити у контексті цього додатку.

3.4 Розробка інтерфейсу користувача

Інтерфейс користувача (UI) додатку Eventify розроблений з особливою увагою до потреб кінцевих користувачів, щоб зробити взаємодію з додатком максимально простою, приємною та ефективною. Основна мета розробки інтерфейсу - забезпечити інтуїтивний доступ до всіх ключових функцій додатку, таких як перегляд подій, підписка на них та управління профілем, при цьому зберігаючи приємний і сучасний візуальний стиль. Оскільки додаток побудований на React, інтерфейс є модульним і компонентно-орієнтованим, що дозволяє легко оновлювати окремі елементи без впливу на загальну структуру. У цьому розділі детально описано ключові екрани додатку, їхню функціональність, візуальні особливості та підхід до забезпечення зручності для користувачів.

Основні принципи розробки інтерфейсу:

- дизайн інтерфейсу створений таким чином, щоб навіть новий користувач, який вперше відкрив додаток, міг швидко розібратися в його функціональності без додаткових інструкцій. Кнопки та елементи управління розташовані там, де користувач інтуїтивно їх очікує;
- інтерфейс розроблений з урахуванням використання на різних пристроях - від великих екранів десктопів до компактних екранів мобільних телефонів. Це досягається завдяки респонсивному дизайну, який автоматично адаптує розташування елементів залежно від розміру екрана;
- візуальний стиль додатку дотримується принципів мінімалізму: використовується обмежена кольорова палітра (нейтральні тони з акцентами для кнопок), чіткі шрифти та лише необхідні елементи. Це допомагає уникнути перевантаження інтерфейсу та зосередити увагу користувача на головних функціях;
- користувач завжди отримує чіткий відгук про свої дії. Наприклад, після підписки на подію з'являється сповіщення типу toast, яке підтверджує успішність операції, або, у разі помилки, інформує про проблему (наприклад, "Ви вже підписані на цю подію");
- інтерфейс спроектований так, щоб користувач міг виконати основні дії (переглянути подію, підписатися, відредагувати профіль) за мінімальну кількість кліків, що економить час і підвищує зручність.

3.4.1 Ключові екрани інтерфейсу

3.4.1.1 Початкова сторінка

Початкова сторінка є першим, що бачить користувач, відкриваючи додаток Eventify, і відіграє роль "вітрини", яка запрошує до взаємодії. Для неавторизованих користувачів сторінка пропонує короткий опис додатку, підкреслюючи його основну цінність - можливість створювати, знаходити та приєднуватися до подій у кілька кліків. У верхній частині екрана розташована кнопка "Увійти" або "Зареєструватися", яка відкриває форму авторизації. Для авторизованих користувачів сторінка трансформується в більш персоналізовану: на ній відображаються рекомендовані події (на основі попередніх підписок або

геолокації), а також кнопка швидкого доступу до створення нової події. У верхній частині сторінки є навігаційна панель із посиланнями на основні розділи: усі події, профіль, створені події. Візуально сторінка оформлена з акцентом на простоту: фон із легким градієнтом, велике зображення або ілюстрація, що символізує події (наприклад, люди на фестивалі), і чіткий заклик до дії. Екран початкової сторінки зображений на рисунку 3.6.

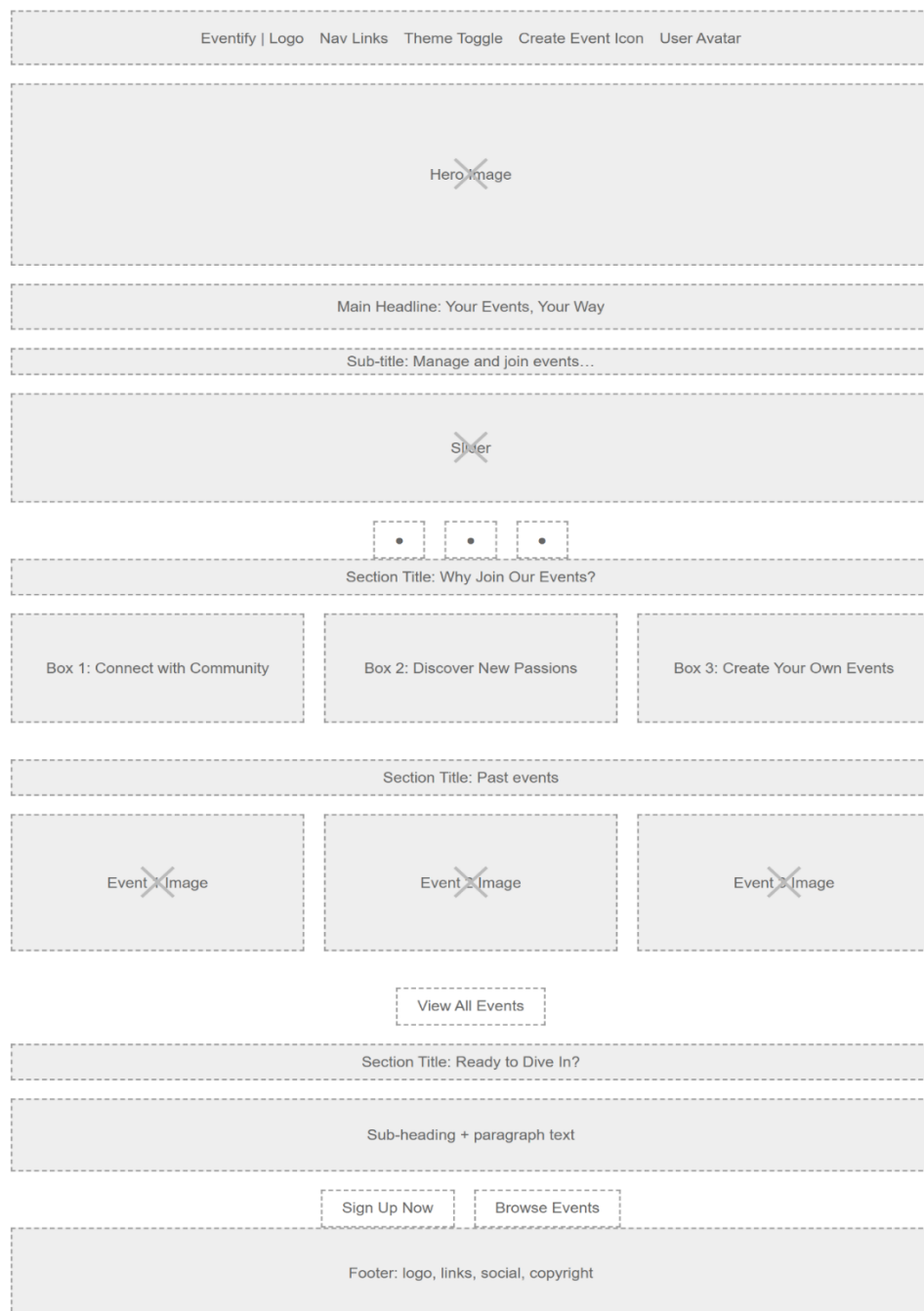


Рисунок 3.6 – Макет головної сторінки

3.4.1.2 Усі події (з сортуванням)

Екран усіх подій є центральним для користувачів, які хочуть знайти цікаві заходи, на які можна підписатися. Події відображаються у форматі карток, розташованих у сітці, що дозволяє одночасно переглядати кілька подій без необхідності прокручувати довгий список. Кожна картка включає основну інформацію: назву, дату, місце та зображення події. Для зручності користувача у верхній частині екрана розташована панель сортування та фільтрації. Фільтри дозволяють сортувати події за датою (найближчі, минулі), геолокацією (наприклад, події поблизу), віковими обмеженнями, а також за категоріями (наприклад, концерти, спортивні заходи). Є можливість пошуку за ключовими словами (наприклад, назва події або місце). При натисканні на картку користувач переходить до детального перегляду події. Візуально картки оформлені з акцентом на зображення: велике зображення займає верхню частину картки, а текстова інформація розташована нижче у компактному форматі. Кольорові акценти (наприклад, голуба кнопка "Details") допомагають швидко знайти ключові дії. Екран усіх подій із сортуванням зображений на рисунку 3.7.

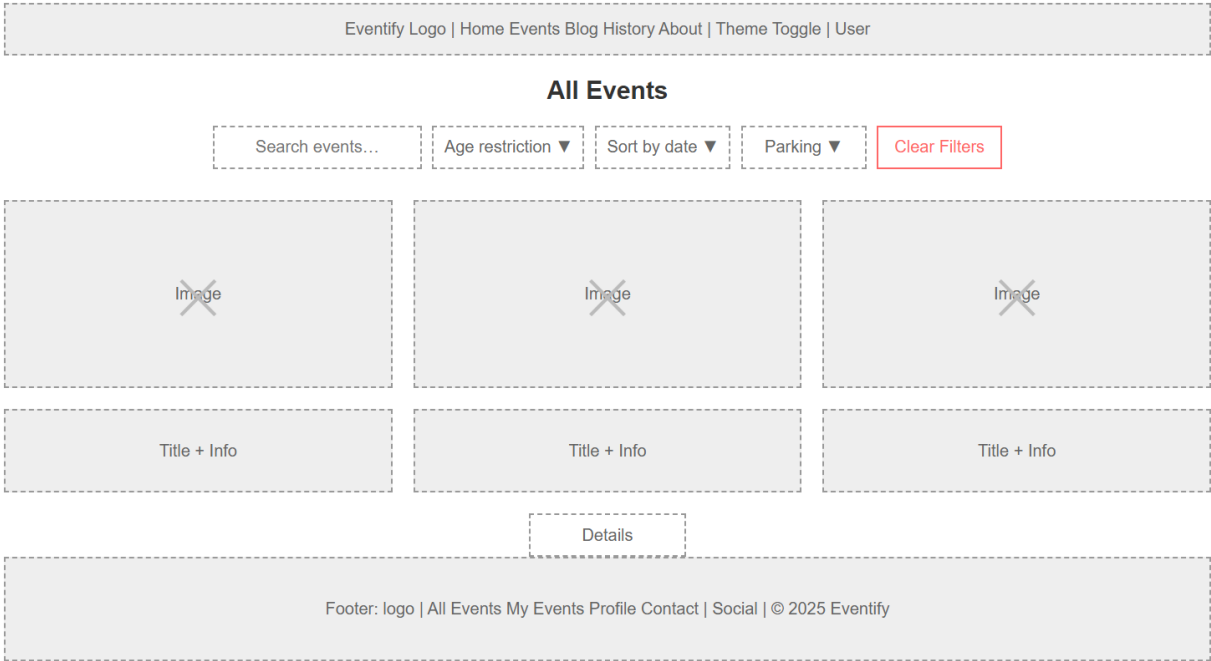


Рисунок 3.7 – Макет сторінки усіх подій

3.4.1.3 Окрема подія

Екран окремої події створений для того, щоб користувач міг отримати повну інформацію про подію та прийняти рішення про участь. На екрані представлена детальна інформація: назва, опис, дата, час початку та закінчення, місце проведення, вікові обмеження, інформація про паркування та організатора. Є окремий блок із кількістю підписників, що додає соціальний аспект (користувач бачить, наскільки подія популярна). Основна кнопка "Subscribe" розташована у нижній частині екрана для швидкого доступу; якщо користувач уже підписаний, кнопка змінюється на "Unsubscribe". Візуально екран оформлений так, щоб ключова інформація (дата, місце) була одразу помітною: велике зображення події розташоване у верхній частині, а текстова інформація структурована за допомогою заголовків та відступів. Екран окремої події зображений на рисунку 3.8.

Header: Logo • Nav Links • Theme Toggle • User Avatar

Event Title

Event Image

Event Description

Subscribers: •• Created by: ••

Date: ••/••/••••

Time: ••:•• — ••:••

Time Before: •• units

Location: City Name

Age Restriction: ••+

Parking: Option

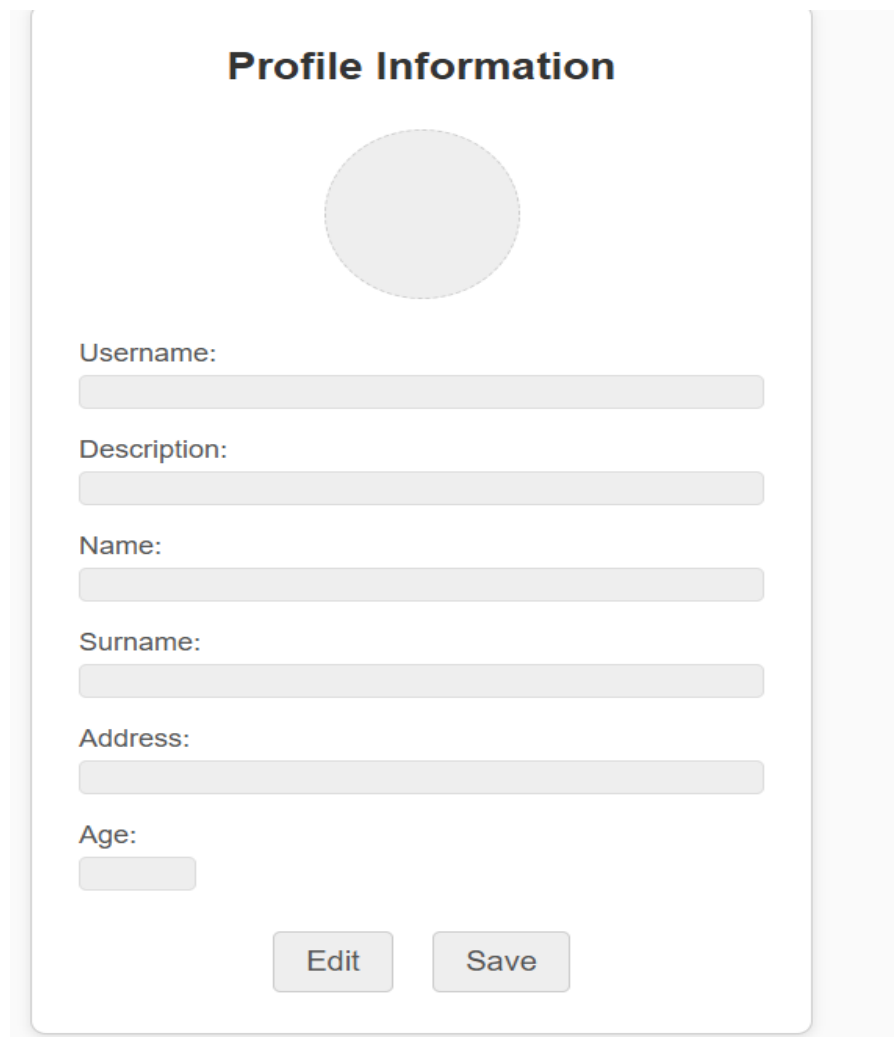
Subscribe

Footer: Logo • Links • Social Icons • © Year Eventify

Рисунок 3.8 - Макет окремої сторінки івента

3.4.1.4 Профіль користувача

Екран профілю є персональним простором користувача, де він може переглядати та редагувати свої дані, а також бачити свою активність у додатку. У верхній частині екрана відображається аватар користувача, ім'я, прізвище та короткий опис профілю. Нижче розташована інформація про вік, email та адресу, якщо користувач їх вказав. Є кнопка "Редагувати профіль", яка відкриває форму, де можна оновити дані, зокрема завантажити новий аватар через інтеграцію з Cloudinary. Завантаження аватара супроводжується попереднім переглядом, щоб користувач міг переконатися, що зображення виглядає коректно. Візуально профіль оформлений у світлих тонах із акцентами на кнопках. Аватар відображається у круглій рамці, а текстова інформація структурована для легкого сприйняття. Екран профілю користувача зображений на рисунку 3.9.



The image shows a user profile form titled "Profile Information". At the top, there is a large circular placeholder for a profile picture. Below this, the form contains several input fields for user data: "Username:", "Description:", "Name:", "Surname:", "Address:", and "Age:". Each label is followed by a corresponding text input field. At the bottom of the form, there are two buttons: "Edit" and "Save". The entire form is presented in a clean, light gray style with rounded corners.

Рисунок 3.9 – Макет профіля користувача

Інтерфейс застосунку Eventify створено з акцентом на зручність, доступність та сучасний дизайн, що забезпечує інтуїтивний користувацький досвід. Основний елемент - картки подій, які компактно відображають ключову інформацію (назву, дату, локацію, зображення) та дозволяють миттєво переглядати деталі або підписуватись. Такий формат добре масштабується для різних екранів.

Висока інтерактивність інтерфейсу проявляється через інтерактивну карту, попередній перегляд аватара та toast-сповіщення, які чітко повідомляють про результати дій без перевантаження екрану. Інтеграція з Cloudinary дає змогу зручно завантажувати зображення з попереднім переглядом [11].

Інтерфейс є адаптивним: сітка карток змінюється на вертикальний список на мобільних пристроях, а навігація - на бургер-меню. Також враховано доступність - усі елементи мають підписи, контрастні кольори й підходять для користувачів із особливими потребами.

Інтерфейс користувача додатку Eventify розроблений з акцентом на зручність, візуальну привабливість та швидкість виконання дій. Кожен екран створений так, щоб користувач міг легко орієнтуватися в додатку, швидко знаходити потрібну інформацію та виконувати основні дії з мінімальними зусиллями. Такий підхід забезпечує позитивний користувацький досвід і сприяє залученню користувачів до активного використання додатку.

3.5 Аналіз та вибір засобів розробки

Для розробки вебзастосунку «Eventify App» було необхідно вибрати оптимальні засоби, які б відповідали вимогам проєкту: швидкість розробки, продуктивність, масштабованість, зручність інтеграції та підтримка сучасних підходів до створення інтерфейсів.

Засоби розробки для Eventify було обрано з урахуванням ключових вимог, що забезпечують ефективну побудову сучасного вебзастосунку. Однією з основних

була швидкість розробки - платформа мала дозволяти швидко створювати компоненти інтерфейсу та інтегрувати їх із базою даних без зайвих ускладнень. У цьому контексті важливу роль відіграє також продуктивність: сторінки повинні завантажуватися швидко, а сам застосунок - бути оптимізованим для пошукових систем, що критично для публічних подій.

Ще однією умовою була масштабованість, тобто здатність проєкту легко адаптуватися до нових функцій або зростання кількості користувачів без суттєвої перебудови архітектури. Важливою також стала простота інтеграції з базами даних та зовнішніми сервісами, оскільки застосунок не передбачає окремого бекенду. Обрані інструменти повинні забезпечувати прямий та безпечний обмін даними на клієнтській стороні.

Крім того, важливим аспектом була зручність стилізації - можливість швидко створювати адаптивні, привабливі та доступні інтерфейси, які добре працюють на різних пристроях. І, нарешті, ефективне управління станом застосунку на клієнтській стороні є критичним для координації дій користувача, зберігання даних сесії та синхронізації з базою даних.

3.5.1 Аналіз засобів для клієнтської частини

Фреймворк для інтерфейсу.

Next.js. Фреймворк на основі React із підтримкою серверного рендерингу (SSR) та статичної генерації (SSG). Переваги: вбудована маршрутизація, оптимізація продуктивності, покращення SEO, підтримка API Routes для серверлес-функцій. Недоліки: складніша крива навчання порівняно з чистим React.

React із Create React App (CRA). Популярний інструмент для створення React-додатків. Переваги: простота налаштування, широка спільнота. Недоліки: відсутність вбудованого SSR/SSG, що ускладнює SEO та продуктивність. Обґрунтування вибору: Next.js обрано через підтримку SSR, що покращує швидкість завантаження та SEO, а також через вбудовану маршрутизацію, яка спрощує розробку [13].

Стилізація.

Tailwind CSS. Утилітарний CSS-фреймворк для швидкого створення адаптивного дизайну. Переваги: швидкість стилізації, адаптивність, мінімізація CSS-коду. Недоліки: початкова крива навчання через використання утилітарних класів.

CSS-модулі. Локальні стилі для React-компонентів. Переваги: ізоляція стилів, простота використання. Недоліки: повільніша розробка через написання CSS вручну, складніше масштабування. Обґрунтування вибору: Tailwind CSS обрано через швидкість стилізації та зручність створення адаптивного дизайну, що особливо важливо для проєкту з різними розмірами екранів [12].

Управління станом.

Redux Toolkit - бібліотека для централізованого управління станом із вбудованими інструментами для спрощення роботи з Redux. Переваги: централізоване управління, підтримка асинхронних запитів через RTK Query, інтеграція з Next.js. Недоліки: додатковий обсяг коду для простих додатків.

Zustand - легка бібліотека для управління станом. Переваги: простота, мінімалістичний синтаксис. Недоліки: менш потужна для складних асинхронних операцій порівняно з RTK Query. Обґрунтування вибору: Redux Toolkit обрано через підтримку RTK Query, що спрощує асинхронні запити до Firestore, та через зручність інтеграції з Next.js [11].

3.5.2 Аналіз засобів для інтеграцій

База даних та авторизація.

Firebase (Cloud Firestore та Authentication) - хмарна платформа для серверлес-архітектури. Переваги: проста інтеграція з фронтендом, реальний час, вбудована авторизація, автоматична масштабованість. Недоліки: залежність від екосистеми Google, обмеження на складні запити.

Робота із зображеннями.

Cloudinary - сервіс для завантаження, зберігання та обробки зображень. Переваги: зручний віджет для завантаження, автоматична оптимізація зображень, інтеграція з фронтендом. Недоліки: залежність від зовнішнього сервісу.

Firebase Storage - альтернатива для зберігання файлів. Переваги: інтеграція з екосистемою Firebase. Недоліки: відсутність вбудованої оптимізації зображень, складніша обробка. Обґрунтування вибору: Cloudinary обрано через зручний віджет для завантаження та автоматичну оптимізацію зображень, що спрощує роботу з медіафайлами.

Для розробки Eventify обрано такі засоби: Next.js для створення інтерфейсу, Tailwind CSS для стилізації, Redux Toolkit для управління станом, Firebase для бази даних та авторизації, а також Cloudinary для роботи із зображеннями. Цей вибір забезпечує швидкість розробки, продуктивність, масштабованість та зручність інтеграції. Next.js дозволяє оптимізувати додаток для SEO та швидкості, Tailwind CSS прискорює створення адаптивного дизайну, Redux Toolkit спрощує асинхронні запити, Firebase забезпечує серверлес-архітектуру, а Cloudinary оптимізує роботу із зображеннями. Ці інструменти відповідають вимогам проекту та створюють основу для подальшого розвитку.

4 РОЗРОБКА ВЕБЗАСТОСУНКУ

4.1 Розробка бази даних

База даних додатку Eventify реалізована з використанням Cloud Firestore - NoSQL бази даних від Firebase, яка ідеально підходить для серверлес-архітектури, оскільки додаток не має окремого бекенду. Firestore забезпечує гнучке зберігання даних у вигляді колекцій та документів, автоматичну масштабованість та підтримку оновлень у реальному часі. У цьому розділі описано процес створення структури бази даних, налаштування правил безпеки, індексів та особливості реалізації.

На етапі проектування (див. розділ 3.2) було визначено три основні сутності: Event (Подія), User (Користувач) та Notification (Сповіщення). Для кожної сутності створено окрему колекцію у Firestore: колекція Events, колекція Users, колекція Notifications.

Колекція Events містить документи, що представляють події, створені користувачами. Кожен документ має унікальний ідентифікатор (uid), який генерується автоматично Firestore під час створення події. Структура документа включає поля, визначені у розділі 3.2: ageRestriction, allowedAge, createdImg, createdby, dateinfo, description, image, location, parkingOption, subscribedCount, timeBeforeType, timebefore, timestamp, title, uid. Для створення документа у цій колекції фронтенд (React) використовує Firebase SDK, викликаючи метод addDoc для додавання нової події або setDoc для оновлення існуючої.

Колекція Users містить документи, що представляють користувачів додатку. Унікальний ідентифікатор (uid) кожного документа генерується Firebase Authentication під час реєстрації користувача і використовується як ID документа у Firestore. Структура документа включає поля: address, age, avatar, description, email, name, subscribedEvents, surname, uid, username. Документ створюється автоматично після реєстрації користувача через Firebase Authentication, а оновлення даних (наприклад, аватара чи адреси) виконується через фронтенд.

Колекція Notifications містить документи, що представляють сповіщення для користувачів (наприклад, нагадування про події). Кожен документ має унікальний ідентифікатор (id), який генерується Firestore. Структура документа включає поля: createdAt, eventId, eventTitle, id, message, read, receiverId, type. Сповіщення створюються автоматично, коли користувач підписується на подію, або за розкладом (наприклад, за день до події) через Firebase Cloud Functions, якщо потрібна серверна логіка.

У межах реалізації програмного забезпечення, що використовує хмарну базу даних Firebase Firestore, важливу роль відіграє захист інформації від несанкціонованого доступу та забезпечення правомірного поводження з даними. Для досягнення цього були налаштовані правила безпеки (Firestore Security Rules), які визначають рівень доступу користувачів до різних колекцій документів у базі даних.

Колекція Events, яка зберігає інформацію про події, налаштована таким чином, щоб забезпечити її загальнодоступність для авторизованих користувачів. Це

означає, що будь-який користувач, який увійшов до системи, може переглядати список подій. Однак створення нових документів дозволено виключно у разі, якщо поле `createdby` вказує на UID користувача, що виконує запит, тобто на автора події. Таким чином реалізується механізм верифікації авторства. Оновлення або видалення події можливе лише для її автора або для користувача з правами адміністратора, що забезпечує контроль за змінами та видаленням інформації.

У колекції `Users` кожен користувач має право доступу виключно до власного документа. Це обмеження діє як на читання, так і на оновлення даних, тим самим гарантується конфіденційність персональної інформації. Створення нових документів у цій колекції відбувається автоматично в момент реєстрації через `Firebase Authentication`, тому ручне додавання користувачів системою заборонено.

Колекція `Notifications` призначена для збереження сповіщень, адресованих конкретним користувачам. Правила доступу побудовані так, що прочитати сповіщення може лише той користувач, UID якого співпадає з полем `receiverId` документа. Створення нових сповіщень дозволено або системі (наприклад, через хмарні функції `Cloud Functions`), або авторизованим користувачам - зокрема, для цілей тестування. Оновлення можливе лише у випадку зміни статусу повідомлення - поля `read`, яке відображає факт прочитання. Дана дія дозволена лише безпосередньо адресатові повідомлення.

Ці правила забезпечують захист даних, дозволяючи користувачам бачити лише власні дані та доступні події, а також запобігають несанкціонованим змінам.

4.2 Налаштування індексів та оптимізація доступу до даних у Firestore

Для забезпечення високої продуктивності та ефективності при роботі з великими обсягами даних у базі `Firestore` було налаштовано систему індексів. Індеси у `Firestore` дозволяють значно пришвидшити виконання запитів, зокрема при фільтрації та сортуванні результатів. У процесі розробки додатку особливу

увагу було приділено оптимізації доступу до даних у колекціях Events та Notifications, які є ключовими для роботи з подіями та користувацькими сповіщеннями відповідно.

У колекції подій було реалізовано індексацію за кількома напрямками. По-перше, індекс на поле timestamp дозволяє сортувати події за часом їх створення, що є необхідним для відображення найновіших подій у стрічці. Додатково, індекс на вкладене поле dateinfo.date забезпечує можливість ефективного сортування подій за датою їхнього фактичного проведення. Для реалізації пошуку за місцем проведення події було створено складений індекс, що охоплює поля широти та довготи (location.lat і location.lon). Це дозволяє виконувати геопросторові запити з мінімальною затримкою, що особливо актуально для сценаріїв, де користувачі шукають події поблизу.

У колекції сповіщень також були налаштовані індекси, які підтримують швидке сортування та фільтрацію. Індекс на поле createdAt дає змогу впорядковувати сповіщення за часом їх створення, забезпечуючи хронологічне відображення нових повідомлень. Складений індекс на поля receiverId та read слугує для вибірки лише непрочитаних повідомлень, адресованих конкретному користувачу, що критично важливо для реалізації функціональності push-сповіщень та іконок із позначенням непрочитаних повідомлень у інтерфейсі користувача.

Налаштування індексів відбувалося через консоль Firebase, яка дозволяє як створювати прості індекси вручну, так і автоматично генерувати необхідні складені індекси в процесі розробки, коли Firestore виявляє складні запити та пропонує створити відповідну конфігурацію.

Серед важливих рішень при побудові структури бази даних варто відзначити денормалізацію частини інформації. Наприклад, у колекції Notifications дублюється назва події з колекції Events у вигляді окремого поля eventTitle. Це дозволяє відображати сповіщення у списку без необхідності додаткових запитів до іншої колекції, що помітно знижує затримку при завантаженні інтерфейсу.

Ще однією ключовою особливістю є підтримка оновлень у реальному часі. Використання функціоналу onSnapshot, який надає Firestore, дозволяє автоматично

відображати зміни у списках подій і сповіщень без потреби у ручному оновленні сторінки. Такий підхід забезпечує сучасний користувацький досвід і створює відчуття інтерактивності.

З метою оптимізації роботи з великими списками даних була реалізована пагінація. Завдяки використанню функцій `limit` і `startAfter` стало можливим поступове завантаження подій порціями, що не лише пришвидшує відображення контенту, але й зменшує навантаження на клієнтські пристрої.

Також у процесі розробки було впроваджено обробку можливих помилок при виконанні запитів. Наприклад, у разі втрати підключення до мережі або недоступності Firestore користувач отримує відповідне повідомлення, що дозволяє уникнути непорозумінь і покращує загальне враження від роботи з додатком.

Загалом, використання Cloud Firestore у додатку Eventify дало змогу реалізувати масштабовану, надійну й гнучку систему зберігання даних. Вбудована підтримка реального часу, зручна система індексації, гнучкі правила безпеки та прозора інтеграція з фронтендом стали ключовими факторами ефективної реалізації цього рішення.

4.3 Розробка клієнтської частини

Клієнтська частина додатку Eventify розроблена з використанням Next.js - фреймворку на основі React, який забезпечує серверний рендеринг (SSR), статичну генерацію (SSG) та маршрутизацію на основі файлів. Використання Next.js дозволяє покращити продуктивність додатку, оптимізувати SEO та спростити розробку завдяки вбудованим інструментам.

4.3.1 Налаштування проєкту

Розробка розпочалася з ініціалізації Next.js-проєкту за допомогою команди ``npx create-next-app``. Для стилізації використовувалася бібліотека Tailwind CSS, яка інтегрується через конфігураційний файл ``tailwind.config.js``, дозволяючи швидко

створювати адаптивний дизайн. Для роботи з Firebase було встановлено Firebase SDK, що забезпечує інтеграцію з Cloud Firestore та Firebase Authentication. Для управління станом додатку підключено Redux Toolkit разом із бібліотекою `'next-redux-wrapper'`, яка забезпечує коректну роботу Redux із серверним рендерингом у Next.js.

4.3.1.1 Розробка основних функціональних модулів

Клієнтська частина додатку складається з кількох ключових сторінок, які відповідають основним екранам (див. розділ 3.4): початкова сторінка, усі події, окрема подія та профіль користувача. Кожна сторінка реалізована як окремий файл у директорії `'pages'` у Next.js, що автоматично створює відповідний маршрут: початкова сторінка (`pages/index.js`) → усі події (`pages/events.js`) → окрема подія (`pages/events/[id].js`) → профіль користувача (`pages/profile.js`).

Початкова сторінка відображає статичний контент для неавторизованих користувачів (опис додатку, кнопки входу/реєстрації) та персоналізований контент для авторизованих (рекомендовані події). Для перевірки статусу авторизації використовується Firebase Authentication через контекст, який ініціалізується у файлі `'_app.js'`. Сторінка рендериться на сервері (SSR) за допомогою `'getServerSideProps'`, що дозволяє завантажувати рекомендовані події ще до відображення [13].

Сторінка `pages/events.js` для перегляду всіх подій завантажує дані з Firestore через `'getServerSideProps'`, що забезпечує швидке відображення початкового списку подій. Для пагінації та сортування (наприклад, за датою або геолокацією) додатково використовуються клієнтські запити через `'useEffect'`. Сортування реалізовано через Firestore-запити з індексами, а фільтрація виконується на клієнтській стороні після завантаження даних.

Динамічний маршрут для окремої події створюється за допомогою файлу `'[id].js'`. Дані події завантажуються через `'getServerSideProps'`, використовуючи параметр `'id'` для запиту до Firestore через `'getDoc'`. Для відображення геолокації використано бібліотеку React-Leaflet, яка рендерить інтерактивну карту на клієнтській стороні [14].

Сторінка профілю доступна лише авторизованим користувачам і завантажує дані з Firestore через ``getServerSideProps``. Для редагування профілю використовується форма, яка оновлює документ користувача через ``setDoc``. Завантаження аватара реалізовано через Cloudinary віджет.

Інтеграція з Firebase охоплює кілька аспектів:

- Firebase Authentication ініціалізується у файлі ``_app.js`` для глобального доступу до статусу авторизації. UID користувача зберігається у стані Redux;
- Firestore SDK використовується для запитів до колекцій Events, Users та Notifications [12]. Для оновлень у реальному часі (наприклад, сповіщення) застосовується слухач ``onSnapshot`` на клієнтській стороні;
- доступ до даних контролюється через правила Firestore (див. розділ 4.1).

Для завантаження зображень використано Cloudinary віджет ``CldUploadWidget``, який інтегровано у форми створення/редагування події та профілю. URL зображення, отриманий від Cloudinary, зберігається у відповідних полях документів Firestore (image для Events, avatar для Users). Для оптимізації зображень налаштовано автоматичне стиснення через Cloudinary.

Redux Toolkit із ``next-redux-wrapper`` використовується для управління станом. Основні стани включають дані користувача, список подій та сповіщення. Redux ініціалізується у файлі ``providers.tsx``, що забезпечує доступ до стану на всіх сторінках.

Особливості реалізації:

- використання ``getServerSideProps`` дозволяє завантажувати дані на сервері, покращуючи SEO та швидкість першого відображення;
- пагінація та кешування даних у Redux зменшують кількість запитів до Firestore;
- усі запити обробляються з відображенням повідомлень про помилки через сповіщення типу toast.

Розробка клієнтської частини на Next.js дозволила створити продуктивний та SEO-оптимізований додаток із швидким відображенням сторінок та ефективною інтеграцією з Firebase та Cloudinary.

4.4 Розробка інтерфейсу вебзастосунку

Інтерфейс вебзастосунку Eventify створений так, щоб користувачам було зручно та зрозуміло взаємодіяти з додатком. Усі сторінки розроблені з урахуванням простоти, інтуїтивності та адаптивності, щоб вони добре виглядали як на великих екранах, так і на мобільних пристроях.

4.4.1 Початкова сторінка

Цю сторінку я зробив як головну точку входу в додаток. Для користувачів, які ще не увійшли в систему, я додав короткий опис додатку та кнопки для входу або реєстрації, розташовані у верхній частині. Для тих, хто вже авторизований, я показую кілька рекомендованих подій, щоб одразу зацікавити користувача. Також зверху є панель навігації з посиланнями на інші розділи додатку, а на мобільних пристроях вона згортається в меню для економії місця. Екран початкової сторінки зображений на рисунку 4.1.

4.4.2 Усі події (з сортуванням)

На цій сторінці я розмістив список усіх доступних подій у вигляді карток. Кожна картка показує основну інформацію: назву, дату, місце та зображення. Зверху я додав панель для сортування та пошуку, щоб користувачі могли вибирати події за датою, місцем або шукати за ключовими словами. На кожній картці є кнопка для підписки на подію, яка змінюється на кнопку відписки після натискання. Якщо подій багато, я зробив так, щоб можна було підвантажувати наступні, натиснувши на кнопку внизу. На маленьких екранах картки вишиковуються одна під одною. Екран усіх подій із сортуванням зображений на рисунку 4.2.

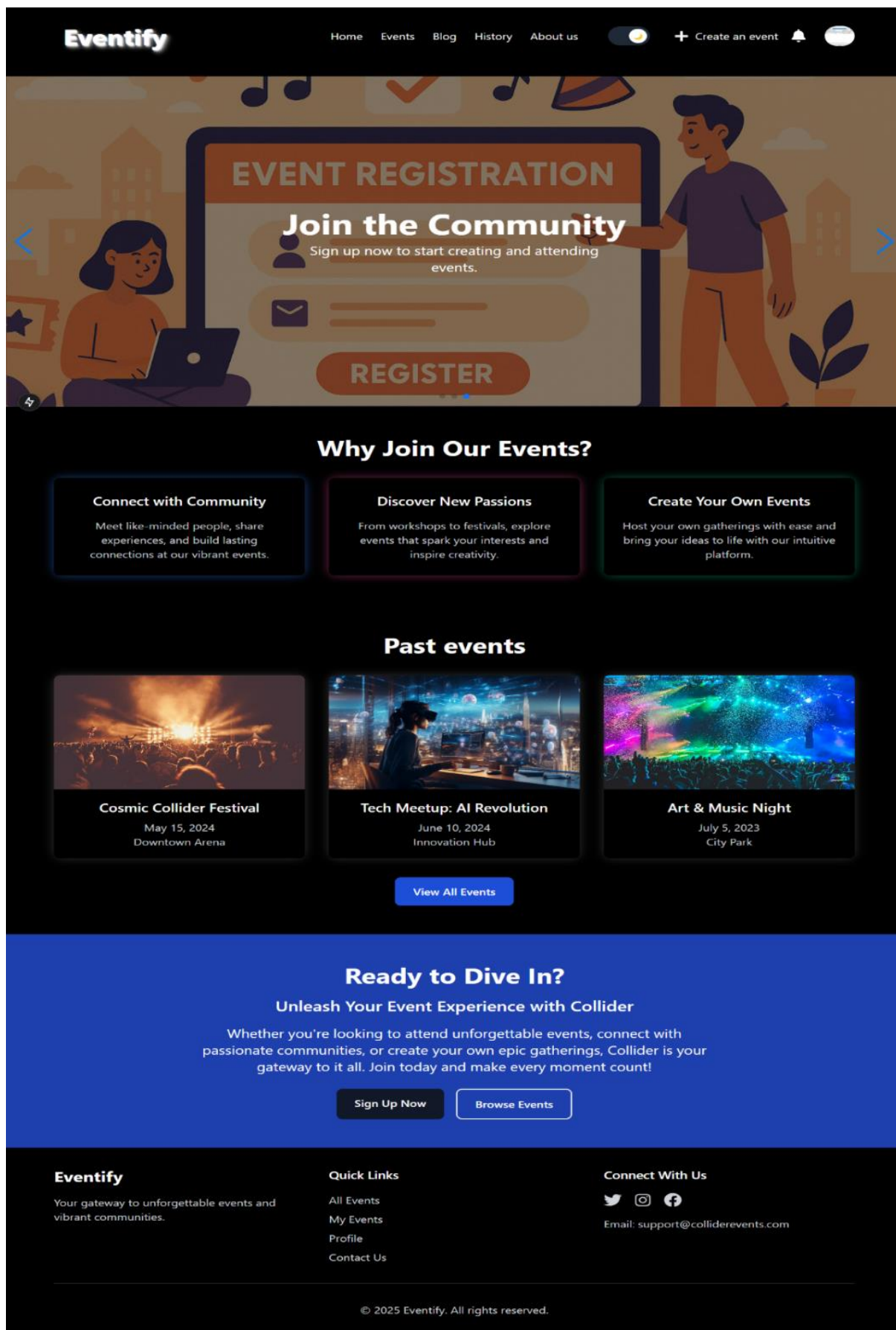


Рисунок 4.1 – Головна сторінка

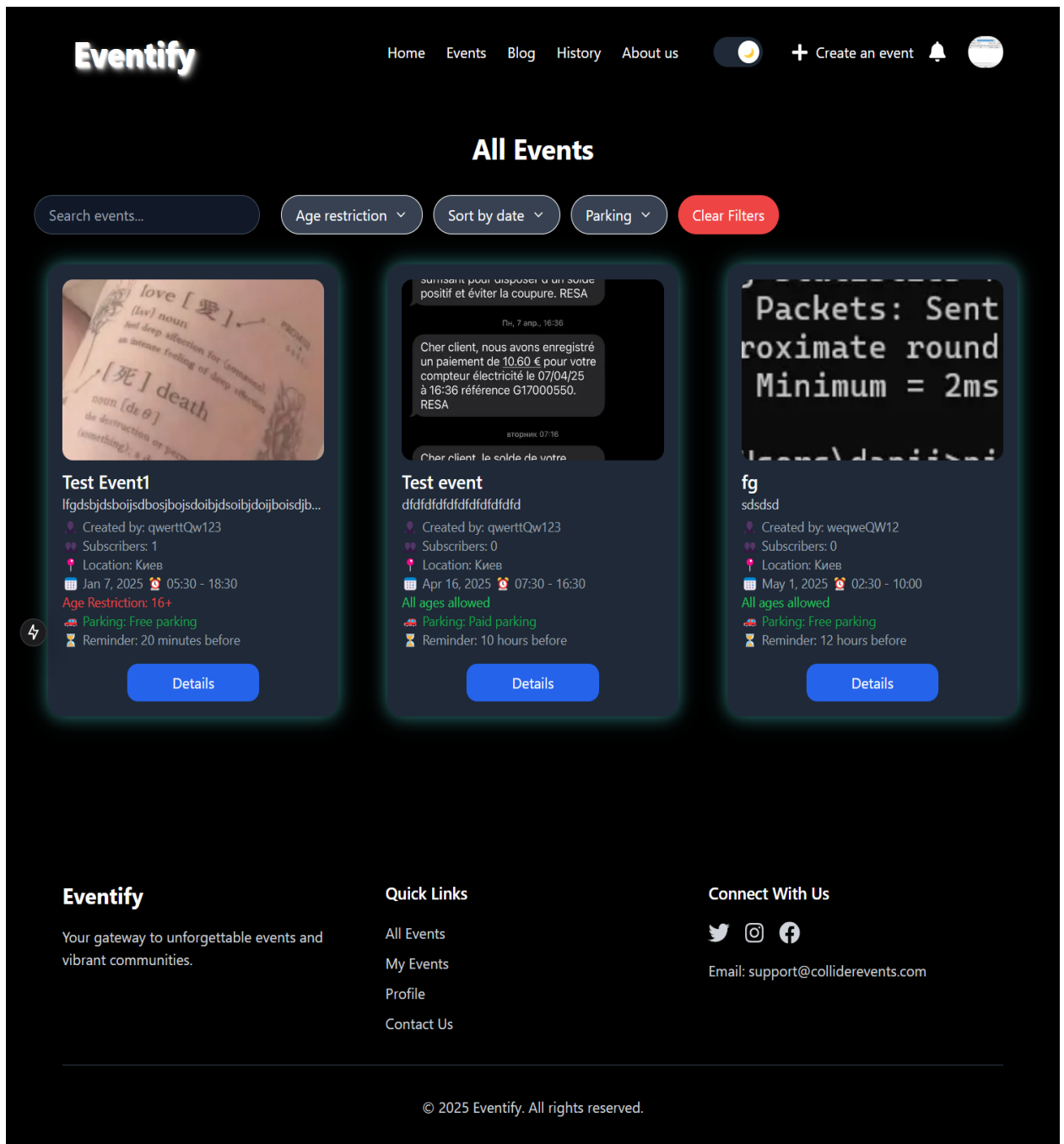


Рисунок 4.2 – Усі івенти

4.4.3 Окрема подія

Ця сторінка показує деталі про одну подію. Я розмістив велике зображення зверху, а під ним - усю інформацію: назву, опис, дату, час, місце, карту з геолокацією, вікові обмеження, наявність паркування та кількість підписників. Є кнопка для підписки, а якщо користувач створив цю подію, він побачить додаткові кнопки для редагування або видалення. Карта дозволяє наближати місце події, а на

маленьких екранах я зменшив її розмір, щоб усе вмістилося. Екран окремої події зображений на рисунку 4.3.

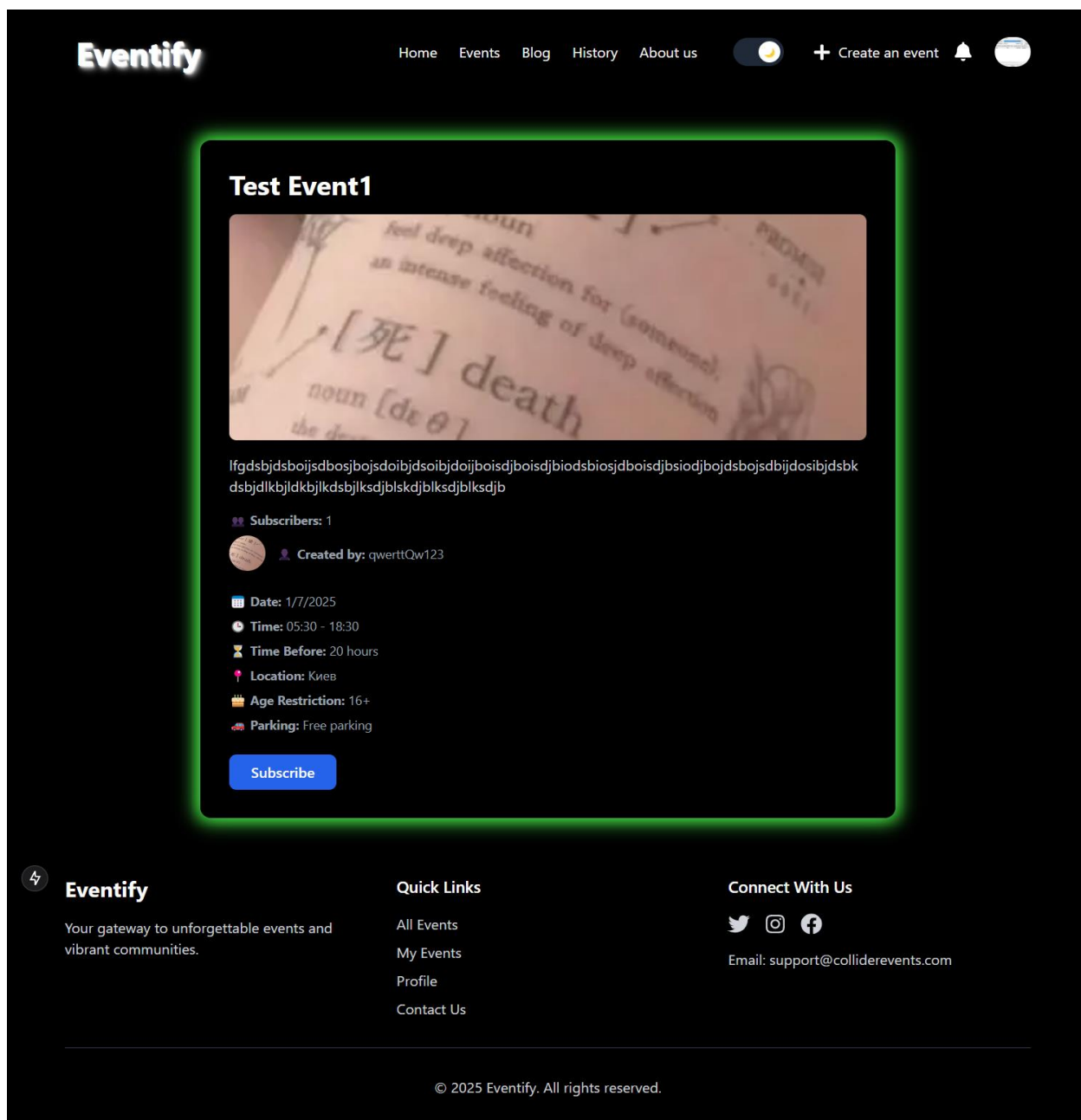


Рисунок 4.3 – Соло івент

4.4.4 Профіль користувача

На сторінці профілю я показую основну інформацію про користувача: аватар, ім'я, прізвище, e-mail, вік, адресу та опис. Є кнопка для редагування, яка відкриває форму, де можна змінити дані або завантажити новий аватар. Унизу я додав два списки: один із подіями, які користувач створив, а другий - із подіями, на які він

підписаний. Обидва списки можна прокручувати вбік, щоб швидко переглянути події. На мобільних пристроях форма редагування займає весь екран для зручності. Екран профілю користувача зображений на рисунку 4.4.

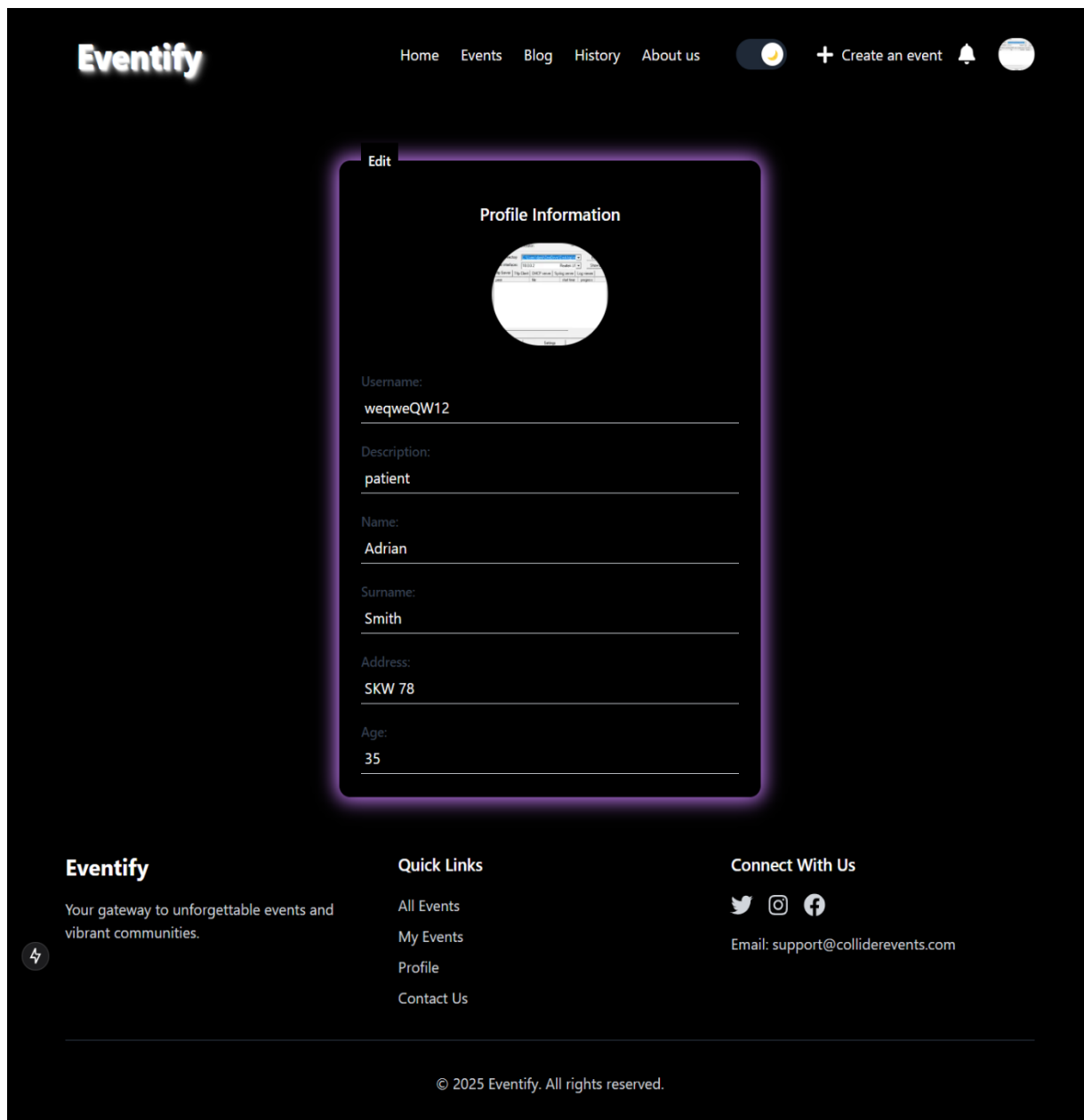


Рисунок 4.4 – Профіль користувача

Я зробив інтерфейс адаптивним, щоб він добре виглядав на різних пристроях. Для дій, які потребують підтвердження (наприклад, редагування профілю), я використав спливаючі вікна. Після кожної дії користувач бачить коротке сповіщення, наприклад, про успішну підписку на подію.

Інтерфейс додатку Eventify вийшов простим і зручним, щоб користувачі могли швидко знаходити потрібні функції та зручно працювати з додатком.

4.5 Опис функцій API

У цьому підрозділі наведено основні функції API, які використовуються у додатку Eventify для управління подіями та користувачами. Функції створені за допомогою Redux Toolkit Query та забезпечують взаємодію з Firebase Firestore та Firebase Authentication. Для кожної функції вказано її код та короткий опис призначення.

4.5.1 Функції управління подіями (eventApi)

Лістинг 4.1 - Функція deleteEvent

```
```typescript
deleteEvent: builder.mutation<null, string>({
 async queryFn(eventId) {
 try {
 const eventRef = doc(db, "events", eventId);
 await deleteDoc(eventRef);
 return { data: null };
 } catch (err: any) {
 return {
 error: {
 status: 500,
 data: err.message || "Failed to delete event",
 },
 };
 }
 },
 invalidatesTags: ["Events"],
}),
```
```

Функція видалення події у вебзастосунку реалізована для забезпечення можливості користувача керувати власним розкладом та редагувати вміст бази даних відповідно до змін потреб або обставин. Ця функція надає користувачу

інтерфейсний елемент (зазвичай у вигляді кнопки або контекстного меню), що дозволяє ініціювати процес видалення певної події. Вона видаляє документ із колекції `events`.

Лістинг 4.2 - Функція createEvent

```
```typescript
createEvent: builder.mutation<Event, Event>({
 async queryFn(eventData) {
 try {
 const eventsCollection = collection(db, "events");
 const newDocRef = doc(eventsCollection);
 const newEvent: Event = {
 ...eventData,
 uid: newDocRef.id,
 };
 await setDoc(newDocRef, {
 ...newEvent,
 timestamp: serverTimestamp(),
 });
 return { data: newEvent };
 } catch (err: any) {
 return {
 error: { status: 500, data: err.message || "Failed to
create event" },
 };
 }
 },
 invalidatesTags: ["Events"],
}),
```
```

Функція створення нової події є ключовим елементом вебзастосунку для планування, управління організації подій. Вона дозволяє користувачу додавати власні події до системи, вказуючи основні атрибути, такі як назва, дата, час, місце проведення, опис тощо. З технічного погляду функція забезпечує введення даних, їх валідацію, надсилання на сервер та збереження в базі даних. Вона додає подію до колекції `events` у Firestore, автоматично генеруючи унікальний ідентифікатор.

У вебзастосунку “Eventify App”, який використовує Firebase Firestore як базу даних, функція отримання списку всіх подій реалізується для виведення актуального переліку запланованих заходів користувача або всієї системи. Firestore

є NoSQL-базою, тому дані зберігаються у вигляді колекцій і документів, а запити реалізуються асинхронно.

Лістинг 4.3 - Функція fetchAllEvents

```

```typescript
fetchAllEvents: builder.query<Event[], void>({
 queryFn: () => ({ data: [] }),
 async onCacheEntryAdded(
 arg,
 { updateCachedData, cacheDataLoaded, cacheEntryRemoved },
) {
 try {
 await cacheDataLoaded;
 const eventsQuery = collection(db, "events");
 const unsubscribe = onSnapshot(eventsQuery, (snapshot) => {
 const eventsData = snapshot.docs.map((doc) => ({
 uid: doc.id,
 ...doc.data(),
 })) as Event[];
 updateCachedData(() => eventsData);
 });
 await cacheEntryRemoved;
 unsubscribe();
 } catch (error) {
 console.error("Error in fetchAllEvents:", error);
 }
 },
 providesTags: ["Events"],
}),
```

```

Функція використовується для завантаження даних із колекції events, що містить документи з інформацією про кожну подію: назва, дата, час, місце проведення, опис тощо. Дані можуть бути представлені у вигляді списку, календаря.

Вона працює в реальному часі, оновлюючи дані, коли змінюється колекція.

Наступною функцією є функція fetchUserEvents. Функція призначена для завантаження подій, створених конкретним користувачем. Вона повертає список подій, оновлюючи дані в реальному часі.

Функція завантаження подій відповідає за отримання та відображення актуального списку подій у вебінтерфейсі застосунку. Вона реалізована з

використанням хмарної бази даних Firebase Firestore та забезпечує асинхронну взаємодію з колекцією подій (events). Основна мета - відображення релевантної інформації для користувача в режимі реального часу або після відкриття застосунку.

Лістинг 4.4 - Функція fetchUserEvents

```

```typescript
fetchUserEvents: builder.query<Event[], string>({
 queryFn: () => ({ data: [] }),
 async onCacheEntryAdded(
 userId,
 { updateCachedData, cacheDataLoaded, cacheEntryRemoved },
) {
 try {
 await cacheDataLoaded;
 if (!userId) return;
 const eventsQuery = query(
 collection(db, "events"),
 where("createdby", "==", userId)
);
 const unsubscribe = onSnapshot(eventsQuery, (snapshot) => {
 const eventsData = snapshot.docs.map((doc) => ({
 uid: doc.id,
 ...doc.data(),
 })) as Event[];
 updateCachedData(() => eventsData);
 });
 await cacheEntryRemoved;
 unsubscribe();
 } catch (error) {
 console.error("Error in fetchUserEvents:", error);
 }
 },
 providesTags: ["Events"],
}),
```

```

Функція updateEvent дозволяє оновлювати дані існуючої події. Функція оновлення події дозволяє змінювати раніше створені записи в базі даних, наприклад, при редагуванні дати, часу, місця чи опису події. Ця функція є критично важливою для забезпечення актуальності даних у вебзастосунку організації подій та зручності користувачів. Вона змінює документ у Firestore, додаючи оновлений час.

Лістинг 4.5 - Функція updateEvent

```

```typescript
updateEvent: builder.mutation<Event, Event>({
 async queryFn(eventData) {
 try {
 const eventRef = doc(db, "events", eventData.uid);
 await setDoc(
 eventRef,
 {
 ...eventData,
 timestamp: serverTimestamp(),
 },
 { merge: true }
);
 return { data: eventData };
 } catch (err: any) {
 return {
 error: { status: 500, data: err.message || "Failed to
update event" },
 };
 }
 },
 invalidatesTags: ["Events"],
}),
```

```

4.5.2 Функції управління користувачами (userApi)

Лістинг 4.6 - Функція deleteUser

```

```typescript
deleteUser: builder.mutation<void, string>({
 async queryFn(uid) {
 try {
 const userRef = doc(db, "users", uid);
 await deleteDoc(userRef);
 return { data: undefined };
 } catch (error: any) {
 return { error: { status: 500, data: error.message } };
 }
 },
}),
```

```

Функція призначена для видалення користувача. Вона видаляє документ із колекції `users`.

Лістинг 4.7 - Функція logout

```
```typescript
logout: builder.mutation<null, void>({
 queryFn: () => ({ data: null }),
 async onCacheEntryAdded(_, { cacheEntryRemoved }) {
 try {
 auth.signOut();
 localStorage.clear();
 useUserStore.getState().setUser(null);
 await cacheEntryRemoved;
 } catch (error) {
 console.error("Logout failed:", error);
 }
 },
}),
```
```

Функція відповідає за вихід користувача з системи. Вона завершує сесію та очищає локальні дані.

Лістинг 4.8 - Функція updateUser

```
```typescript
updateUser: builder.mutation<
 AppUser,
 { uid: string; data: Partial<AppUser> }
>({
 async queryFn({ uid, data }) {
 try {
 const userRef = doc(db, "users", uid);
 await updateDoc(userRef, data);
 const updatedUserDoc = await getDoc(userRef);
 if (!updatedUserDoc.exists()) {
 throw new Error("User not found after update");
 }
 const updatedUser = updatedUserDoc.data() as AppUser;
 return { data: updatedUser };
 } catch (error: any) {
 return { error: { status: 500, data: error.message } };
 }
 },
}),
```
```

Функція дозволяє оновлювати профіль користувача. Вона оновлює документ у Firestore.

Лістинг 4.9 - Функція registration

```

```typescript
registration: builder.mutation<AppUser | null, Registerform>({
 async queryFn(data) {
 try {
 const userCredential = await
createUserWithEmailAndPassword(
 auth,
 data.email,
 data.password,
);
 const user = userCredential.user;
 const userData: AppUser = {
 uid: user.uid,
 email: user.email || "",
 username: data.username,
 description: "",
 avatar: { file: "", url: "" },
 name: "",
 surname: "",
 address: "",
 age: null,
 };
 await setDoc(doc(db, "users", user.uid), userData);
 return { data: userData };
 } catch (error) {
 const errorMessage =
 (error as Error).message || "An unknown error
occurred";
 return { error: { status: 500, data: errorMessage } };
 }
 },
}),
```

```

Функція призначена для реєстрації нового користувача. Вона створює обліковий запис і додає дані до Firestore.

Ці функції забезпечують основну логіку взаємодії фронтенду з базою даних та авторизацією у додатку Eventify.

ВИСНОВКИ

У процесі розробки вебзастосунку Eventify було створено функціональний та зручний інструмент для організації, пошуку та підписки на події. Основна мета проєкту - забезпечити користувачам простий і швидкий доступ до створення та перегляду подій - була досягнута завдяки ретельному плануванню, проєктуванню та реалізації.

Для розробки використано сучасний стек технологій. Клієнтська частина створена на базі Next.js, що дозволило оптимізувати продуктивність додатку, покращити SEO та спростити маршрутизацію. Cloud Firestore від Firebase використано як основну базу даних, яка забезпечила гнучке зберігання даних, автоматичну масштабованість та оновлення в реальному часі. Firebase Authentication застосовано для авторизації користувачів, а правила безпеки Firestore допомогли захистити дані. Для роботи із зображеннями інтегровано Cloudinary, що спростило завантаження та управління медіафайлами. Управління станом додатку реалізовано через Redux Toolkit, що забезпечило централізовану та ефективну обробку даних.

У рамках проєкту розроблено зручний інтерфейс із ключовими сторінками: початкова сторінка, перегляд усіх подій із сортуванням, окрема подія та профіль користувача. Інтерфейс створений адаптивним, інтуїтивним та орієнтованим на користувача, що дозволяє зручно працювати з додатком на різних пристроях. Реалізація API для управління подіями та користувачами через Redux Toolkit Query забезпечила ефективну взаємодію з базою даних, включаючи створення, оновлення, видалення подій та обробку авторизації.

Результатом роботи став вебзастосунок Eventify, який дозволяє користувачам легко створювати події, переглядати доступні заходи, підписуватися на них та керувати своїм профілем. Додаток відповідає поставленим вимогам, забезпечуючи швидкість, безпеку та зручність використання.

У майбутньому проєкт можна розвивати, додавши нові функції, такі як інтеграція із соціальними мережами для поширення подій, система рекомендацій на основі вподобань користувача або можливість створення групових чатів для учасників подій. Також можна покращити продуктивність шляхом оптимізації запитів до Firestore та додавання офлайн-підтримки для роботи без доступу до мережі.

Розробка Eventify стала цінним досвідом, що дозволив поглибити знання у створенні сучасних вебзастосунків із використанням серверлес-архітектури та NoSQL баз даних.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

- 1 Гуменюк, І. В. Івент-менеджмент: організація подій та свят: навч. посіб. [Текст] / І. В. Гуменюк. – Київ: Центр учбової літератури, 2020. – 248 с.
- 2 Дрозд, В. П. Менеджмент в індустрії подій [Текст]: навч. посіб. / В. П. Дрозд. – Львів: ЛРІДУ НАДУ, 2019. – 216 с.
- 3 Пархоменко, О. А. Івент-агенції в Україні: тенденції розвитку та маркетингові стратегії просування [Текст] / О. А. Пархоменко // Економіка і суспільство. – 2020. – № 22. – С. 118–122.
- 4 Product: Event Management Software. Bizzabo. URL: <https://www.bizzabo.com/event-management-software> (date of access: 28.04.2025).
- 5 Ковальчук, А. П. Сучасні технології розробки веб-застосунків [Текст]: навч.-метод. посіб. / Ковальчук А. П., Савчук О. М. – Вінниця: ВНТУ, 2022. – 168 с.
- 6 Бондар, В. І. Проектування інформаційних систем [Текст]: навч. посіб. / В. І. Бондар. – Київ : Ліра-К, 2020. – 312 с.
- 7 Фіалко, В. Ю. Основи проектування баз даних [Текст]: навч. посіб. / В. Ю. Фіалко. – Тернопіль : ТНТУ, 2019. – 248 с.
- 8 Глушаков, І. В. Основи веб-програмування [Текст]: навч. посіб. / І. В. Глушаков. – Харків : ХНУРЕ, 2018. – 210 с.
- 9 Анісімов, А. В. Інформаційні системи та бази даних [Текст]: навчальний посібник для студентів факультету комп'ютерних наук та кібернетики / А. В. Анісімов, П. П. Кулябко. – Київ. – 2017. – 110 с.
- 10 Ситник, О. В. Розробка веб-застосунків із використанням React і Node.js [Текст]: монографія / О. В. Ситник. – Львів : ЛНУ, 2021. – 184 с.
- 11 Прамодкумар, Дж. Садаладж NoSQL: новая методология разработки нереляционных баз данных [Текст] / Дж. Садаладж Прамодкумар, Мартин Фаулер. – Диалектика-Вильямс, 2017. - 192 с. - ISBN978-5-8459-1920-5
- 12 Томашевський, О. М. Інформаційні технології та моделювання бізнес-процесів [Текст]: навчальний посібник / О. М. Томашевський, Г. Г. Цетелик,

М. Б. Вітер, В. І. Дубук. - К.: «Центр Учбової літератури», 2012. - 296 с. – ISBN 978-617-673-003-3

13 Дакетт, Джон HTML и CSS. Разработка и создание веб-сайтов [Текст] / Джон Дакетт. - Эксмо, 2016. - 480 с. - ISBN - 978-5-699-64193-2

14 Тиленс Томас, Марк React в действии [Текст] / Марк Тиленс Томас. - Питер, 2018. - 368 с. - ISBN - 5446109996, 9785446109999

ДОДАТОК А

ЛІСТИНГИ ПРОГРАМ

Лістинг А.1 - Функція login

```

```typescript
login: builder.mutation<AppUser, { email: string; password:
string }>({
 async queryFn({ email, password }) {
 try {
 const userCredential = await signInWithEmailAndPassword(
 auth,
 email,
 password,
);
 const user = userCredential.user;
 const userDoc = await getDoc(doc(db, "users", user.uid));
 if (userDoc.exists()) {
 const userData = { ...user, ...userDoc.data() } as
AppUser;

 useUserStore.getState().setUser(userData);
 useUserStore.getState().setInitialized(true);
 const accessToken = await user.getIdToken();
 Cookies.set("accessToken", accessToken, {
 sameSite: "strict",
 expires: 1,
 });
 }
 return { data: userData };
 } else {
 return {
 error: {
 status: 404,
 data: {
 message: "User document not found",
 code: "USER_NOT_FOUND",
 },
 },
 };
 }
 }
} catch (error) {
 let errorMessage = "An unknown error occurred";
 if (error instanceof FirebaseError) {
 switch (error.code) {
 case "auth/user-not-found":
 errorMessage = "User not found";
 break;
 case "auth/wrong-password":
 errorMessage = "Incorrect password";
 break;
 }
 }
}
}

```

```

 default:
 errorMessage = error.message;
 }
 }
 return {
 error: {
 status: 500,
 data: {
 message: errorMessage,
 code: "LOGIN_ERROR",
 },
 },
 };
}
},
)),
),

```

Функція відповідає за авторизацію користувача. Вона перевіряє email та пароль через Firebase Authentication та повертає дані користувача.

Слайд 1

# Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій  
Кафедра комп'ютерні системи та мережі



## РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ ОРГАНІЗАЦІЇ ПОДІЙ ТА ВЗАЄМОДІЇ КОРИСТУВАЧІВ “EVENTIFY APP”

виконав студент 4 курсу  
групи КНТ-512сп  
ІВАНЧЕНКО Данііл  
керівник КУДЕРМЕТОВ Р.К.

2025

Слайд 2

**Метою** даної роботи є проєктування та реалізація вебзастосунку, що відповідатиме вимогам користувачів щодо зручності організації подій, взаємодії між учасниками та ефективної модерації контенту

Для досягнення мети необхідно виконати **наступні завдання** :

- проаналізувати предметну область і визначити актуальність розробки;
- дослідити наявні аналоги веб -сайтів у цій галузі, визначити їхні переваги та недоліки;
- обрати оптимальні веб-технології для реалізації функціоналу;
- спроектувати архітектуру, структуру та модель взаємодії користувачів.;
- реалізувати структуру у вигляді повноцінного веб -ресурсу;
- розробити функціонал, що забезпечить повноцінну роботу івент-агентства;
- провести тестування готового продукту та оцінити його ефективність.

Слайд 3

## Eventify App: новий погляд на організацію подій

У сучасному цифровому світі організація подій все більше переходить у віртуальний простір. Швидкий обмін інформацією, зручна реєстрація та ефективна комунікація стали ключовими факторами успіху. Дипломна робота присвячена розробці веб-застосунку "Eventify App", який покликаний спростити створення, управління та просування подій, а також забезпечити інтерактивну взаємодію між користувачами

**Актуальність теми** обумовлена зростаючим попитом на інструменти, що автоматизують процеси івент-менеджменту та підвищують залученість учасників. "Eventify App" прагне поєднати зручність, масштабованість та інтеграцію сучасних технологій, відповідаючи вимогам цифрового суспільства та ринку подієвих послуг.

Слайд 4

## Аналіз аналогів: Meetup, Bizzabo, Party.Space

### Meetup

Глобальна платформа для створення та пошуку спільнот за інтересами. Переваги: глобальне охоплення, інтуїтивний інтерфейс, мобільний застосунок. Недоліки: платна підписка для організаторів, відсутність українського інтерфейсу, обмежена популярність в Україні

### Bizzabo

SaaS-платформа для гібридних, віртуальних та офлайн-подій, орієнтована на бізнес-сегмент. Переваги: широка функціональність, інтуїтивний інтерфейс, розширена аналітика. Недоліки: складність інтеграції, висока вартість, орієнтація на корпоративний сегмент

### Party.Space

Український веб-застосунок для віртуальних святкових подій. Переваги: соціальна взаємодія в реальному часі, сильний емоційний ефект, гейміфікація. Недоліки: технічна складність, фокус виключно на онлайн, відсутність традиційних елементів планування

Дослідження наявних рішень на ринку веб-застосунків для організації подій дозволяє виявити можливості для удосконалення власного продукту. Аналіз функціональних можливостей, технологічної основи, UX/UI, цільового сегмента ринку та відгуків користувачів є ключовим для розробки інноваційного та якісного продукту

## Слайд 5

## Основні функції системи Eventify App

- ❖ Авторизація та реєстрація - безпечний вхід та створення нового облікового запису
- ❖ Особистий кабінет - перегляд та управління зареєстрованими подіями, персональними даними;
- ❖ Перегляд та пошук подій - доступ до майбутніх подій, фільтрація за критеріями;
- ❖ Створення та управління подіями - для організаторів: публікація, редагування та відстеження реєстрацій.

Веб-застосунок Eventify App реалізує ключові функціональні можливості, що забезпечують повний цикл взаємодії користувача з платформою

Система Eventify App передбачає три основні типи користувачів, кожен з яких має власні функціональні потреби та модель взаємодії. Розуміння цих ролей є ключовим для побудови надійної, масштабованої та інтуїтивно зрозумілої системи, що забезпечує баланс між функціональністю, безпекою та зручністю використання

- Звичайні користувачі (Учасники) - шукають та відвідують події. Потреби: простий інтерфейс, ефективний пошук, швидка реєстрація, доступ до актуальної інформації

- Організатори подій - створюють та керують подіями. Потреби: розширені можливості керування, інструменти комунікації з учасниками, аналітичні інструменти.

- Адміністратори системи - відповідальні за загальне управління платформою. Потреби: повний доступ до функціоналу, управління ролями, контроль за даними та безпекою.

## Слайд 6

## Роль моделювання в розробці «Eventify App»

### ❖ Формалізація вимог

Діаграми прецедентів відображають взаємодію користувачів із системою, уточнюючи функціональні сценарії

### ❖ Оптимальна архітектура

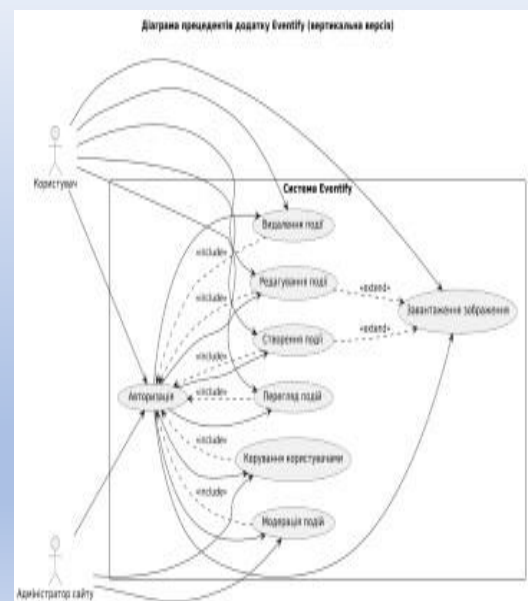
Структурне та поведінкове моделювання допомагає вибрати найкращу архітектуру фронтенду

### ❖ Верифікація та тестування

Діаграми активності виявляють "вузькі місця" та запобігають помилкам до написання коду

### ❖ Комунікаційний інструмент

Моделі забезпечують прозорість структури Eventify для всіх учасників команди



Слайд 7

## Архітектура вебсервісу Eventify App

Архітектура Eventify базується на серверлес-підході, де клієнтська частина взаємодіє напряму з Firebase. Це спрощує розробку, забезпечує масштабованість та легку інтеграцію із зовнішніми сервісами, такими як Cloudinary для завантаження зображень

### Клієнтська частина (Frontend)

- ✓ Реалізована на React.js для інтерактивного інтерфейсу
- ✓ Використовує Redux Toolkit для управління станом
- ✓ Інтегрується з Cloudinary для завантаження зображень
- ✓ Напряму взаємодіє з Firebase для запитів та даних

### Firebase як серверлес-платформа

- ✓ Замінює традиційний бекенд, пришвидшуючи розгортання
- ✓ Firebase Authentication для авторизації користувачів
- ✓ Firestore для зберігання даних у реальному часі
- ✓ Firebase Functions для опціональної серверлес-логіки

Слайд 8

## Проектування Cloud Firestore: сутності та структура

Cloud Firestore є основною NoSQL базою даних Eventify, що гнучко структурує дані у вигляді колекцій та документів. Це забезпечує масштабованість та зручність доступу до інформації про події, користувачів та сповіщення

### Event (Подія)

Документ у колекції Events з детальною інформацією: назва, опис, дата, час, місце, організатор, вікові обмеження, геолокація, URL зображень з Cloudinary та кількість підписників

### User (Користувач)

Документ у колекції Users з інформацією про користувача: ім'я, прізвище, email, вік, адреса, опис профілю, URL аватара та масив ідентифікаторів подій, на які підписано

### Notification

(Сповіщення)  
Документ у колекції Notifications: час створення, ID події, назва події, текст сповіщення, ID отримувача, статус прочитання та тип сповіщення

## Слайд 9

## Обґрунтування вибору СКБД: Cloud Firestore

Вибір Cloud Firestore для Eventify обґрунтований його відповідністю ключовим вимогам проєкту: гнучкість, масштабованість, оновлення в реальному часі, простота інтеграції та безпека

Таблиця – Порівняння СКБД

Характеристика	Cloud Firestore (Firebase)	MongoDB	PostgreSQL
Тип БД	NoSQL (документно-орієнтована, хмарна)	NoSQL (документно-орієнтована)	Реляційна (SQL)
Гнучкість структури даних	Висока, дозволяє додавати поля на ходу	Висока, аналогічно Firestore	Низька, строго задані схеми
Масштабованість	Автоматична, без серверів	Хороша, але потребує налаштування кластерів	Можлива, через реплікацію/шардінг вручну
Оновлення в реальному часі	Вбудовано	Потрібні сторонні інструменти (наприклад, Socket.IO)	Не підтримується нативно
Інтеграція з фронтендом (React)	Дуже проста, через Firebase SDK	Потребує бекенду або сторонніх бібліотек	Потребує бекенду
Підтримка авторизації	Через Firebase Authentication	Потрібно впроваджувати окремо	Потрібно налаштовувати зовнішньо
Безпека доступу до даних	Правила Firestore Security	Можна реалізувати через middleware	Використовує контроль доступу через SQL-ролі
Підтримка складних запитів	Обмежена, прості фільтрації	Завдяки агрегаціям	Повноцінний SQL
Інфраструктура	Serverless (непотрібен власний сервер)	Потребує окремого хостингу або серверу	Потребує окремого сервера
Залежність від екосистеми	Висока (Firestore/Google)	Низька, відкритий стандарт	Низька, відкритий стандарт

## Слайд 10

## Розробка інтерфейсу користувача та засоби розробки

Інтерфейс Eventify розроблений з акцентом на інтуїтивність, адаптивність та сучасний візуальний стиль. Вибір інструментів забезпечує швидкість розробки та високу продуктивність

- Next.js (Фреймворк для інтерфейсу)  
Обрано за підтримку серверного рендерингу (SSR) для покращення SEO та швидкості завантаження, а також за вбудовану маршрутизацію
- Tailwind CSS (Стилізація)  
Використано для швидкого створення адаптивного дизайну та мінімізації CSS -коду, що критично для різних розмірів екранів
- Redux Toolkit (Управління станом)  
Забезпечує централізоване управління станом та спрощує асинхронні запити до Firestore через RTK Query
- Cloudinary (Робота із зображеннями)  
Обрано за зручний віджет для завантаження та автоматичну оптимізацію зображень, що спрощує обробку медіафайлів

Слайд 11

## Розробка бази даних: Cloud Firestore

База даних Eventify реалізована за допомогою Cloud Firestore, NoSQL бази даних від Firebase, ідеальної для серверлес-архітектури. Вона забезпечує гнучке зберігання даних у колекціях та документах, автоматичну масштабованість та оновлення в реальному часі

Основні сутності

- ❖ Events: Події, створені користувачами, з унікальними ідентифікаторами та детальною інформацією
- ❖ Users: Користувачі додатку, з даними, що генеруються Firebase Authentication
- ❖ Notifications Сповіщення для користувачів, що створюються автоматично або за розкладом

Для захисту інформації налаштовані правила безпеки Firestore, що визначають рівень доступу до колекцій. Колекція Events загальнодоступна для перегляду авторизованими користувачами, але створення, оновлення та видалення подій дозволено лише авторам або адміністраторам. Колекція Users дозволяє доступ лише до власного документа, а Notifications доступні лише адресатам

Слайд 12

## Налаштування індексів та оптимізація доступу

Для забезпечення високої продуктивності та ефективності роботи з великими обсягами даних у Firestore налаштовано систему індексів. Індеси значно пришвидшують виконання запитів, особливо при фільтрації та сортуванні результатів

### Індексація подій

Індеси на поле timestamp та dateinfo.date дозволяють сортувати події за часом створення та проведення. Складений індекс на location.lat і location.lon забезпечує ефективні геопросторові запити.

### Денормалізація даних

Дублювання eventTitle у колекції Notifications зменшує кількість запитів, пришвидшуючи завантаження інтерфейсу.

### Індексація сповіщень

Індекс на createdAt для хронологічного відображення. Складений індекс на receiverId та read для вибірки непрочитаних повідомлень конкретному користувачу.

### Оновлення в реальному часі

Використання onSnapshot дозволяє автоматично відображати зміни у списках подій та сповіщень, забезпечуючи сучасний користувацький досвід.

Реалізована пагінація з використанням limit та startAfter для поступового завантаження подій, що зменшує навантаження на пристрої. Також впроваджено обробку помилок при виконанні запитів, покращуючи загальне враження від роботи з додатком

## Слайд 13

## Розробка клієнтської частини та інтерфейсу

Клієнтська частина «Eventify App» розроблена на Next.js, що забезпечує серверний рендеринг (SSR), статичну генерацію (SSG) та маршрутизацію на основі файлів. Це покращує продуктивність та SEO. Для стилізації використовується Tailwind CSS, а для управління станом – Redux Toolkit

### ❖ Початкова сторінка

Головна точка входу, що відображає опис додатку та кнопки входу/реєстрації для неавторизованих, або рекомендовані події для авторизованих користувачів

### ❖ Усі події

Список подій у вигляді карток з інформацією, панеллю сортування та пошуку. Підтримка пагінації та адаптивний дизайн

### ❖ Окрема подія

Детальна інформація про подію, включаючи зображення, опис, дату, час, місце, карту з геолокацією та кнопки підписки/редагування/видалення

### ❖ Профіль користувача

Основна інформація про користувача, можливість редагування, списки створених та підписаних подій. Адаптивний дизайн для мобільних пристроїв

Інтеграція з Firebase охоплює Firebase Authentication та Firestore SDK. Для завантаження зображень використовується Cloudinary. Застосунок забезпечує швидке відображення сторінок та ефективну взаємодію з усіма сервісами

## Слайд 14

## Висновки

Вебзастосунок «Eventify App» успішно реалізовано як функціональний та зручний інструмент для організації, пошуку та підписки на події. Основна мета - забезпечення простого та швидкого доступу до створення та перегляду подій - досягнута завдяки ретельному плануванню та використанню сучасного стеку технологій

Eventify App відповідає поставленим вимогам, забезпечуючи швидкість, безпеку та зручність використання. У майбутньому проєкт можна розвивати, додаючи інтеграцію із соціальними мережами, систему рекомендацій або групові чати, а також оптимізувати продуктивність та додати офлайн-підтримку.

### Використані технології

- Next.js для клієнтської частини
- Cloud Firestore як основна база даних
- Firebase Authentication для авторизації
- Cloudinary для управління зображеннями
- Redux Toolkit для управління станом

### Ключові функції та інтерфейс

- Створення та перегляд подій
- Підписка на заходи
- Управління профілем користувача
- Адаптивний та інтуїтивний інтерфейс
- Ефективна взаємодія з базою даних через API