МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

Кафедра інформаційних систем

МАГІСТЕРСЬКА РОБОТА

на тему: «РОЗРОБКА СИСТЕМИ АВТОМАТИЗОВАНОГО ПЛАНУВАННЯ ЧАСУ ТА МІСЦЯ ПРОВЕДЕННЯ ПОДІЙ»

Виконав: студент групи ПМІМ-22,
спеціальності 122 "Комп'ютерні науки",
Кушка А.В

Керівник: доктор фізико-математичних наук,
_____ Венгерський П.С.

Рецензент: д.т.н., професор
кафедри кібербезпеки
_____ Винокурова О.А.

Національна шкала _____

Кількість балів: _____

Оцінка: ECTS _____

Львів – 2022

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

Кафедра інформаційних систем

Спеціальність 122 "Комп'ютерні науки"

«ЗАТВЕРДЖУЮ»
Завідувач кафедри
проф. Шинкаренко Т.А.
"05" 09 **2022 року**

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Кушці Андрію Васильовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: "Розробка системи автоматизованого планування часу та місця проведення подій"

керівник роботи Венгерський Петро Сергійович, доктор фіз.-мат. наук, професор,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені Вченою радою факультету від "___" _____ 2022 року № _____

2. Строк подання студентом роботи "12" грудня 2022 року

3. Вихідні дані до роботи
   - набір типових подій різної тривалості, періодичності повторення та місця проведення;
   - набір тестових користувачів, з різними іменами, електронними поштами та домашніми адресами;
   - профіль у системі Google для інтеграції з сервісами Google Карти та Google Календар.

4. Зміст магістерської роботи (перелік питань, які потрібно розробити)
   - огляд аналогів;
   - алгоритми автоматичного планування часу та місця проведення подій;
   - вебзастосунок для планування подій як вручну, так і автоматично;
   - синхронізація подій з додатком Google Календар.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
   - зовнішній вигляд програм-аналогів;
   - ілюстрації до опису алгоритмів (процес "відсікання" подіями вільних часових проміжків протягом періоду автопланування часу, геометрична медіана та центр мас для автоматичного пошуку місця події);
   - зовнішній вигляд розробленого вебзастосунку (усі наявні сторінки, процес використання основних функцій системи, в.т.ч. автоматичне планування подій).

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання: "05" вересня 2022 року

Керівник роботи _____ **Венгерський П.С.**
(підпис) (прізвище та ініціали)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз ринку, пошук аналогів, їх переваг та недоліків	12.09.2022	
2	Пошук та огляд літератури	19.09.2022	
3	Розробка алгоритмів необхідних для виконання роботи	3.10.2022	
4	Планування архітектури проєкту, визначення переліку необхідних інструментів	10.10.2022	
5	Дизайн основних сторінок сайту, розробка головної сторінки, профілю й автентифікації користувачів	17.10.2022	
6	Розробка користувацького інтерфейсу для ручного створення, редагування та перегляду деталей подій	31.10.2022	
7	Реалізація розроблених алгоритмів автоматичного планування часу та місця проведення подій	14.11.2022	
8	Розробка додаткових функцій: синхронізації з гугл календарем, вигляд на різних пристроях тощо	21.11.2022	
9	Написання тексту роботи	28.11.2022	
10	Виправлення зауважень наукового керівника	05.12.2022	
11	Підготовка доповіді на захист	08.12.2022	

Студент:
_____ **Кушка А.В.**
(підпис) (прізвище та ініціали)

ЗМІСТ

ВСТУП 2

1 ОПИС ТА ПОСТАНОВКА ЗАДАЧІ 4
1.1 Основні вимоги до системи 4
1.2 Архітектура системи 5

2 ДОСЛІДЖЕННЯ НАЯВНИХ ЗАСТОСУНКІВ 6
2.1 Google Календар 6
2.2 Microsoft Outlook Calendar 7
2.3 Microsoft Project 8

3 ОПИС АЛГОРИТМІВ 10
3.1 Автоматичне планування часу звичайної події 10
3.2 Автоматичне планування часу події, що повторюється 12
3.3 Автоматичне планування місця події 13

4 ВИКОРИСТАНІ ТЕХНОЛОГІЇ 16
4.1 Клієнтська частина 16
4.2 Серверна частина 16
4.3 Хостинг 17
4.4 Сервіси Google 17

5 ОПИС РОБОТИ ПРОГРАМИ 18
5.1 Опис основних можливостей системи 18
5.2 Планування місця події 25
5.3 Часова шкала та планування часу події 28
5.4 Автоматичне планування часу 30
5.5 Планування часу подій, що повторюються 35
5.6 Додаткові властивості системи 40

ВИСНОВКИ 47

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 48

Додаток А. Код функції автоматичного планування часу простої події 50

Додаток Б. Код функції автоматичного планування часу події, що повторюється 52

Додаток В. Код функції автоматичного планування місця події 54

Додаток Г. Код функції перевірки наявності повторення події в межах вибраного дня 56

ВСТУП

У сучасному технологічному світі кожному з нас доводиться виконувати безліч справ з різних сфер життя протягом обмеженого відрізка часу [1]. Для того щоб не заплутатися і грамотно розставити завдання, потрібно складати план: у паперовому блокноті, або ж використовувати комп’ютерні програми-планери [2]. Такі програми можна умовно поділити на дві категорії: застосунки для планування особистого часу користувача й професійні бізнес-системи для планування проєктів.

До першої категорії належать різноманітні списки завдань та календарі. Усі вони дозволяють певною мірою впорядкувати завдання і навіть визначити точні часові рамки. Але, на жаль, таким застосункам часто бракує функціоналу, наприклад автоматизованого планування завдань чи зустрічей [3]. Щодо другої категорії, то там функціоналу значно більше. Такі програми дозволяють планувати проєкти по днях, тижнях, місяцях, визначати відповідальних людей для кожного завдання, з’ясовувати взаємозв’язок між завданнями. Деякі з них навіть містять певне автоматизоване планування. Але призначаються схожі застосунки саме для бізнесу і не придатні для особистого використання. До того ж вони дорогі, а отже не доступні широкому загалу.

Проте, ми не можемо стверджувати, що нікому, крім менеджерів великих проєктів, такі програми не потрібні. Адже навіть під час навчання в університеті нам необхідно знаходити час як для самостійного вивчення різних предметів, інколи в один день, так і для командних проєктів, до яких залучаються студенти зі різними розпорядками дня. Тому на сьогодні існує потреба у застосунку для особистого використання, який би міг автоматично запланувати завдання користувача, або ж зустріч декількох користувачів відповідно до їх розкладів справ. У цьому й полягає **актуальність** цієї роботи.

Відповідно **метою** роботи є дослідження наявних застосунків з ручним і автоматичним плануванням часу і розробка власної програми, яка б дозволяла користувачам доручати планування окремих завдань комп'ютеру і була доступна для широкого загалу.

1 ОПИС ТА ПОСТАНОВКА ЗАДАЧІ

Основна **задача** цієї роботи полягає в тому, щоб розробити застосунок, який повинен дозволити користувачеві планувати свої завдання та події, налаштовувати час подій як вручну, так і автоматично, а також запрошувати на них інших користувачів.

1.1 Основні вимоги до системи

Система повинна мати такі функції:

- створення і видалення власних подій;
- перегляд подій, на часовій шкалі;
- перегляд списку подій;
- перегляд і редагування деталей власної події: назва, опис, посилання на вебзустріч, час початку, час закінчення, тривалість, повторюваність події, координати, адресу та опис місця проведення;
- планування подій, що повторюються з різною періодичністю (наприклад щодня, по буднях, щотижня, в останню неділю місяця тощо);
- можливість редагувати окремі повторення події;
- автоматичне планування часу події з урахуванням її тривалості та встановлених параметрів;
- автоматичне планування часу події, що повторюється з урахуванням графіку користувачів у дні повторення події (редагування окремих повторень події у такі дні);
- автоматичний пошук адреси та опису місця проведення події за вибраною точкою на карті;
- автоматичний пошук місця проведення події враховуючи домашні адреси усіх учасників події;
- додавання та видалення учасників;
- перегляд списку учасників;
- перегляд списку подій, на які користувача запрошено;

- можливість приховати список учасників від учасників власної події;
- позначення власної події як виконаної;
- перегляд інформації про подію, на яку користувача запрошено;
- можливість бачити чи може учасник відвідати подію;
- скасування участі у події.

Додаткові функції системи

- адаптивність вебзастосунку до розміру екрана користувача (монітор, ноутбук, планшет, телефон);
- налаштування мови сайту (українська / англійська);
- налаштування формату часу (12 годин / 24 години);
- налаштування двофакторної авторизації за допомогою одного з наявних мобільних застосунків (наприклад Google Authenticator, DUO);
- інтеграція з застосунком Google Календар (імпорт та експорт подій).

1.2 Архітектура системи

Система повинна бути побудована у формі веб-застосунку, який складається з трьох основних частин:

- **клієнтська частина** зі зручним користувацьким інтерфейсом, яка працює в браузері;
- **серверна частина,** яка опрацьовує дані, отримані від користувача, і надає клієнтській частині дані, які потрібно відобразити на сайті (профіль користувача, список подій і налаштування);
- **база даних**, що містить дані користувачів і подій з їх налаштуваннями та списками учасників.

Веб-застосунок повинен бути у відкритому доступі, для того, щоб люди з різних куточків планети мали змогу ним користуватися. Для цього потрібно використати можливості хмарних сервісів для хостингу вебсайтів.

2 ДОСЛІДЖЕННЯ НАЯВНИХ ЗАСТОСУНКІВ

Як було сказано у вступній частині, існує два типи програм-планерів. Одні призначені для особистого використання, а інші – для бізнесу. Розглянемо найвідоміших представників з кожної категорії.

2.1 Google Календар

Найпопулярнішою програмою-планером для особистого використання є Google Календар [4]. Цей застосунок надає користувачу зручний інтерфейс для ручного планування завдань по днях (при перегляді розкладу на місяць) або по годинах (у випадку перегляду справ на тиждень чи день).

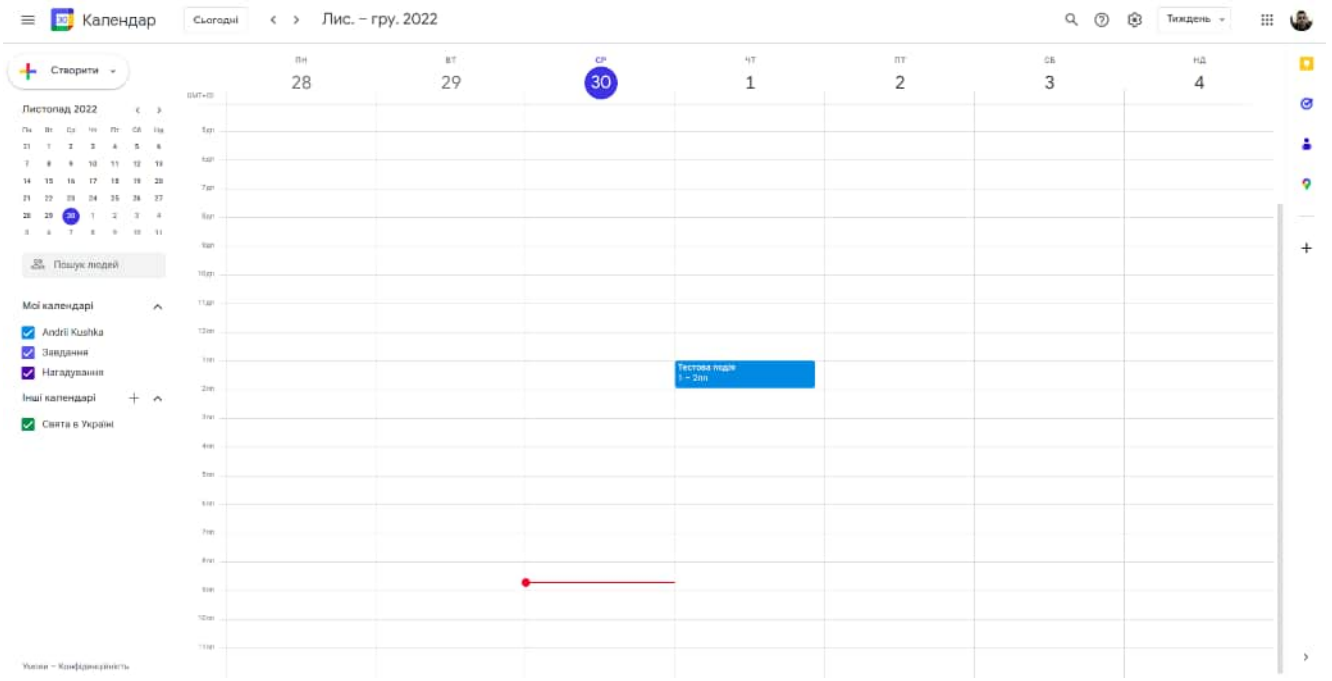

Рисунок 2.1 – Інтерфейс програми «Google Календар»

Для створення завдання достатньо протягнути вказівником миші від часу початку до часу завершення справи/зустрічі. До зустрічі можна додати учасників, а також подивитись розклад тих користувачів, які дали на це дозвіл [4].

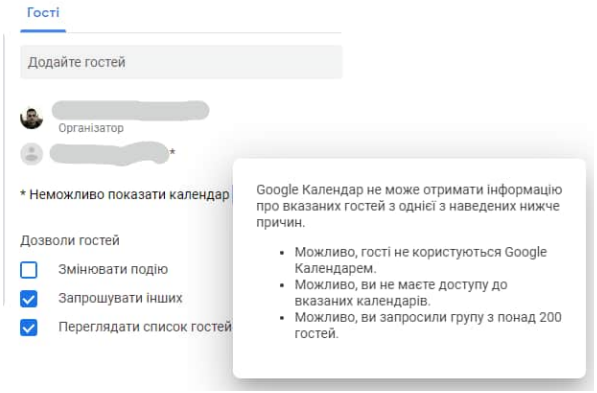

Рисунок 2.2 – Інтерфейс додавання гостей до зустрічі

Як можна бачити на рисунку 2.2 у програмі Google Календар існує функція перегляду розкладів гостей, яка покликана спростити користувачу вибір часу зустрічі, проте має певні обмеження у зв'язку з конфіденційністю цієї інформації.

Google Календар є гарним інструментом для планування часу вручну, але іноді в людей не вистачає ресурсів, щоб усе спланувати власноруч. Також залишається не вирішеною проблема планування часу зустрічі, який би підходив усім учасникам. Розв'язання цих проблем ми показуємо у розділі 2.

2.2 Microsoft Outlook Calendar

Також однією з найпопулярніших програм, які призначені як для особистого використання, так і для бізнесу, є Outlook Calendar [5]. Як і Google Календар він має зручний інтерфейс для ручного планування подій:

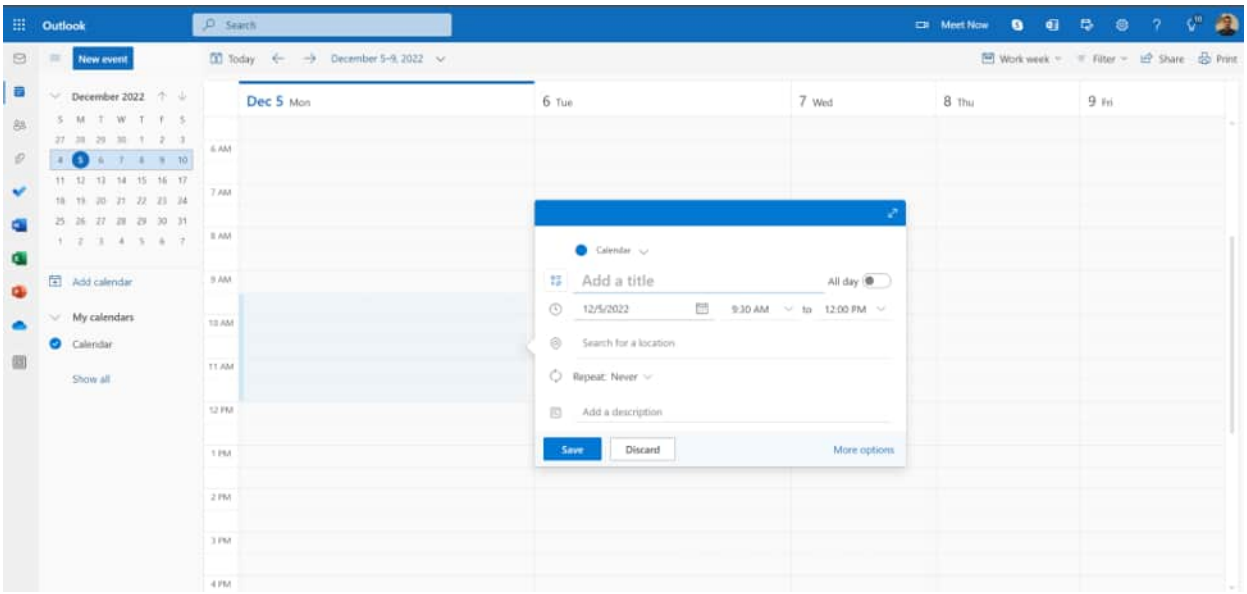

Рисунок 2.3 Інтерфейс програми «Outlook Calendar»

На відміну від Google Календаря, Outlook показує організатору, чи котрийсь з учасників зайнятий в період проведення конкретної події. Також при додаванні нових учасників він попереджає про це і пропонує змінити час події.

Утім, вибрати час події все одно мусить організатор, графік якого часто не дозволяє задуматись над часом і місцем проведення та проводити опитування серед учасників.

2.3 Microsoft Project

Серед програм які мають функцію автоматизованого планування часу найбільш помітною є програма Microsoft Project [6]. Вона призначена для планування проєктів у бізнесі, тому цільовою аудиторією цієї програми є саме менеджери компаній. Оскільки плануються завдання глобального характеру на проєкті, час у названій системі вимірюється днями, а не годинами.

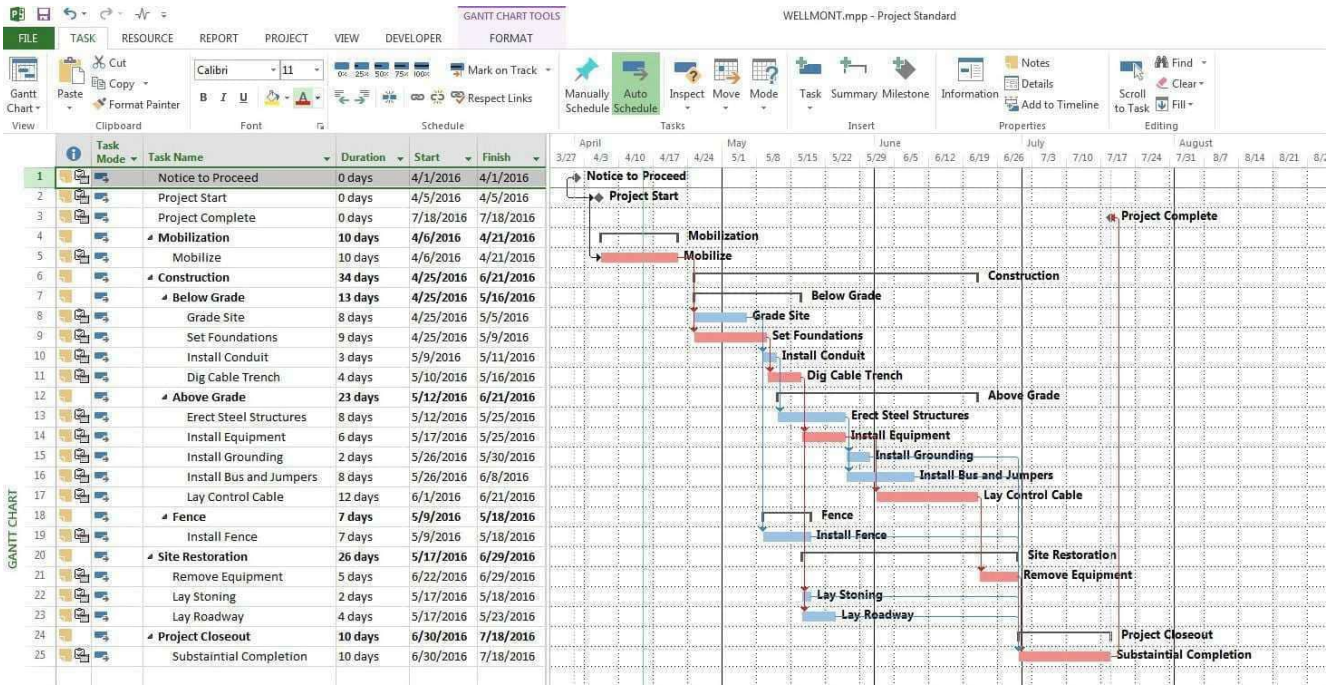

Рисунок 2.4 Інтерфейс програми Microsoft Project.

У цій програмі можна створювати завдання автоматично, визначивши необхідну кількість днів на його виконання і відношення з іншими завданнями. Завдання можна побачити в списку (див Рисунок 2.4 зліва) і у вигляді діаграми Ганта (див Рисунок 2.4 справа). Залежно від відношень між завданнями, нове

може бути розміщене одразу після завдання, від якого воно залежить, або ж паралельно з ним, у межах термінів конкретного проєкту.

Для особистого користування така програма не підходить з декількох причин. По-перше, не вийде запланувати точний час початку і кінця певної події. По-друге, для автоматичного планування потрібно налаштовувати зв’язки між завданнями, що ускладнює процес планування.

Додатки для особистого використання повинні не лише мати достатній функціонал, а й бути доступними для широкого загалу, зокрема простими у користуванні. Аналіз двох програм показав, що жодна із них не володіє усіма потрібними якостями. Отже, розв’язання проблеми полягає у тому, щоб на основі поєднання сильних сторін різних видів програм-планерів, створити застосунок, який би відповідав усім зазначеним раніше вимогам.

3 ОПИС АЛГОРИТМІВ

У більшості функцій розроблена система покладається на наявні інструменти використаних технологій (див. Розділ 4). Проте було розроблено алгоритми для основних функцій системи, якими вона відрізняється від своїх аналогів, а саме: автоматичним плануванням часу та місця проведення подій.

3.1 Автоматичне планування часу звичайної події

Нехай користувач має на своєму календарі певну кількість власних подій, а також подій, на які його запросили інші користувачі системи. Події можуть повторятись з певною періодичністю. У такому випадку будемо розглядати кожне повторення як окрему подію. Також на подію можуть бути запрошені учасники. Пошук часу для такої події вимагатиме врахування усіх власних подій учасників, а також подій, на які вони запрошені. Для простоти викладу розглядатимемо події як відрізки задані парою значень: час початку та час кінця.

Користувач хоче запланувати подію заданої тривалості у вказаних часових межах. Він визначає дату і час, коли найраніше подія може початися (наприклад через пів години після поточного часу), а також інші дату і час - крайній термін для автоматичного пошуку (наприклад до кінця тижня, а саме: до 23:59 неділі). Також користувач може обмежити пошук в межах кожної доби задавши час початку та кінця дозволеного періоду (наприклад від 09:00 ранку до 17:00 вечора, щоб заборонити планування робочих зустрічей в позаробочий час).

Кроки алгоритму:

1. Об'єднати в один список усі власні події користувача, події на які він запрошений, усі події учасників, а також можливі повторення кожної з подій у часових межах автоматичного пошуку, заданих користувачем.

2. Якщо задані часові рамки в межах доби, додати до списку подій з попереднього кроку зайняті ділянки кожного дня, недоступні для автоматичного планування.

3. Отриманий список подій трансформуємо у масив часових інтервалів, фільтруємо усі неунікальні елементи з нього, а також їх сортуємо.

4. Знаходимо список вільних часових інтервалів протягом доступного для автопланування періоду часу. Спочатку бажаний список складається з одного елементу - інтервалу, який відображає часові межі автоматичного планування. Далі по черзі беремо кожну подію з вхідного списку і "вирізаємо" з кожного елемента вихідного масиву частину, яка накладається з цією подією:

- Якщо подія знаходиться поза межами інтервалу, ми залишаємо його таким як є і переходимо до наступного;

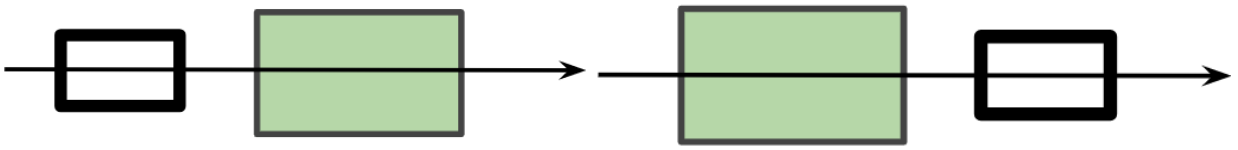

Рисунок 3.1 Подія знаходиться за межами інтервалу

- Якщо подія накладається на інтервал частково, час початку інтервалу змінюємо на час кінця події, або час кінця інтервалу змінюємо на час початку події, в залежності від того, з якого боку подія накладається на інтервал;

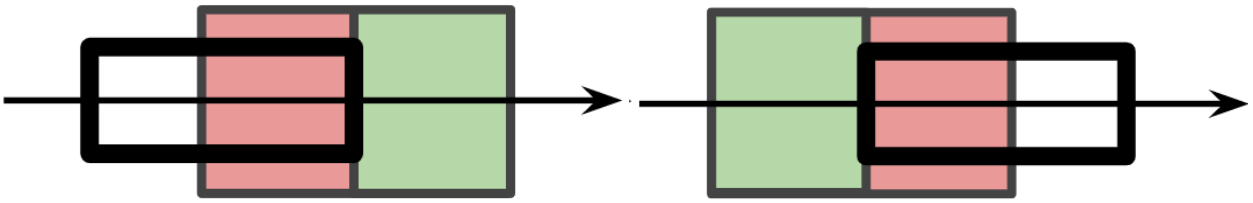

Рисунок 3.2 Подія накладається на інтервал частково

- Якщо подія повністю накладається на інтервал, тоді ми такий інтервал видаляємо з вихідного списку, оскільки його довжина дорівнюватиме нулю;

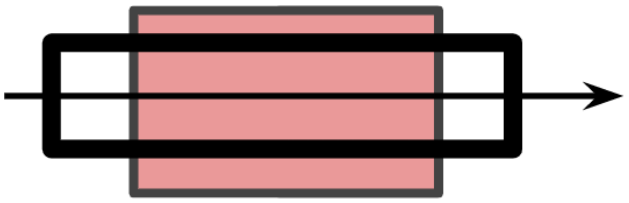

Рисунок 3.3 Подія накладається на інтервал повністю

- Якщо подія лежить на інтервалі, так що залишається місце від початку інтервалу до події та від події до кінця інтервалу, ми розбиваємо такий інтервал на дві частини та додаємо їх у вихідний масив;

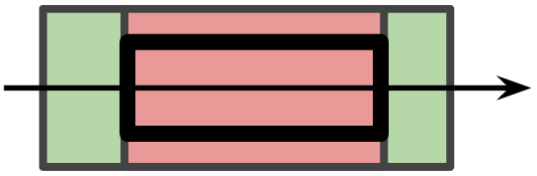

Рисунок 3.4 Подія лежить у межах інтервалу

Для кожної наступної події список вільних інтервалів буде іншим, тому після виконання операції, описаної вище, для кожної події, список інтервалів буде складатись виключно з вільних інтервалів часу.

5. Сортуємо список отриманих інтервалів за часом початку кожного з них

6. Шукаємо у списку перший інтервал тривалість якого буде більшою або рівною тривалості події, час для якої ми шукаємо.

7. Змінюємо час початку події на часу початку знайденого інтервалу, а кінець події знаходимо додавши тривалість до нового часу початку.

Код алгоритму автоматичного планування звичайної події наведено в додатку А.

3.2 Автоматичне планування часу події, що повторюється

Нехай подія, час для якої хоче запланувати користувач, повторюється з певною періодичністю. Обмежимося чотирма видами повторень: по днях, по тижнях, по місяцях, по роках. Користувач може задати частоту повторення (наприклад щодня, кожні два тижні тощо). Якщо період повторення - тиждень, користувач може задати в які саме дні тижня подія буде повторятись (наприклад у будні). У випадку повторення раз в місяць, можна вказати який саме це день місяця (наприклад 28 грудня, у четверту середу грудня, останню середу грудня тощо).

Також важливо вказати умову завершення повторень події. Подія може повторюватись нескінченно, завершуватись в певну дату, або ж через певну кількість повторень. У певні дні в користувача можуть змінитись плани, або ж він може захворіти тощо. Саме тому користувач повинен мати змогу змінити час певного повторення події, не змінюючи їх усіх.

Автоматичне планування події, що повторюється, полягає у пошуку в часових межах, заданих користувачем, часу початку першої події, а також корегуванні усіх наступних повторень події у дні, де повторення не вдається запланувати по графіку. Код функції перевірки наявності повторення події в межах вибраного дня надано у додатку Г.

Кроки алгоритму:

1. автоматично запланувати час першої події (див 3.1), запам'ятавши список вільних часових інтервалів;

2. видалити окремі повторення події, які виділяються з графіку, у часових межах автоматичного планування;

3. пройтися по днях від початку автопланування до його крайнього терміну;

4. якщо в цей день є повторення події, тоді серед вільних інтервалів в межах цього дня знайти такий що містить повторення події;

5. якщо такого інтервалу немає, знайти найближчий інтервал зліва або справа, якого буде достатньо щоб помістити в нього повторення події цього дня;

6. якщо в межах дня не вдається знайти жодного інтервалу, скасувати повторення події цього дня.

Код алгоритму автоматичного планування часу події, що повторюється наведено в додатку Б.

3.3 Автоматичне планування місця події

Нехай користувач запросив на подію певну кількість учасників. Кожен учасник події, а також організатор задали свою домашню адресу. Припустивши, що усі учасники добиратимуться на подію з дому, можна знайти точку на карті, яка мінімізує відстань від дому кожного з учасників до місця події. У цьому і полягає задача автоматичного планування місця події.

Розглянемо декілька варіантів пошуку центральної точки:

Центр мас - зважене середнє значення заданих точок по кожній з координат [7]. Обчислюється за формулою:

$$\frac{\sum_{i=1}^{m} m_i x_i}{\sum_{i=1}^{m} m_i} \tag{3.1}$$

де m_i - маса i-ої точки, x_i - i-та точка.

Якщо присутність усіх учасників події однаково важлива, маса кожної точки буде дорівнювати одиниці, а отже формула 3.1 виродиться у формулу середнього арифметичного.

Геометрична медіана - точка яка мінімізує суму евклідових відстаней від неї до кожної з точок [8]. Обчислюється за формулою:

$$\arg\min_{y \in \mathbb{R}^n} \sum_{i=1}^{m} d(x_i, y) \tag{3.2}$$

де n - вимір (у нашому випадку дорівнює 2), m - кількість точок (учасників події), x_i - i-та точка, $d(a,b)$ - евклідова відстань між точками a та b, argmin - мінімальне значення аргументу функції на множині [8].

На рисунку 3.5 жовтим кольором виділено геометричну медіану, а темно-синім кольором – центр мас.

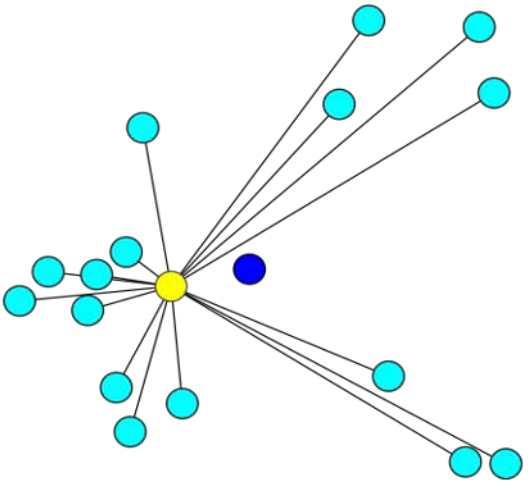

Рисунок 3.5 Геометрична медіана та центр мас

На центр мас більшою мірою впливають викиди (outliers), наприклад учасники які живуть в інших містах. Медіана ж дозволяє визначити місце події в населеному пункті де проживає більша кількість учасників, а отже є більш корисною для розв'язання нашої задачі.

Значення геометричної медіани шукають наближено за допомогою ітеративних алгоритмів. Далі буде описано алгоритм, реалізований в межах цієї роботи.

Кроки алгоритму:

1. Як можна бачити з рисунка 3.5, геометрична медіана знаходиться неподалік від центру мас. Отже, візьмемо центр мас як початкове наближення нашого алгоритму.

2. Знайдемо мінімальне і максимальне значення усіх точок по кожній з координат, оскільки шукана точка повинна знаходитись в цих межах. Початковий крок алгоритму знайдемо поділивши різницю максимальних і мінімальних значень навпіл.

3. Поки крок алгоритму більший за наперед задане значення похибки наближення (ε):

4. Рахуємо суму евклідових відстаней від поточного наближення до заданих точок.

5. Від цього наближення рухаємось по черзі вертикально (чи горизонтально) вгору, вниз, вліво та вправо на поточний крок алгоритму, і в кожній з точок рахуємо суму відстаней до решти точок**.**

7. За наступне наближення алгоритму візьмемо точку, сумарна відстань від якої до решти точок виявилась мінімальною.

8. Ділимо крок алгоритму на два і повторюємо кроки 4-8 поки крок алгоритму не стане меншим за наперед задане ε.

Код алгоритму автоматичного планування місця події наведено в додатку В.

4 ВИКОРИСТАНІ ТЕХНОЛОГІЇ

Реалізація інформаційної системи виконана у формі повноцінного веб-застосунку. Він складається з трьох основних частин: клієнтської частини (React), серверної частини (Node.js), та бази даних (PostgreSQL).

4.1 Клієнтська частина

За зовнішній вигляд сайту, зручність користувацького інтерфейсу, а також швидкість відклику відповідає JavaScript бібліотека React [9]. ЇЇ суть полягає в, тому що додаток містить лише одну HTML-сторінку, яка при перемиканні між сторінками сайту не змінюється цілком, а оновлюється лише її частина за допомогою коду. Це відбувається значно швидше, ніж завантаження нової HTML-сторінки. Такі застосунки називають **односторінковими** [10].

Також разом з ним, в клієнтській частині, було використано бібліотеку MaterialUI, яка надає зручні інструменти для розмітки сторінки (Flexbox, Grid Layout), елементи форм та іконки. Ця бібліотека допомагає будувати вебдодатки, які мають однаково гарний вигляд як на великих екранах моніторів, так і на мобільних пристроях (смартфонах, планшетах тощо) [11].

Клієнт комунікує з сервером за допомогою HTTP-запитів, які відправляє бібліотека Axios. Axios.js це HTTP-клієнт, тобто може використовуватися як в браузері, так і на сервері (Node.js) за допомогою однакового коду [12].

4.2 Серверна частина

Авторизація користувачів, усі обчислення, а також керування подіями відбувається на стороні сервера. Серверна частина сайту побудована на JavaScript фреймворку Node.js, який призначений для виконання високопродуктивних мережевих застосунків, написаних мовою програмування JavaScript. Перевагою Node.js є його асинхронна модель виконання коду, заснована на обробці подій у не блокувальному режимі, що забезпечує обробку великої кількості паралельних запитів [13].

Разом з ним було використано вебфреймворк Express. Його перевагами є простота та гнучкість побудови вебзастосунків на платформі Node.js [14].

Усі дані про користувачів та події зберігаються в базі даних PostgreSQL. PostgreSQL це потужна, об’єктно-реляційна база даних з відкритим кодом, що використовує і розширює можливості мови запитів SQL [15].

4.3 Хостинг

Вебзастосунок працює на безплатній версії хостингу Render [16]. Налаштовано CI/CD інтеграцію з репозиторієм на Github. При будь-яких змінах в головній гілці, оновлюється і сам сайт. [17] Також у вебдодатку використовується безкоштовна версія хмарної бази даних PostgreSQL, надана цим хостингом. Render — хмарна платформа як сервіс (PaaS - Platform as a service), яка дозволяє програмістам публікувати свої вебдодатки безплатно, а при зростанні кількості користувачів надає змогу масштабувати інфраструктуру сайту (збільшити швидкодію, розмір сховища тощо), за встановлену суму [18].

4.4 Сервіси Google

Для вибору місця проведення події, а також пошуку адреси, вибраної на карті використано сервіс Google Maps API. Він надає зручну компоненту користувацького інтерфейсу у вигляді карти, а також низку корисних HTTP-запитів, які надсилає сайт, щоб знайти адресу за координатами, або ж навпаки геокодувати адресу, щоб отримати точку на карті [19].

Також користувач може прив'язати свій обліковий запис Google, імпортувати власні події зі застосунку Google Calendar та навпаки, експортувати події у календар користувача. Для цього використовується сервіс Google Calendar API. Він дозволяє програмно керувати календарями до яких користувач надав доступ [20].

5 ОПИС РОБОТИ ПРОГРАМИ

5.1 Опис основних можливостей системи

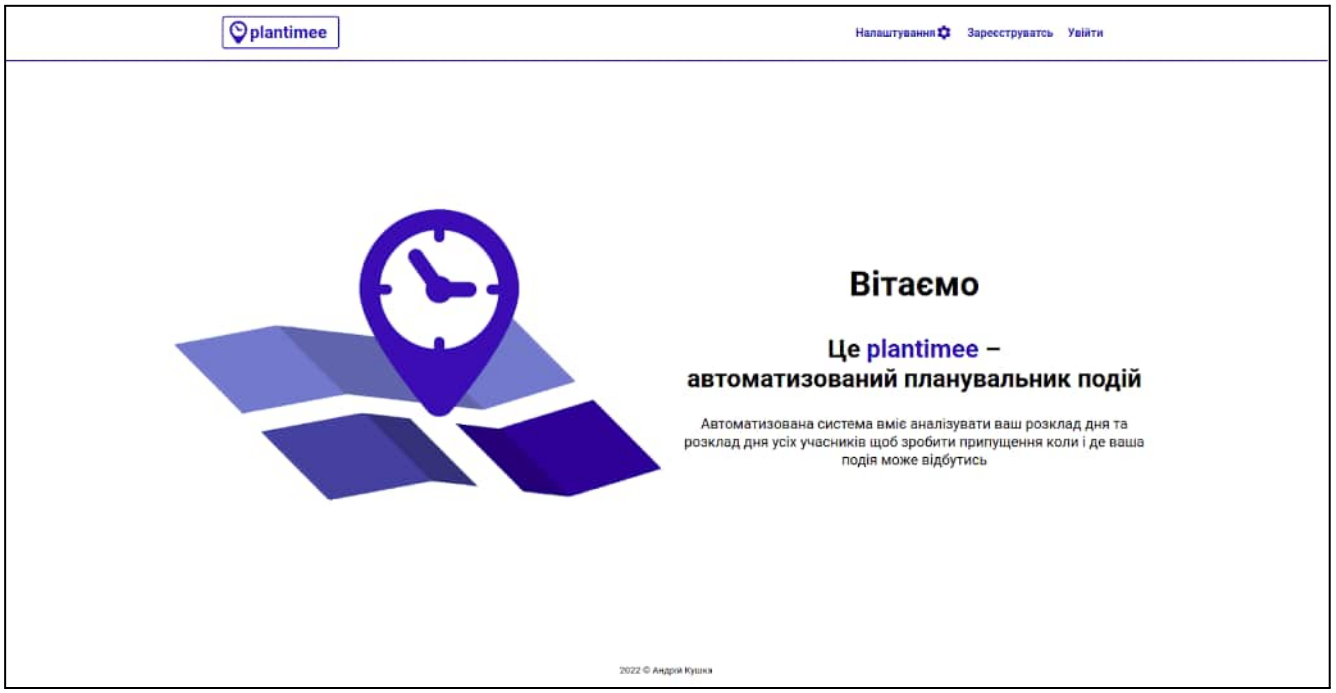

Рисунок 5.1 Домашня сторінка сайту

Додаток передбачає авторизацію користувачів за нікнеймом та паролем. Також при реєстрації можна ввести свою електронну пошту та повне ім’я, які потім зможуть побачити інші користувачі, запрошуючи на свою подію.

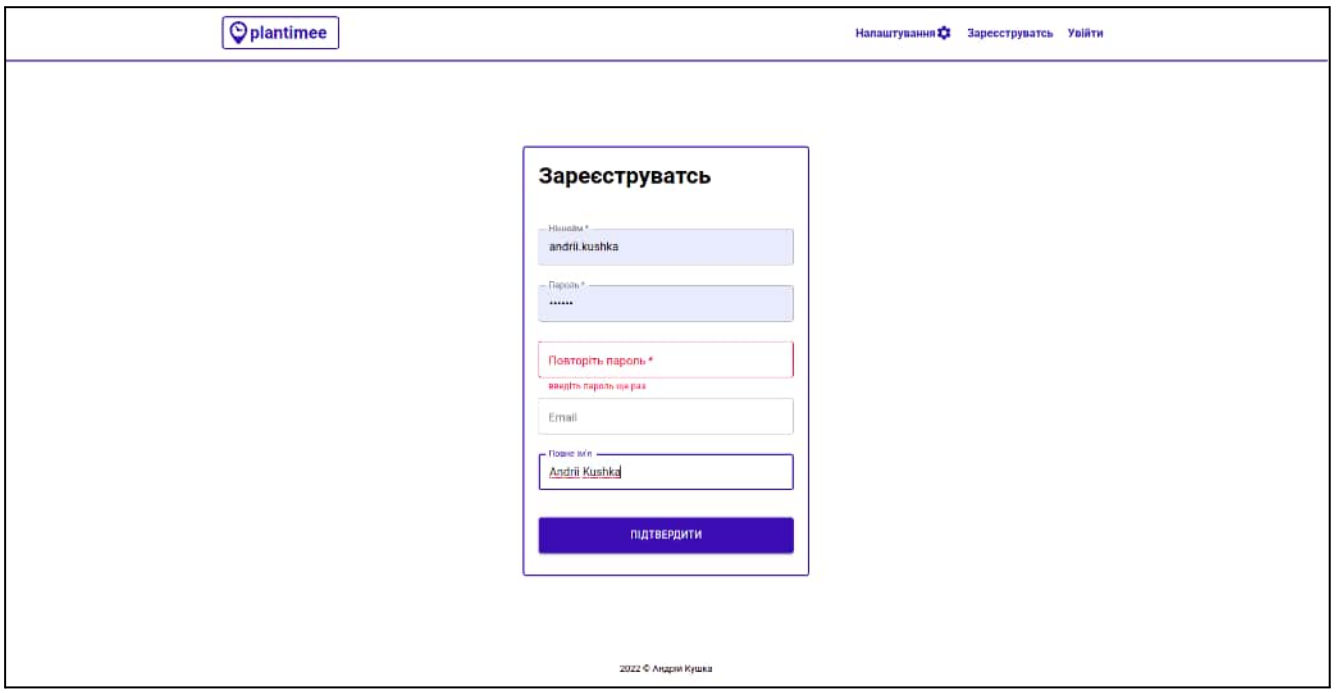

Рисунок 5.2 Сторінка реєстрації

Зареєстрований користувач може увійти в систему за своїм нікнеймом та паролем. Усі паролі в базі даних зберігаються в зашифрованому вигляді, і навіть той, хто має доступ до бази, не може його дізнатись.

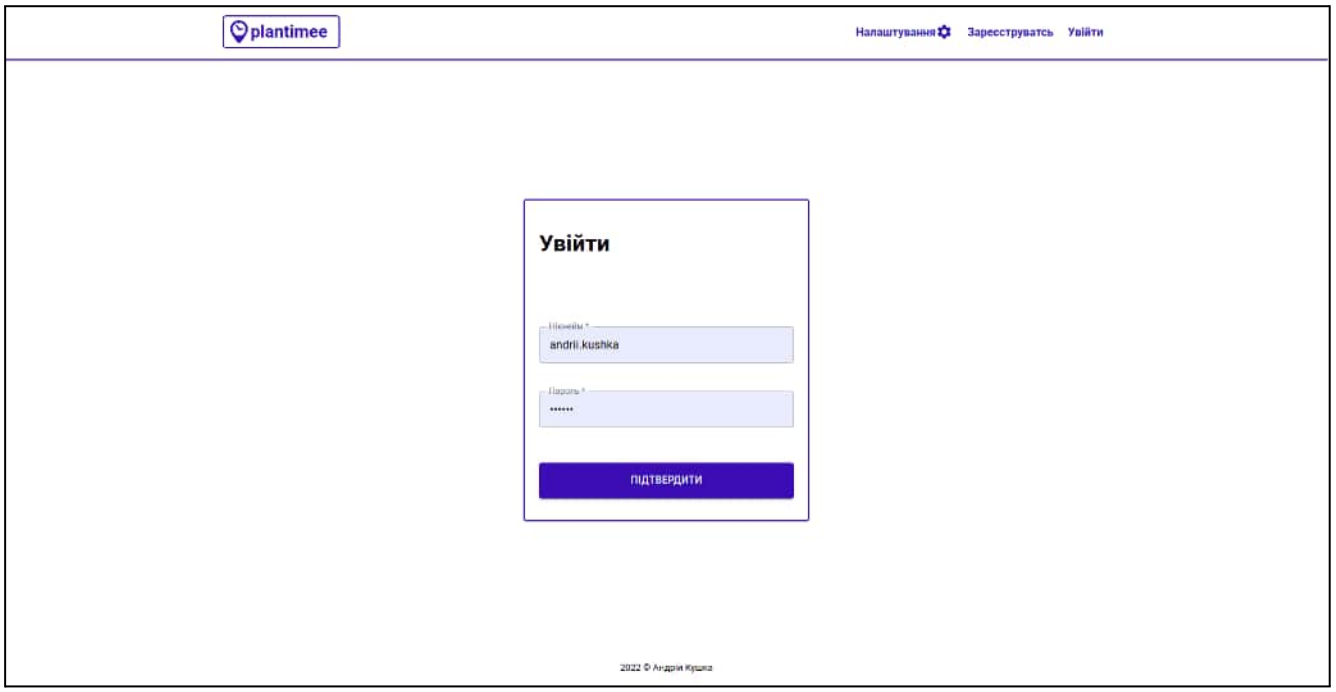

Рисунок 5.3 Сторінка входу в систему

Логотип в лівому верхньому куті є кнопкою, при натисканні на яку додаток перейде на домашню сторінку.

Правильно ввівши нікнейм та пароль, користувачу стає доступним перегляд його профілю, а також панель керування подіями. При натисканні кнопки "Вийти" користувач виходить з системи. При переході на сторінку профілю користувач бачить свої дані.

Також користувач може налаштувати свою домашню адресу, змінити повне ім'я чи електронну пошту.

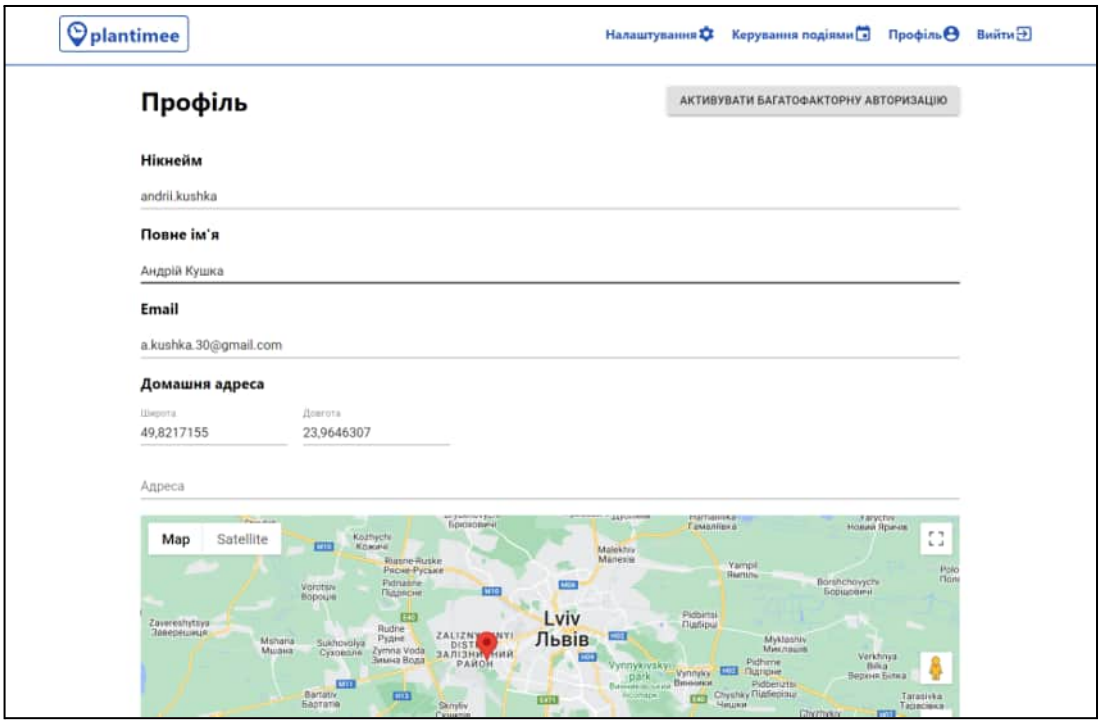

Рисунок 5.4 Сторінка профілю користувача.

На сторінці профілю користувача знаходиться кнопка "Активувати багатофакторну авторизацію", яка дозволяє збільшити рівень захисту персональних даних користувача. Натиснувши на неї користувач побачить модальне вікно зі списком необхідних дій для налаштування цієї функції.

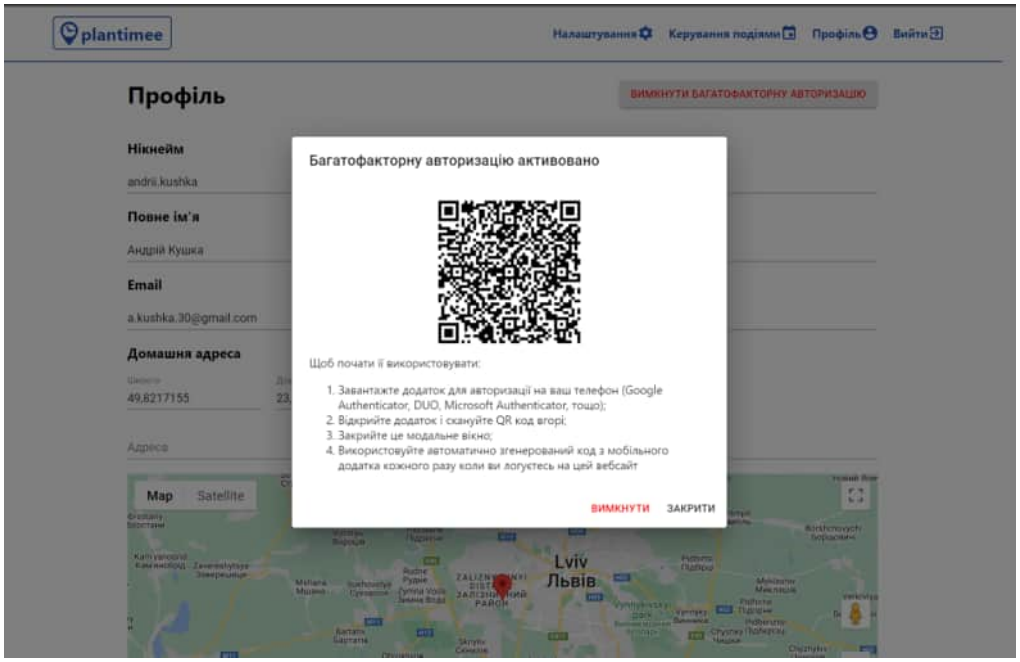

Рисунок 5.5 Налаштування багатофакторної авторизації

Мобільний додаток кожні 30 секунд генерує шестизначний код, який користувач повинен ввести на сторінці входу в систему, після успішної перевірки нікнейму і паролю.

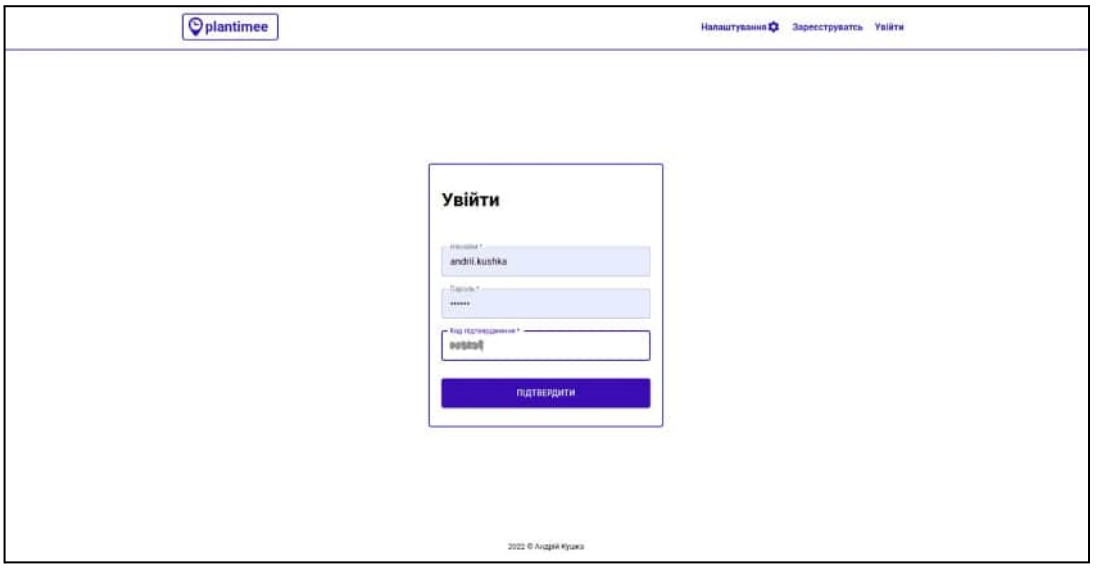

Рисунок 5.6 Вхід за допомогою багатофакторної авторизації

Основна сторінка сайту це "Керування подіями". Вона складається з трьох колонок (зліва направо): "Часова шкала", "Події" і "Налаштування".

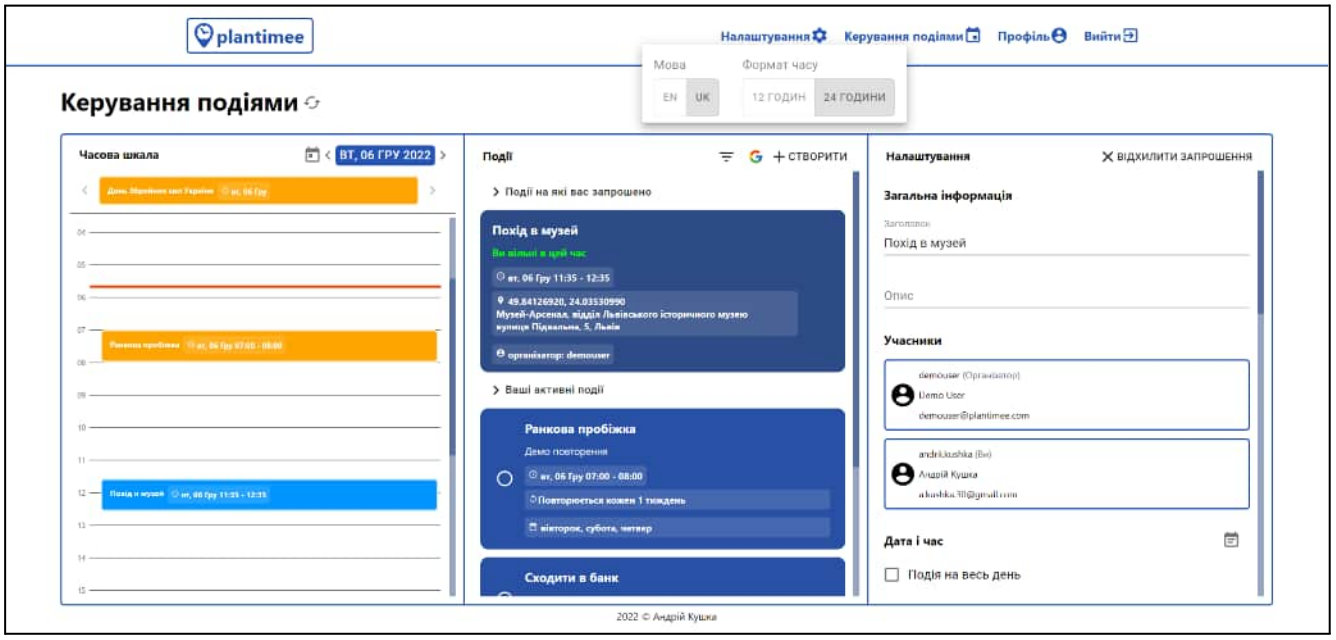

Рисунок 5.7 Сторінка керування подіями

У колонці "Події" знаходиться перелік усіх подій, на які запросили користувача, його власних подій, подій на весь день, а також перелік подій, які є

завершеними. Усі ці блоки подій можна заховати/відобразити, натиснувши на них лівою кнопкою миші, або ж пальцем, у випадку використання мобільних пристроїв. При натисканні на круг зліва від активної події – вона стане виконаною і перенесеться в блок виконаних подій. Виконану подію можна відновити, натиснувши на галочку зліва від неї.

При натисканні на блок з подією, дані про неї з’являться в колонці зліва, а сам блок стане темнішого кольору. Блок з подією, на яку користувача запросили, містить ім’я організатора. При виборі такої події, поля в колонці налаштувань будуть неактивними, а окремі поля не будуть відображені. Також у верхній частині колонки стає доступною кнопка "Відхилити запрошення" для видалення події зі свого календаря, після натискання на яку подія зникне із загального списку подій, а також з часової шкали.

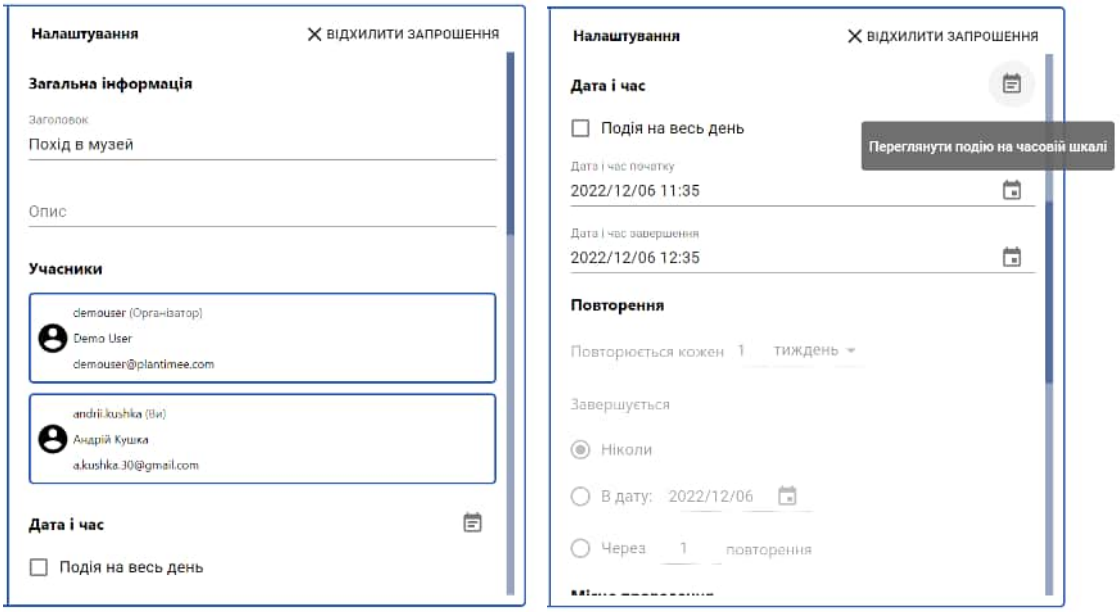

Рисунок 5.8 Інформація про вибрану подію на яку користувач запрошений

Як бачимо на рисунку вище, колонка з інформацією про вибрану подію, на яку користувач запрошений, містить назву та опис події. Зазначено, чи подія триває весь день, дати початку та кінця події, налаштування повторення події, а також місце проведення. Якщо подія має місце проведення, внизу колонки "Налаштування" можна знайти його координати та карту, а також адресу та опис місця якщо вони знайдені.

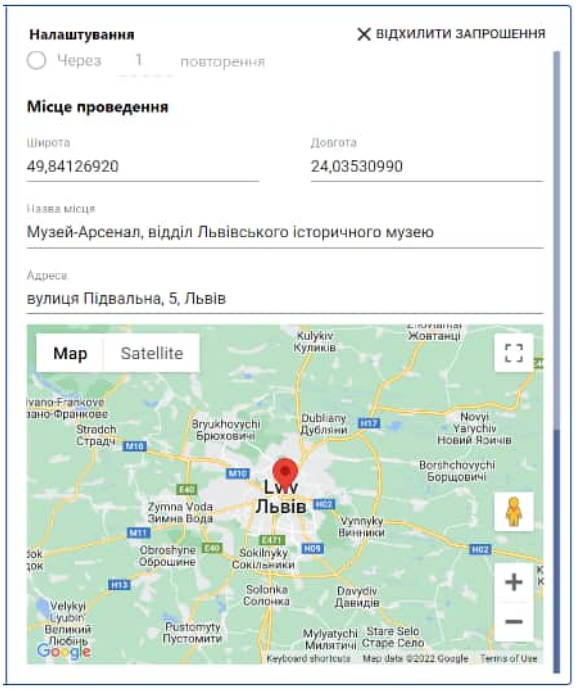

Рисунок 5.9 Карта місця проведення події

У випадку вибору власної події кнопка "Відхилити запрошення" зміниться кнопкою видалення події. Під блоком з учасниками події з'явилось поле для введення нікнейму або електронної пошти іншого користувача з метою запрошення його на власну подію. На цьому прикладі зображена подія, де учасників немає тому в події бере участь лише її організатор:

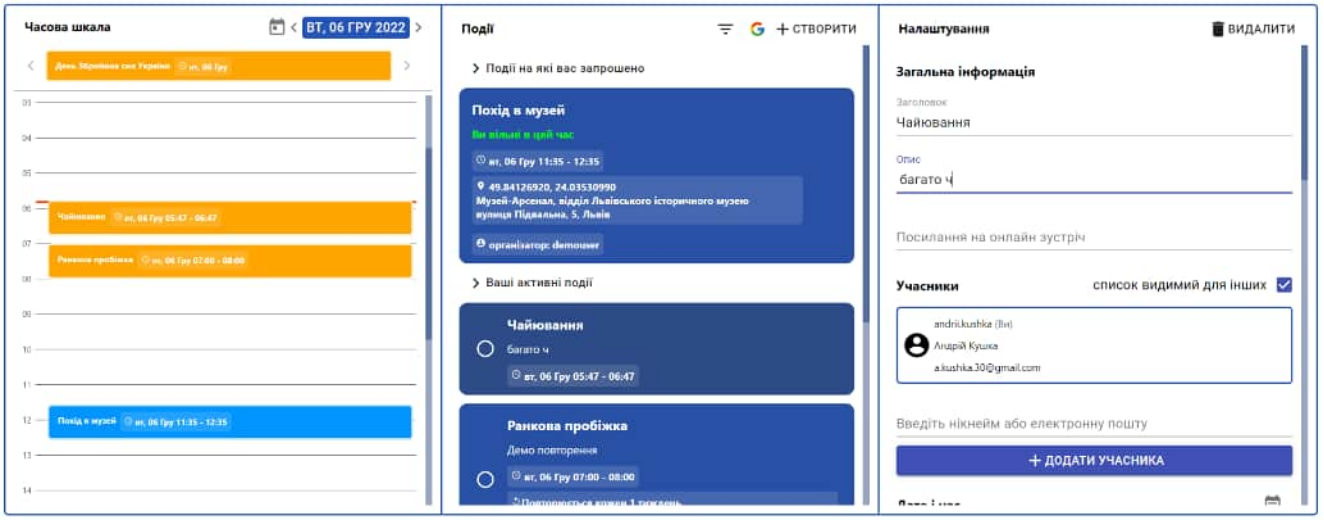

Рисунок 5.10 Налаштування власної події користувача

Поля форми на колонці налаштувань власної події користувача є активними та можуть бути змінені. Зміни збережуться в базу даних щойно користувач

переключиться на інший елемент форми або натисне кнопкою мишки в довільній точці екрана поза полем введення.

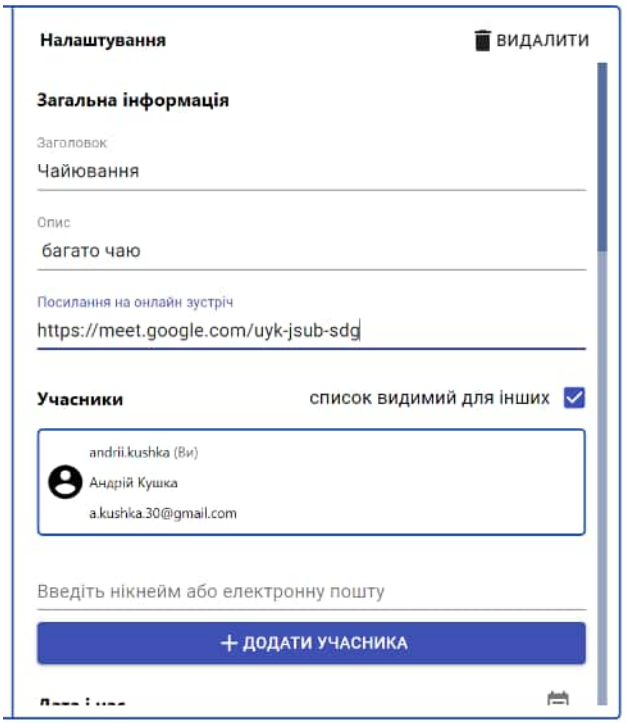

Рисунок 5.11 Внесення змін в інформацію про подію

Як можна побачити на малюнку нижче, ми ввели нікнейм іншого користувача у полі додавання нового учасника. Після натискання кнопки "Додати учасника" він з’явиться у списку.

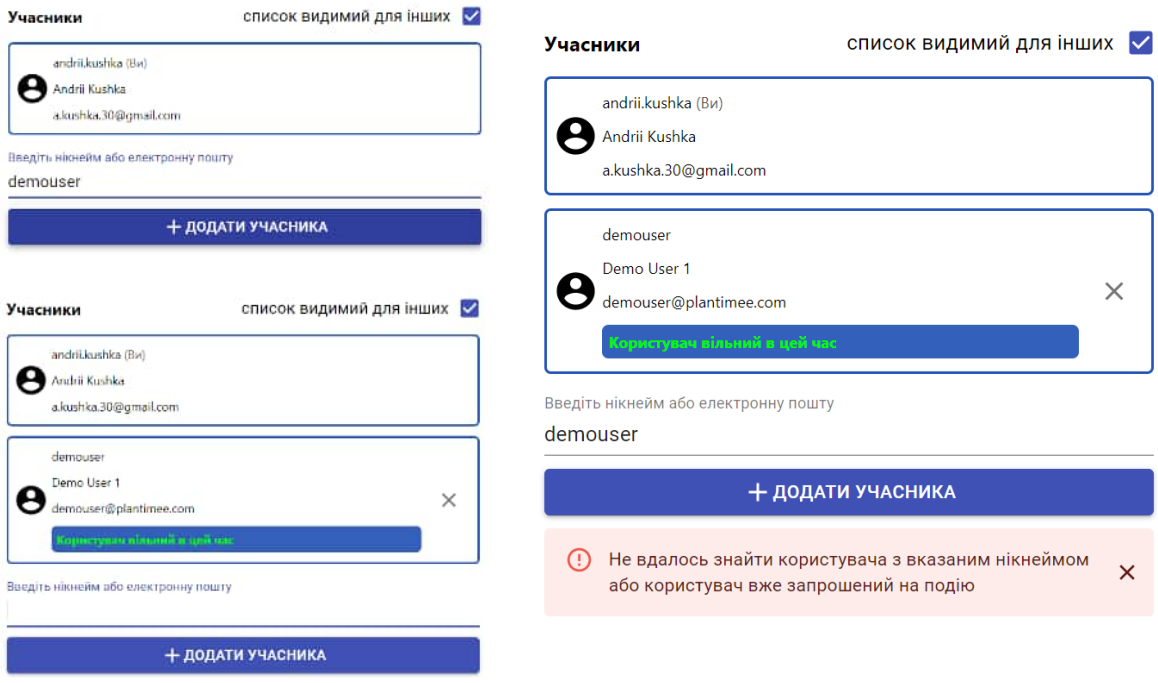

Рисунок 5.12 Додавання учасника на подію

Власник події може бачити нікнейм та повне ім’я користувача, а також його електронну пошту. Також організатор може виключити з події раніше запрошеного учасника, натиснувши відповідну кнопку поруч з ним. Якщо нікнейм чи електронна пошта введені неправильно, або учасник вже запрошений на подію, то користувач побачить відповідне повідомлення.

Також над блоком з учасниками знаходиться прапорець, який дозволяє налаштувати те, чи бачитимуть учасники події інших учасників. Якщо цей прапорець зняти, то учасник буде бачити в списку лише себе та організатора.

5.2 Планування місця події

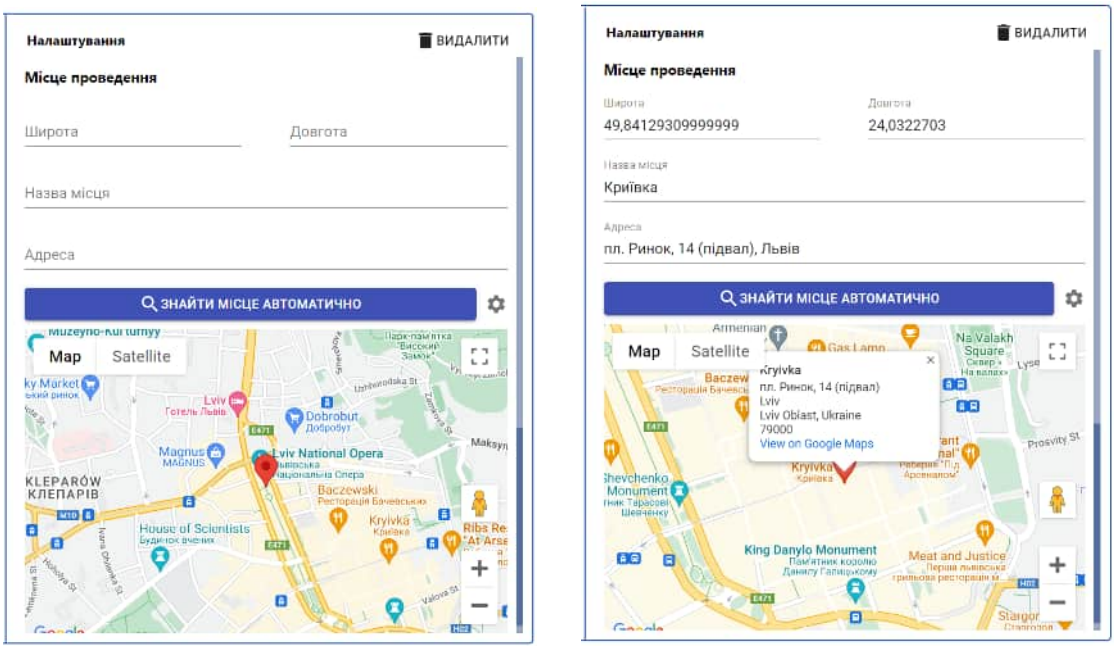

Рисунок 5.13 Налаштування місця проведення власної події

Внизу форми налаштування власної події знаходяться два поля з координатами, поле для назви та адреси місця, а також карта, на якій можна вибрати точку. При зміні точки на карті координати встановлюються автоматично. І навпаки – після введення координат у поля над картою, точка на карті змінить своє положення відповідно. При виборі певних місць, система автоматично підвантажить назву та адресу місця згідно з координатами.

Також над картою розміщена кнопка "Знайти місце автоматично". При натисканні на кнопку з зображенням шестерні, що знаходиться поруч, користувач побачить модальне вікно налаштувань:

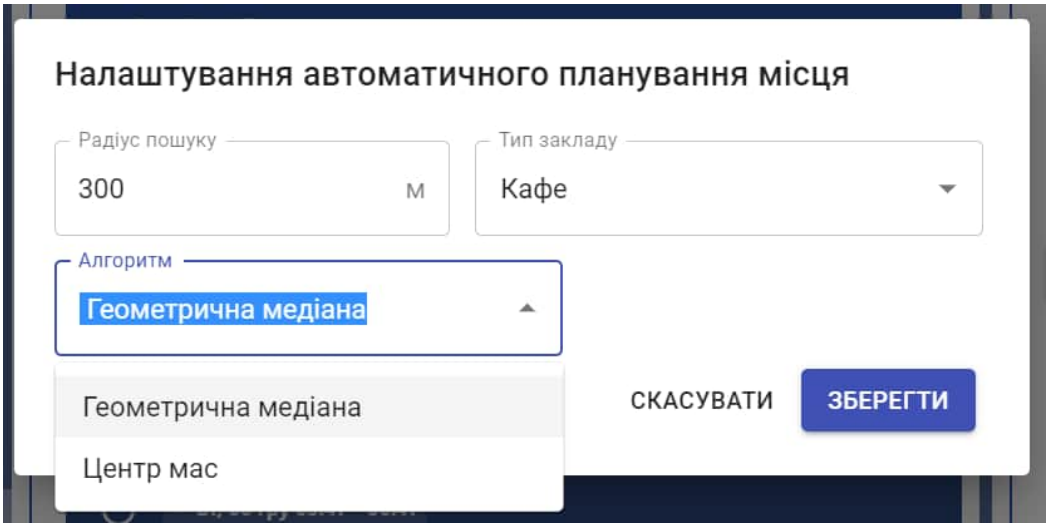

Рисунок 5.14 Налаштування автоматичного планування місця події

Тут можна налаштувати радіус навколо знайденої центральної точки, в межах якого система знайде перелік місць, серед яких користувач зможе вибрати те, що найкраще підходить для його події. Також тут можна обрати тип закладу для фільтрування результатів пошуку й алгоритм пошуку місця. Детальніше ці алгоритми описано у розділі 3.

Розглянемо автоматичний пошук місця події з використанням обох алгоритмів. Нехай ми хочемо запланувати подію - зустріч трьох друзів. Зобразимо на карті трикутник, кути якого це домашні адреси учасників події

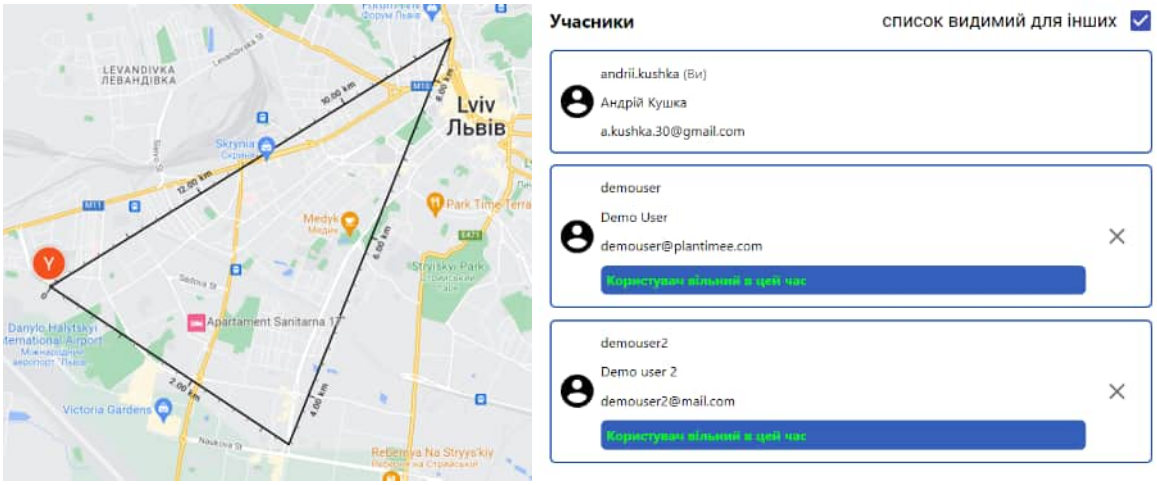

Рисунок 5.15 Початкові умови для планування місця

Спочатку заплануємо місце події використовуючи алгоритм "Геометрична медіана", тип закладу - кафе, радіус - 300 метрів (див рисунок 5.14):

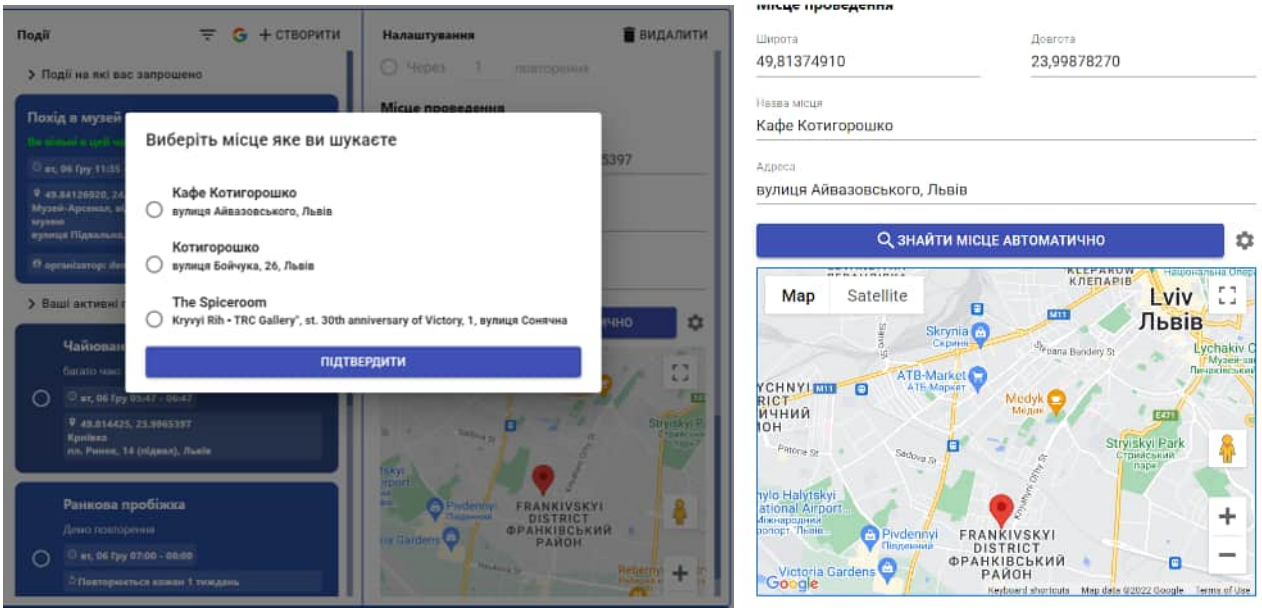

Рисунок 5.16 Місце події заплановано автоматично

Далі заплануємо місце за допомогою алгоритму "Центр мас", тип закладу - ресторан, радіус - 100 метрів

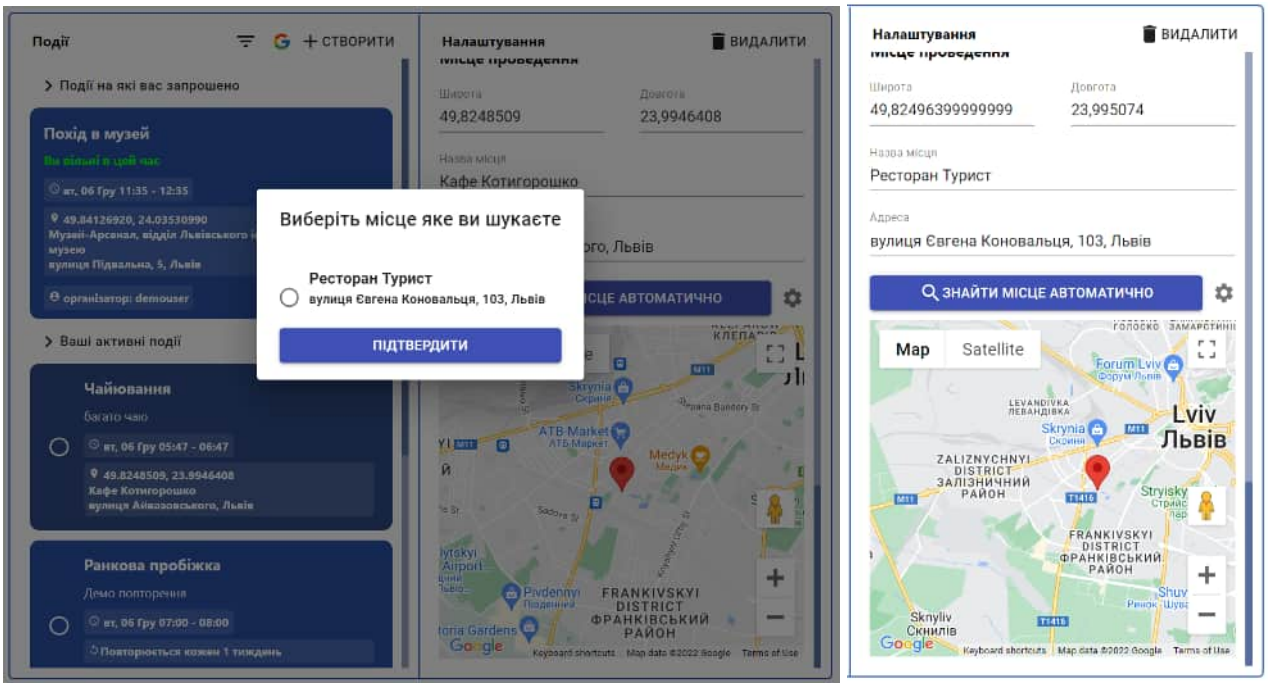

Рисунок 5.17 Місце події заплановано алгоритмом "Центр мас"

Як бачимо, алгоритми дають різні результати, що дозволяє нам вибрати той який найбільше підходить у конкретному випадку.

5.3 Часова шкала та планування часу події

Спочатку розглянемо детальніше колонку з часовою шкалою:

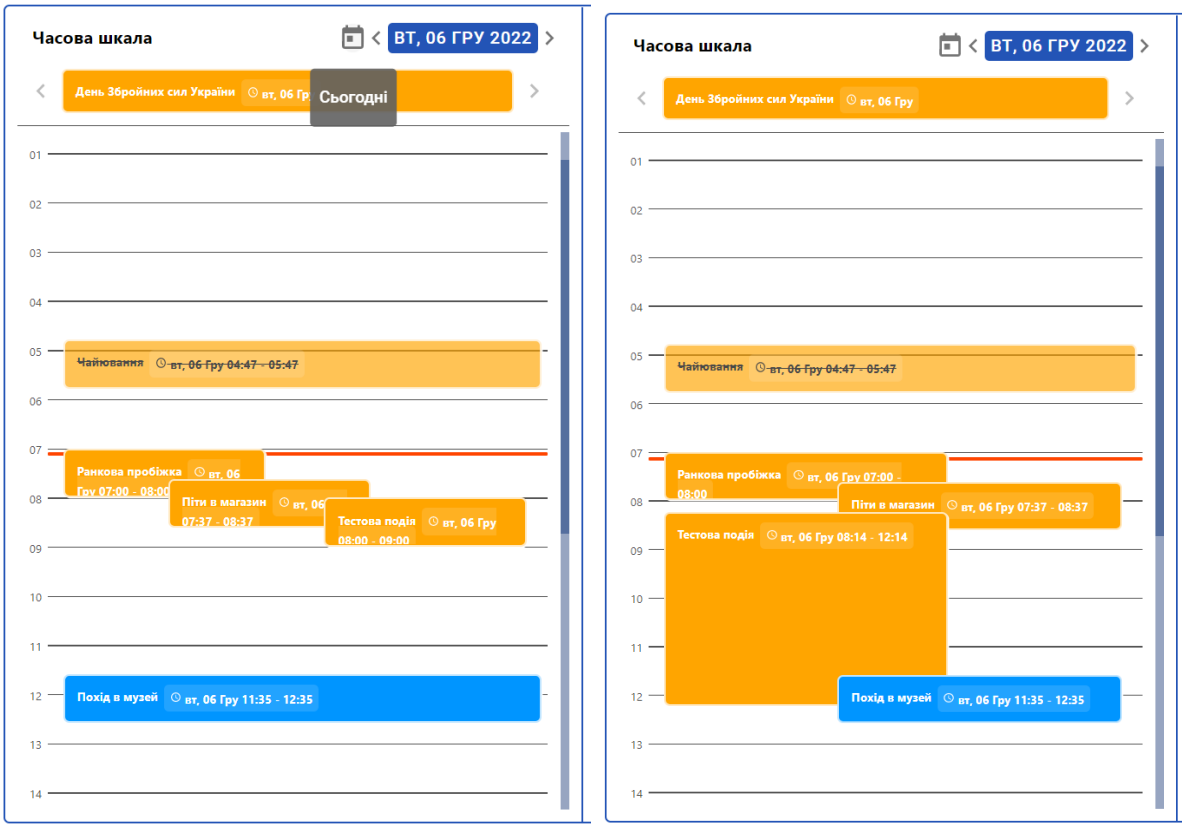

Рисунок 5.18 Часова шкала

На часовій шкалі розміщено поділки, що відображають години в добі, Також у верхній частині колонки ми можемо побачити вибраний день. За замовчуванням на часовій шкалі вибрана сьогоднішня дата (виділяється темно-синім кольором). Користувач може переключатися між датами на часовій шкалі за допомогою стрілок вліво та вправо, а також при натисканні на дату відкриється випадний календар.

Червона лінія близько 7-ї години це поточний час, світло-оранжеві блоки це власні події, світло-сині блоки це події на які користувача запрошено, а напівпрозорі блоки з перекресленим текстом – завершені події. Їхня висота відповідає тривалості, а крайня верхня межа – часу початку. При зміні дат у колонці налаштувань блок на часовій шкалі автоматично змінює свій розмір та положення, а при видаленні події – зникає.

Якщо події накладаються вони змінюють свою ширину і положення, щоб стати в ряд з іншими подіями які тривають паралельно. На рисунку 4.16 можна бачити декілька прикладів таких розміщень.

Події можна позначити як такі що тривають весь день за допомогою відповідного прапорця на колонці "Налаштування". У такому разі вони будуть відображені над часовою шкалою. Якщо в межах одного дня є декілька таких подій ми можемо перемикатися між ними стрілками вліво та вправо:

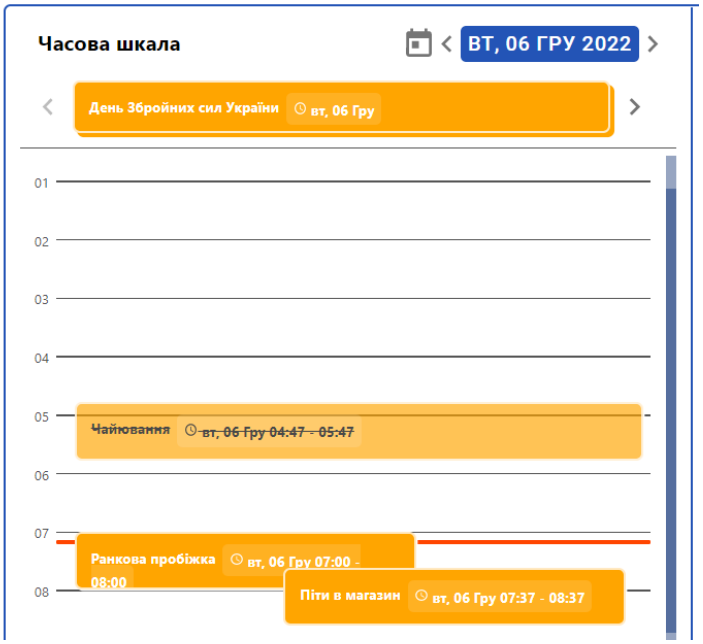

Рисунок 5.19 Часова шкала з двома подіями на весь день

Повернемось до колонки за налаштуваннями події. Під списком учасників ми бачимо панель вибору часу події. Якщо подія відбувається увесь день, з панелей керування часом початку та кінця події зникне можливість вибрати час.

Під полями для вибору дат знаходиться блок, який відображає тривалість події. Ці значення обчислюються автоматично при зміні часу початку і кінця події. Також цю тривалість можна змінювати. При цьому час завершення події буде вираховуватися залежно від часу початку і тривалості. Також час завершення змінюється при зміні часу початку події враховуючи її тривалість.

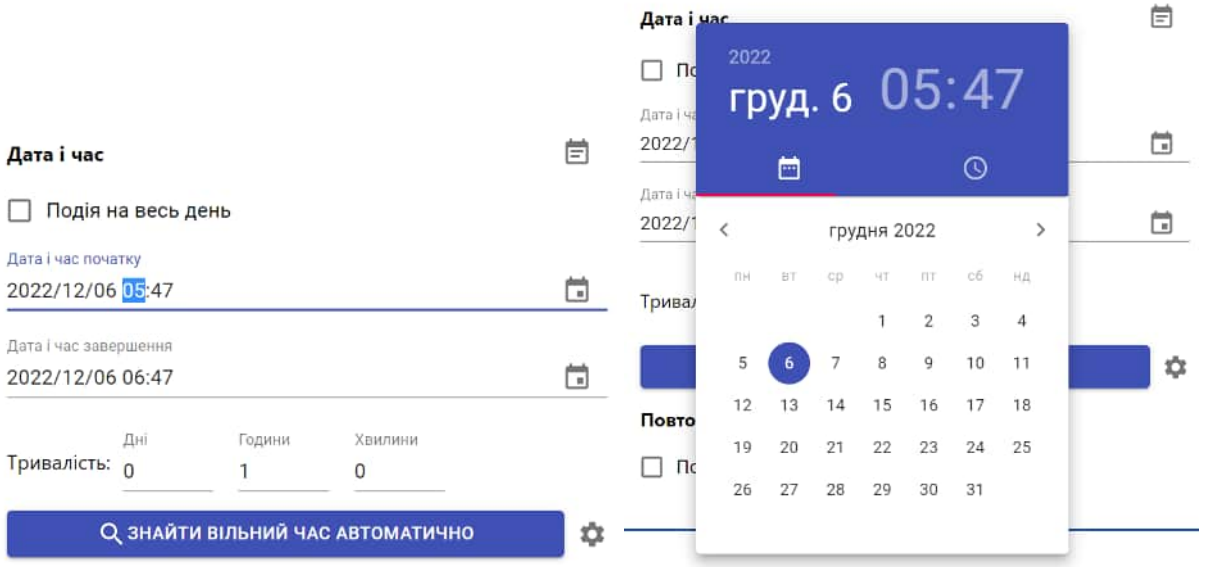

Рисунок 5.20 Налаштування часу проведення власної події

5.4 Автоматичне планування часу

Наприклад, ми хочемо змінити час події "Піти в магазин" автоматично. У нас є три події: з 08:35 по 09:35, з 09:00 по 10:00 та одна подія на яку нас запросили з 11:35 по 12:35. Відповідно цього дня ми маємо три вільні блоки різної тривалості, куди можемо розмістити нашу подію: з 07:26 (поточний час) по 08:35, з 10:00 по 11:35 та з 12:35 до кінця дня.

Зрозуміло, що подія має бути коротшою від тривалостей цих блоків, щоб поміститись в них. Нехай ми хочемо запланувати подію тривалістю 1 годину. Для цього обираємо подію "Піти в магазин" і встановлюємо потрібну нам тривалість:

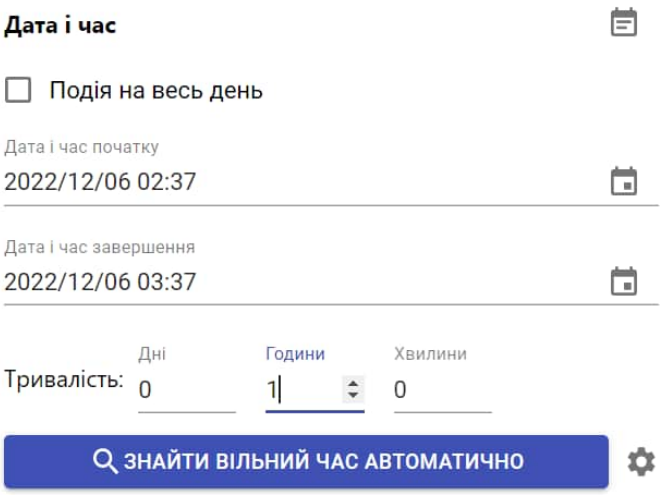

Рисунок 5.21 Вибір тривалості для події

Далі натискаємо кнопку "Знайти вільний час автоматично". Як бачимо, подія запланована на хвилину після поточного часу і має тривалість 1 годину, як ми й планували.

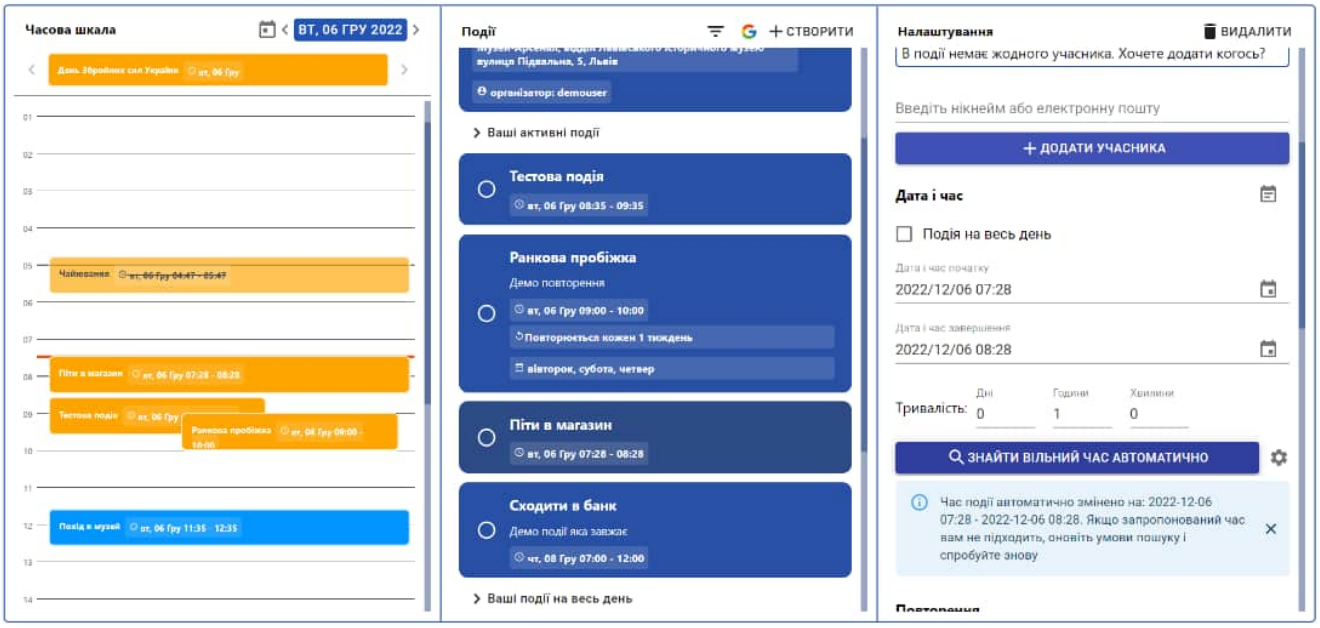

Рисунок 5.22 Подію заплановано автоматично у межах 07:28 - 08:28.

Тепер збільшимо тривалість до півтори години й перевіримо роботу програми у цьому випадку:

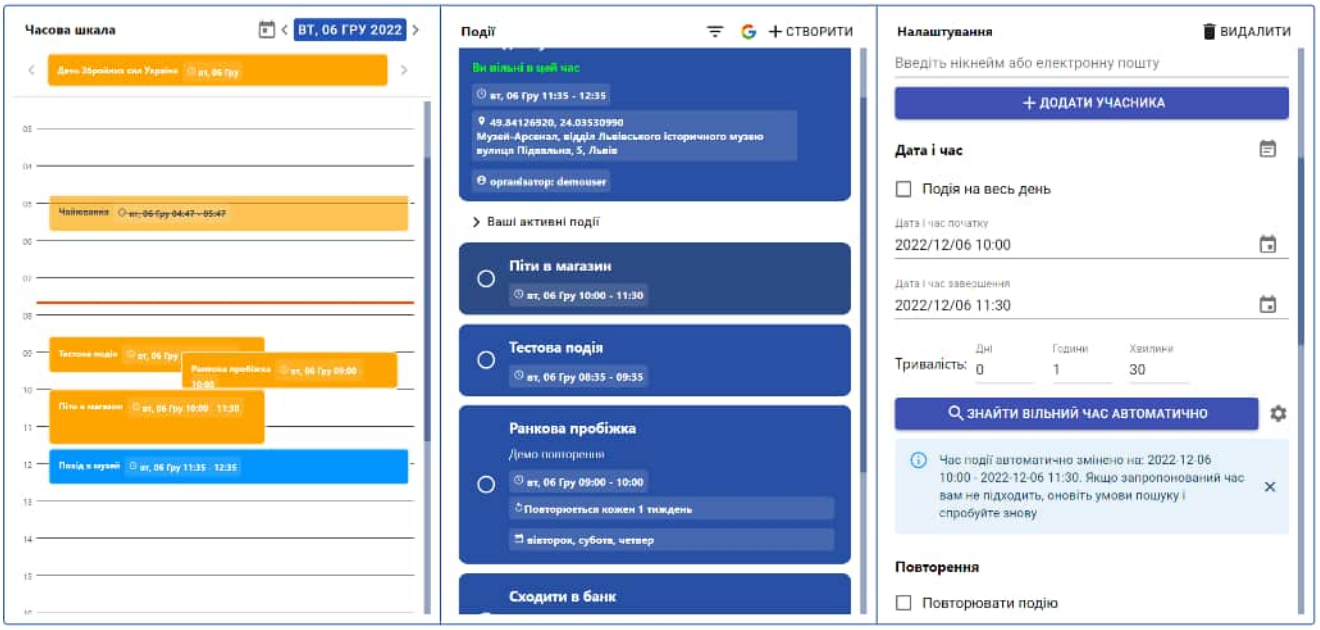

Рисунок 5.23 Подію заплановано автоматично у межах між 10:00 та 11:30

Як бачимо, проміжку між поточним часом і подією "Тестова подія" недостатньо, щоб запланувати подію на півтори години. Зате проміжку між подіями "Ранкова пробіжка" і "Похід в музей" цілком вистачає.

Також вистачає проміжку між подією "Похід в музей" і кінцем доби, щоб запланувати подію, наприклад, на 3 години:

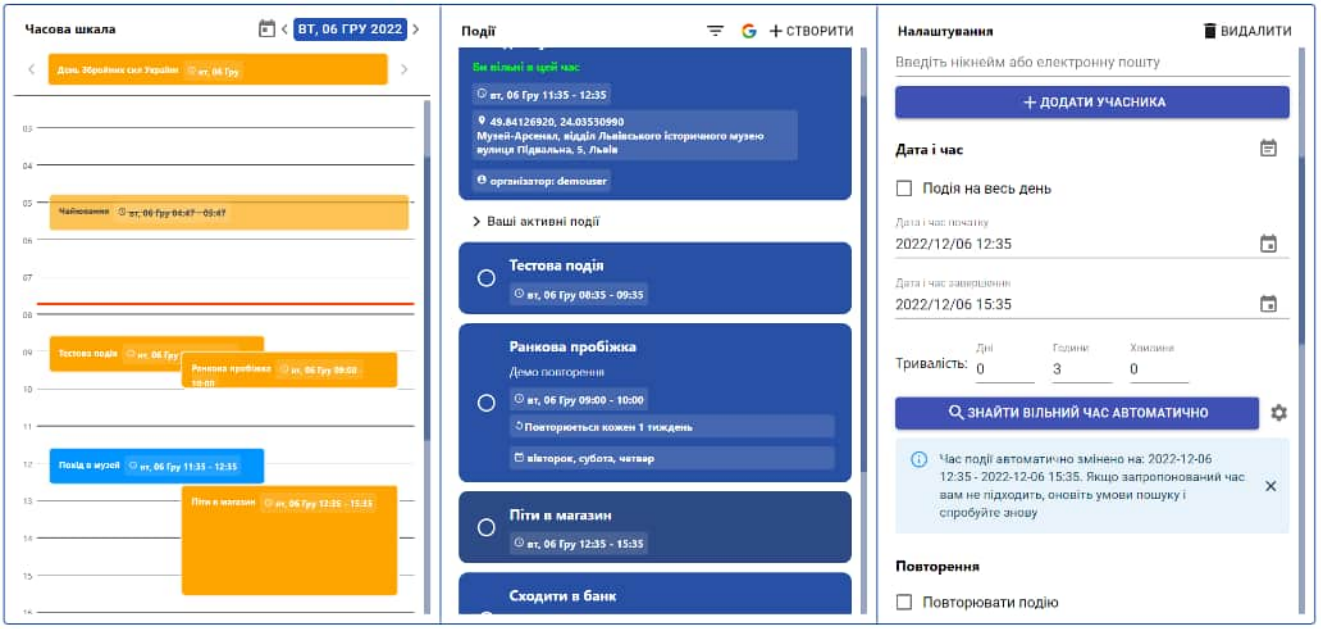

Рисунок 5.24 Подію заплановано автоматично у межах між 12:35 та 15:35

Якщо ж ми хочемо запланувати подію тривалістю 12 годин, нам не вистачить вільного проміжку в добі, куди б можна було її помістити. Перевіримо це:

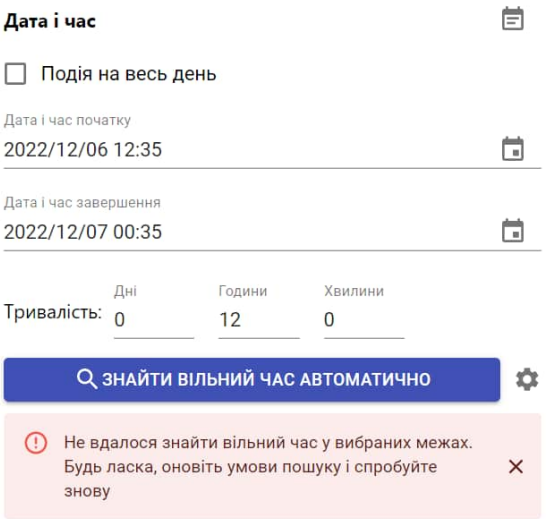

Рисунок 5.25 Подію не вдалось запланувати автоматично

Якщо ми хочемо запланувати подію не цього дня, а, наприклад, протягом доби після поточного часу, нам потрібно відкрити модальне вікно налаштувань автоматичного планування, натиснувши кнопку зі зображенням шестерні біля кнопки "Знайти вільний час автоматично":

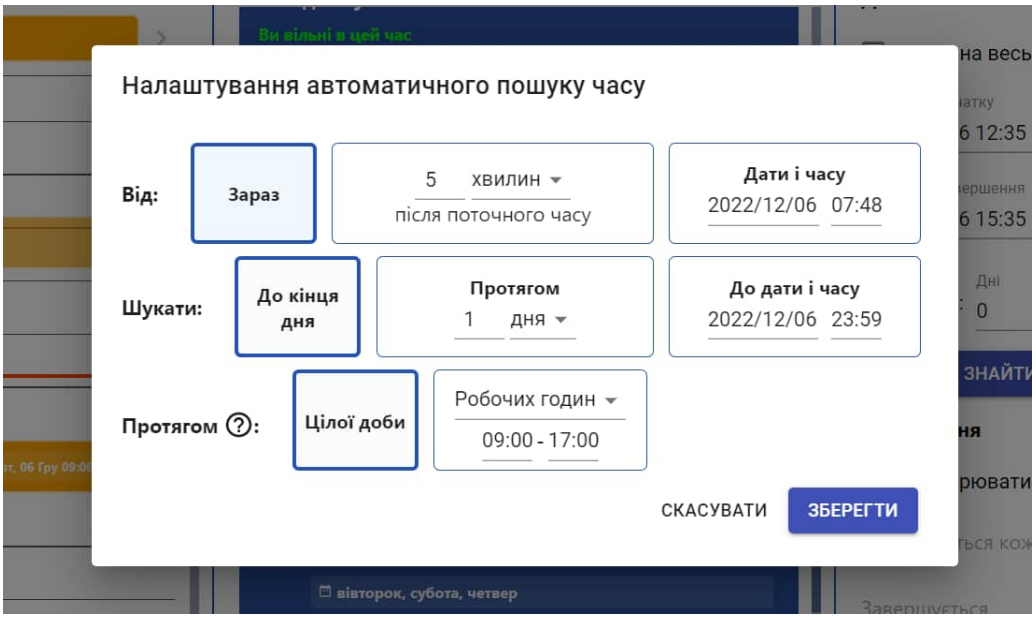

Рисунок 5.26 Налаштування автоматичного пошуку часу

Тут можна вибрати часові обмеження для автоматичного планування, а також часовий період у межах кожної доби, щоб програма не запланувала автоматично, наприклад подію "Похід в магазин" вночі.

Приміром будемо шукати час для події від 13:00 сьогоднішнього дня протягом двох днів, в ранішній час доби (06:00 - 12:00).

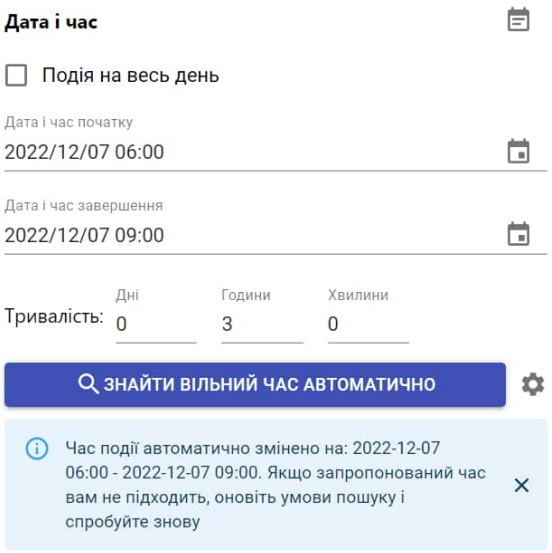

Рисунок 5.27 Подію заплановано автоматично на завтрашній ранок

Також варто розглянути випадок, коли ми хочемо запланувати подію у якої є учасники зі своїми розпорядками дня. Наприклад ми домовились піти з другом в музей. Раптом ми захотіли піти в кафе перед цим. Для цього створили подію "Похід в кафе" і запросили на неї користувача "demouser". Користувач має робочу зустріч перед тим як з нами зустрітись:

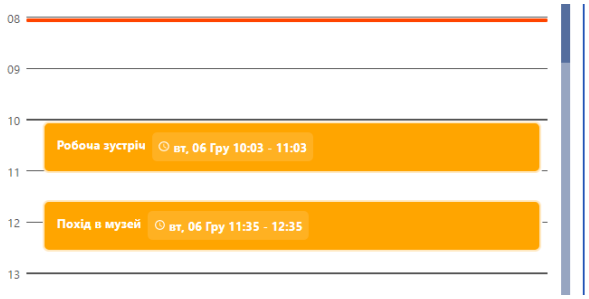

Рисунок 5.28 Часова шкала користувача "demouser"

Тривалість події "Похід в кафе" - півтори години. Цього б було достатньо щоб запланувати її на проміжку між подіями "Ранкова пробіжка" та "Похід в музей" (10:00 - 11:35) у нашому графіку:

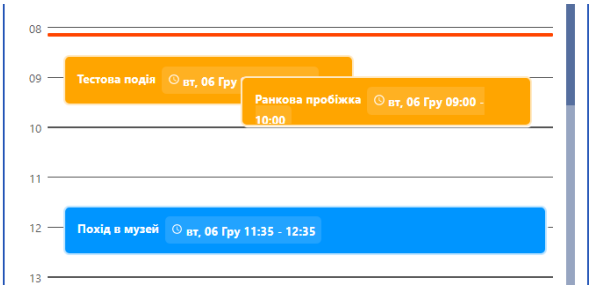

Рисунок 5.29 Часова шкала

Але оскільки користувач "demouser" зайнятий в цей час, похід в кафе вдасться запланували лише після походу в музей, що і станеться натиснувши кнопку "Знайти вільний час автоматично":

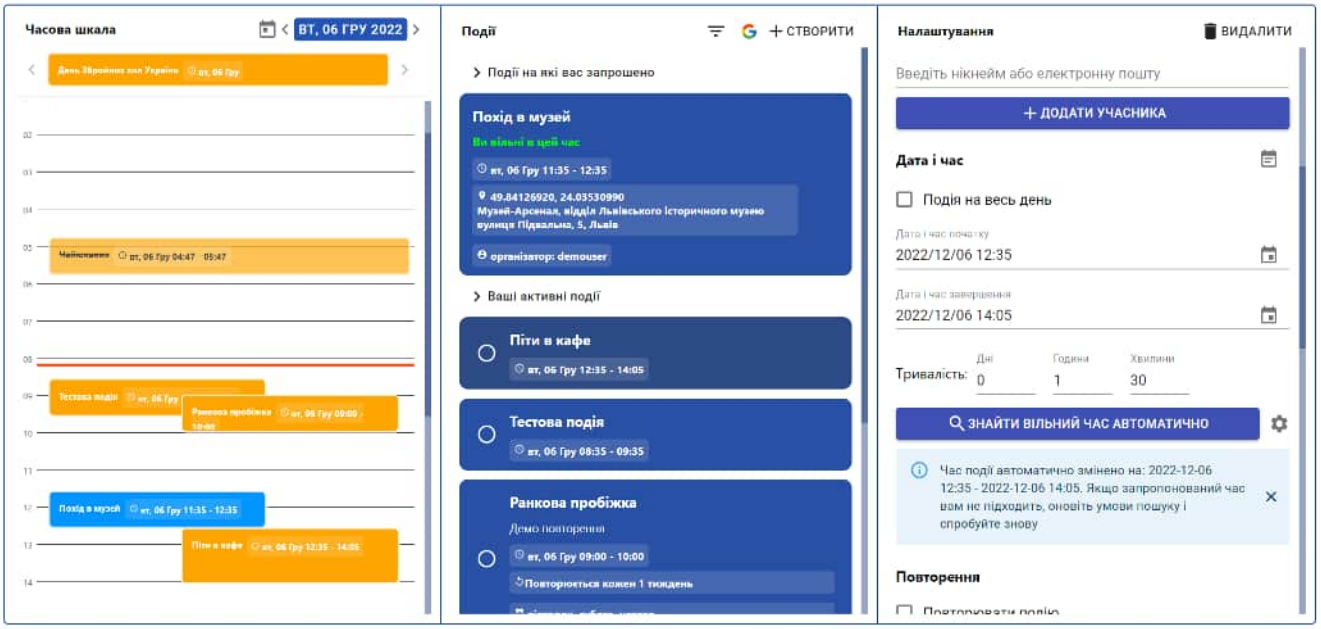

Рисунок 5.30 Подія "Похід в кафе" після автопланування часу

5.5 Планування часу подій, що повторюються

Під панеллю налаштування дати й часу знаходиться блок "Повторення", який дозволяє налаштувати події що відбуваються з певною періодичністю. Наприклад щодня, у будні, щотижня, в останню неділю місяця тощо.

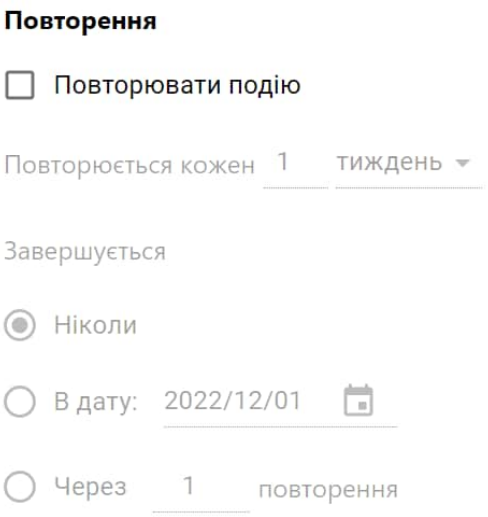

Рисунок 5.31 Повторення події не активовано

Існує 4 види періоду повторень: "щодня", "щотижня", "щомісяця" і "щороку". Між двома послідовними повтореннями події може пройти декілька таких періодів. Цю кількість можна налаштувати, так само як тип періоду та умову завершення повторень події.

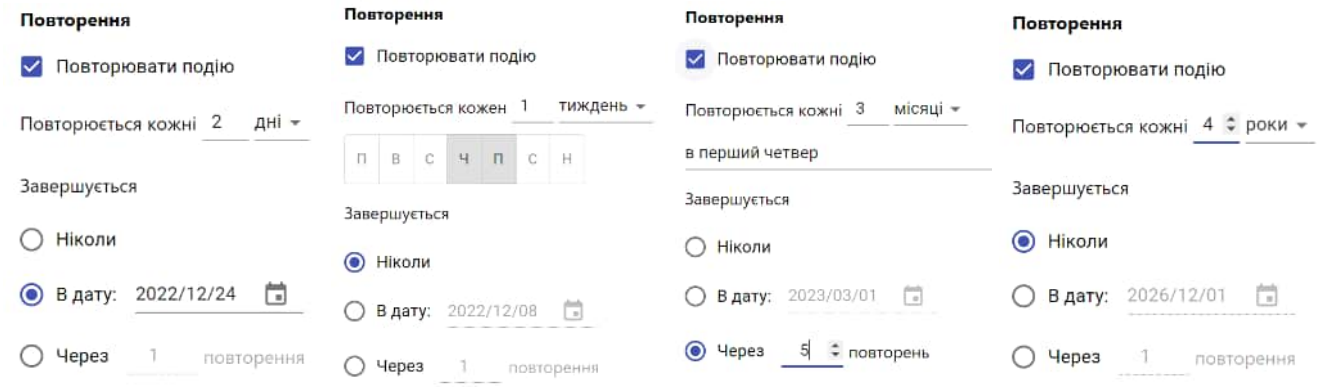

Рисунок 5.32 Різновиди повторень події

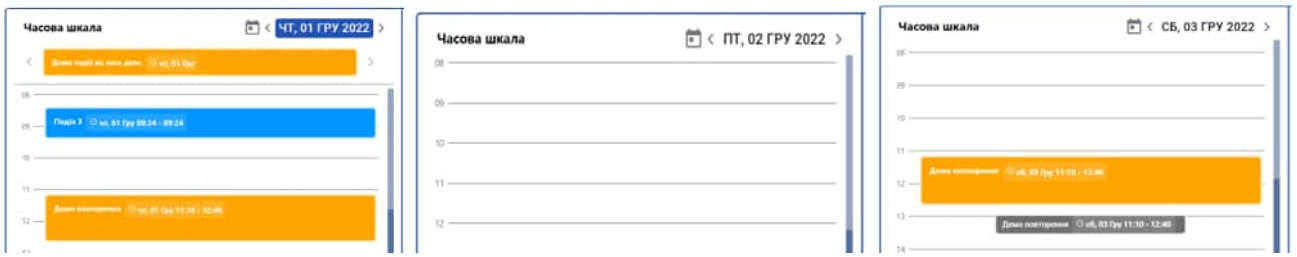

Рисунок 5.33 Подія, що повторюється раз у два дні

При виборі періоду повторень "тиждень" під ним з'явиться 7 кнопок, які відповідають дням тижня у які може відбуватись подія. Наприклад, вибравши дні від понеділка по п'ятницю, можна налаштувати подію, яка відбувається у будні.

Рисунок 5.34 Подія, що повторюється щотижня у четвер і п'ятницю

При виборі періоду повторень "місяць" під ним з'явиться випадний список, який має три можливі варіанти вибору залежно від дня початку події. Припустимо подія починається 28 листопада 2022 року. Це четвертий і останній понеділок місяця, відповідно випадний список буде містити три опції: "Щомісяця 28 числа", "Щомісяця в четвертий понеділок" та "Щомісяця в останній понеділок":

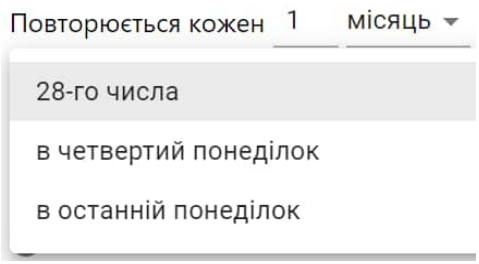

Рисунок 5.35 Варіанти повторення події щомісяця

Також є можливість відредагувати конкретне повторення події, наприклад якщо в якийсь день подія накладається з іншою. Для цього потрібно вибрати повторення на часовій шкалі.

Нехай маємо подію, що повторюється з такими налаштуваннями:

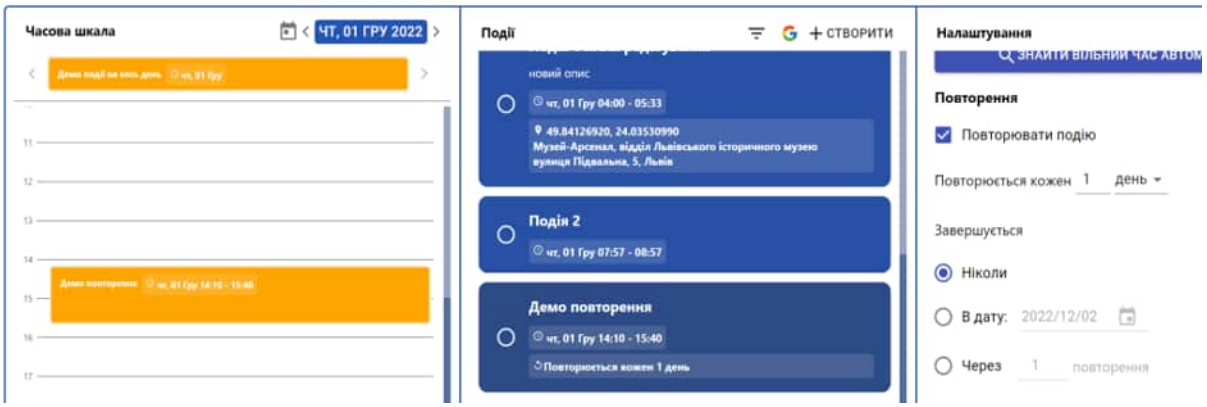

Рисунок 5.36 Подія, що повторюється і її налаштування

Натиснувши по події на часовій шкалі, ми побачимо модальне вікно з двома варіантами вибору, залежно від того що саме хочемо редагувати.

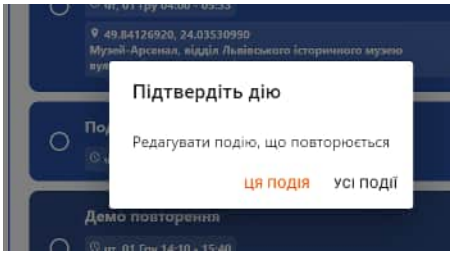

Рисунок 5.37 Вибір повторення події для редагування

Для того, щоб редагувати конкретне повторення події, слід вибрати пункт "Ця подія". Тепер ми можемо налаштувати час цієї події в меню "Налаштування".

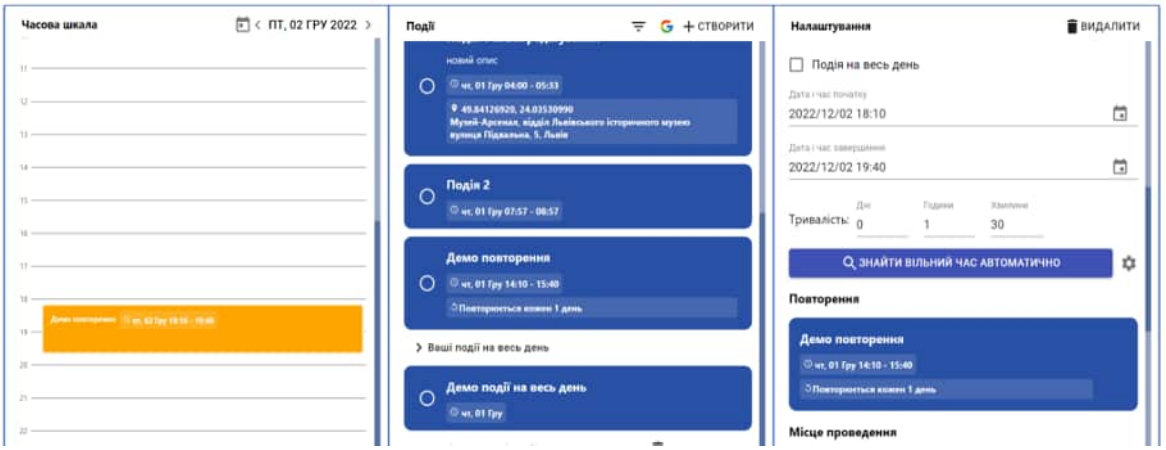

Рисунок 5.38 Редагування повторення події окремо від решти

У блоці повторення можемо бачити блок з головною подією при натисканні на який, ми перейдемо до редагування усього ланцюжка подій і його періодичності.

Ми можемо запланувати час події, що повторюється автоматично, як і будь-якої події. Наприклад, ми хочемо почати бігати по вечорах (18:00 - 23:59) замість ранків. Ми бігаємо три рази на тиждень: у вівторок, четвер та суботу. Якщо в якісь дні у нас будуть важливіші справи, ми хочемо перенести біг на інший час цього ж дня. Наприклад сьогодні вівторок, а у четвер о 18 35 у нас буде зустріч випускників.

Спочатку налаштуємо автоматичне планування:

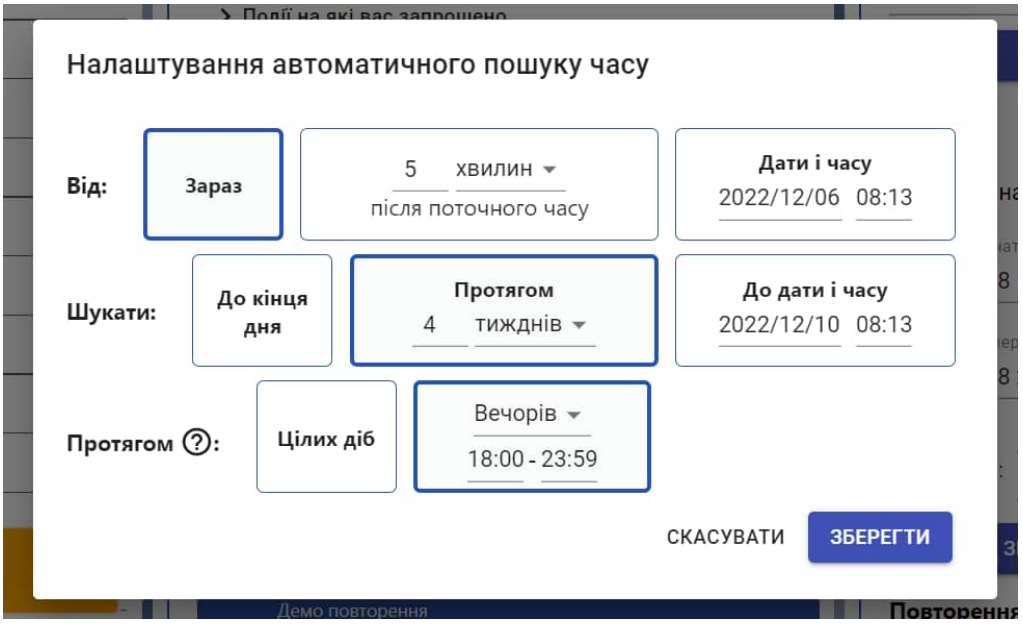

Рисунок 5.39 Налаштування автоматичного пошуку часу

Спробуємо знайти новий час для події натиснувши кнопку "Знайти вільний час автоматично":

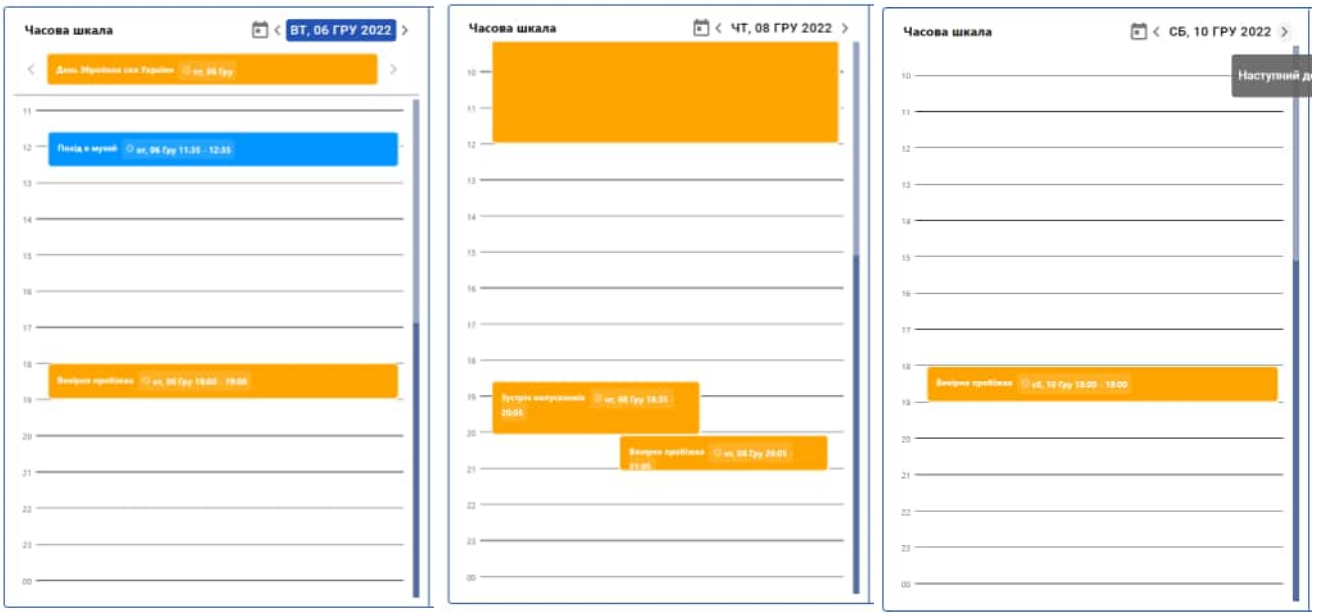

Рисунок 5.40 Час для події яка повторюється змінено автоматично

Якщо в якийсь день не вдасться запланувати повторення події, ми його просто скасовуємо:

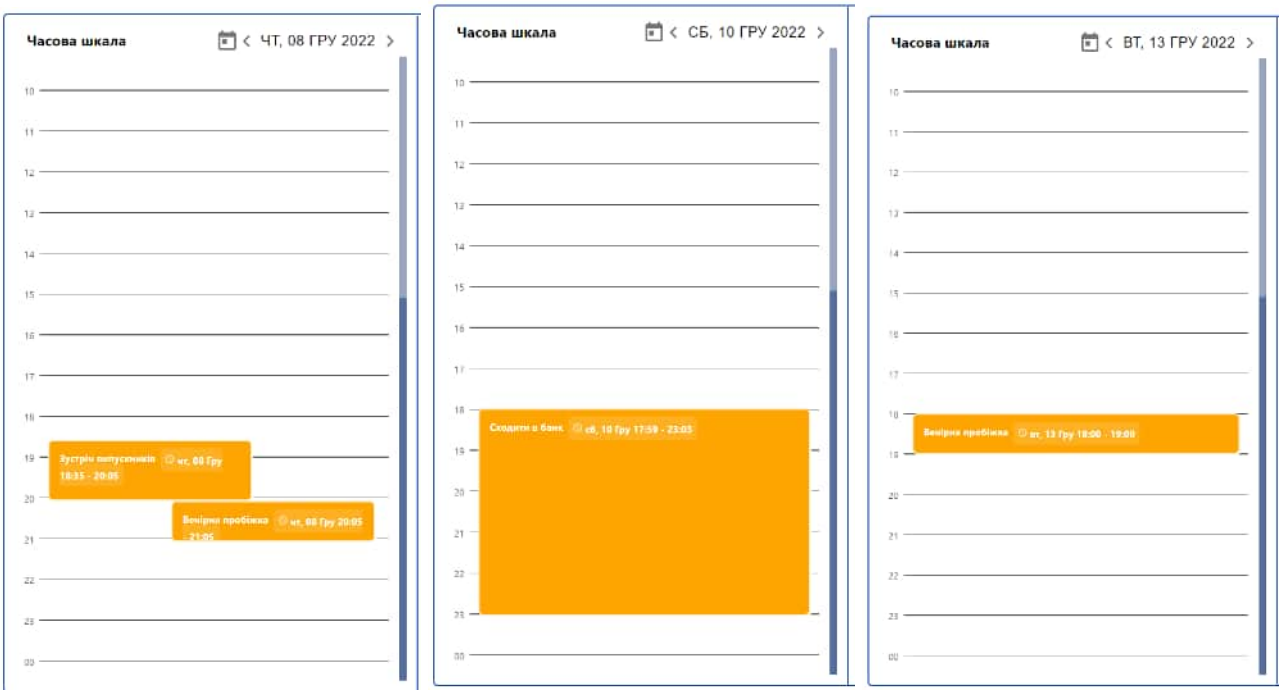

Рисунок 5.41 Повторення події скасоване в суботу, 10 грудня

5.6 Додаткові властивості системи

Сайт має меню налаштувань у верхньому рядку перемикання між сторінками. Налаштувань всього два: вибір мови та вибір формату часу.

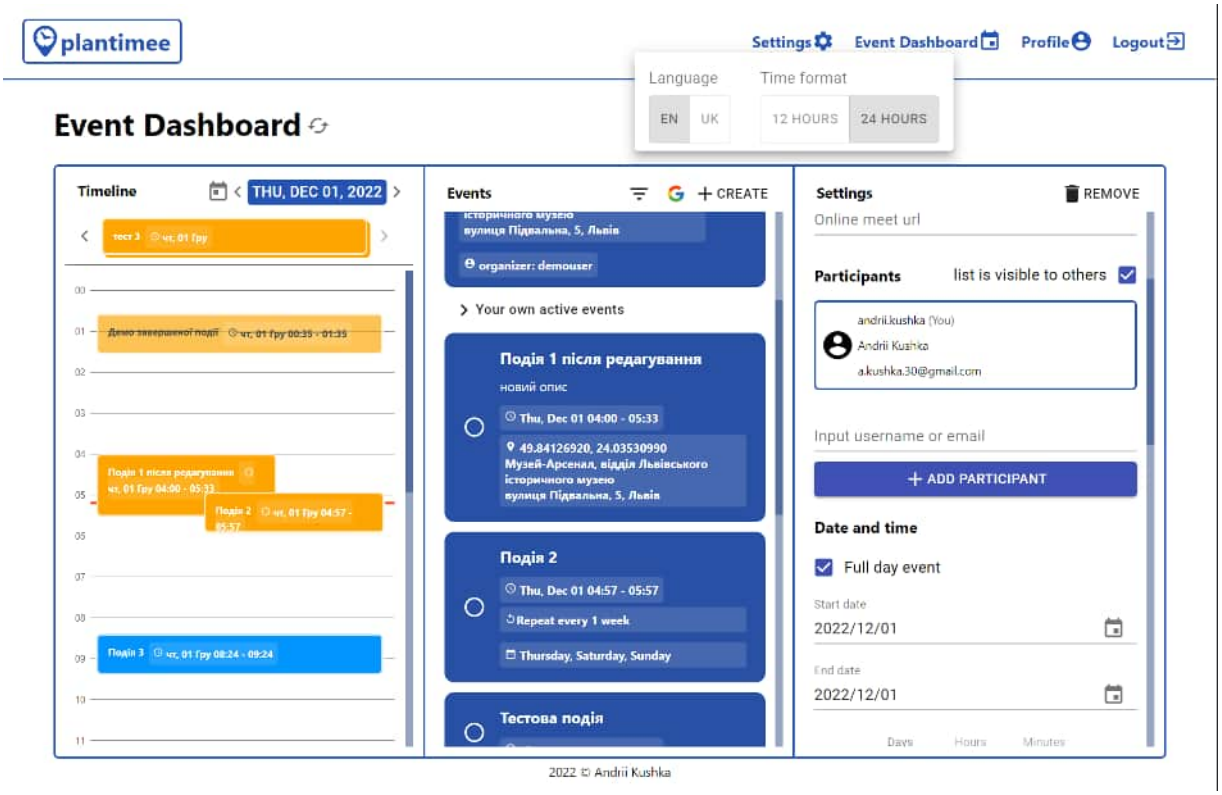

Рисунок 5.42 Англомовна версія сайту

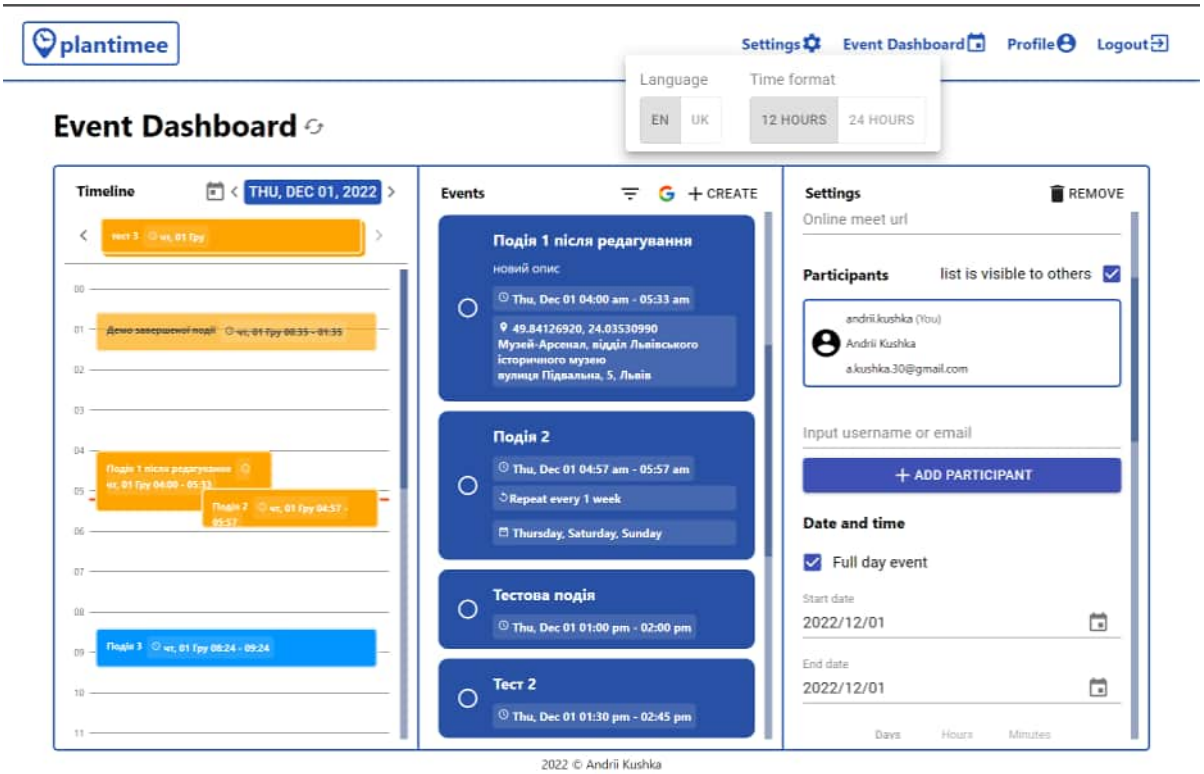

Рисунок 5.43 Застосовано 12-ти годинний часовий формат

У колонці "Події" можна побачити кнопку з іконкою Google. При натисканні на цю кнопку користувач побачить випадне меню синхронізації з Google Календарем. Для того щоб авторизуватися за допомогою Google достатньо натиснути кнопку "Увійти за допомогою Google" і у модальному вікні вибрати свій обліковий запис Google, а також надати всі необхідні дозволи для вебзастосунку.

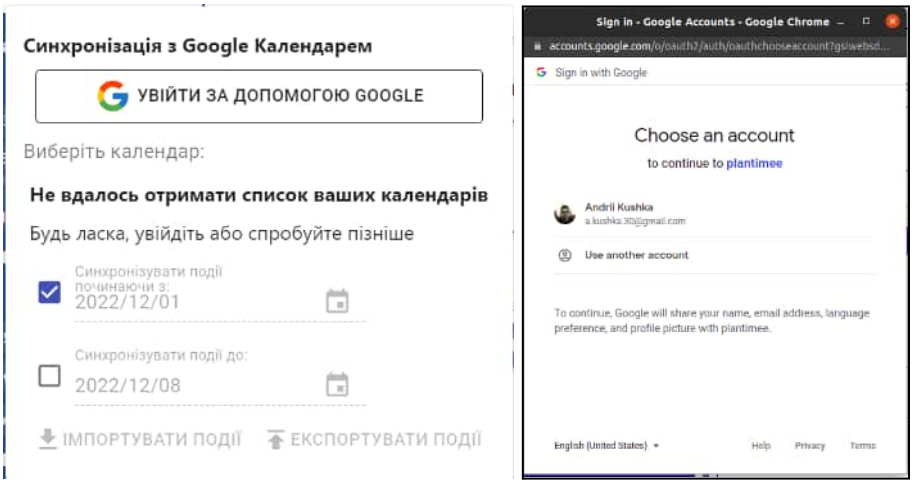

Рисунок 5.44 Вхід за допомогою Google

Після входу користувач побачить свою фотографію та ім'я в меню "Синхронізація з Google Календарем". Також там буде доступний список календарів користувача, фільтри по даті, а також дві кнопки "Імпортувати події" та "Експортувати події".

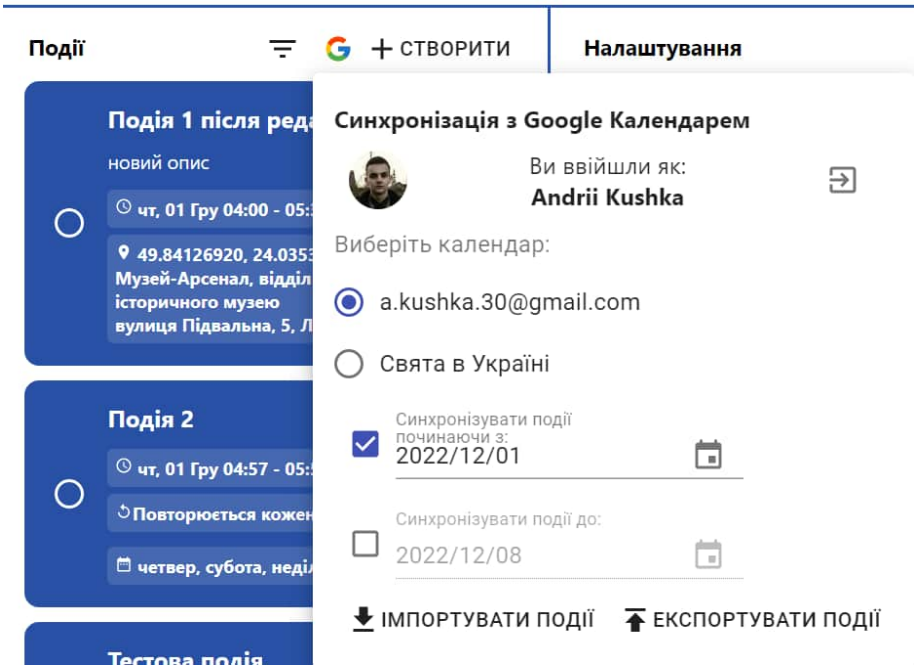

Рисунок 5.46 Синхронізація з Google Календарем

Для того щоб імпортувати події з Google Календаря, потрібно вибрати конкретний календар (їх може бути декілька), обрати фільтр по даті й натиснути кнопку "Імпортувати події". За кілька секунд вони з'являться в списку подій, а також на часовій шкалі.

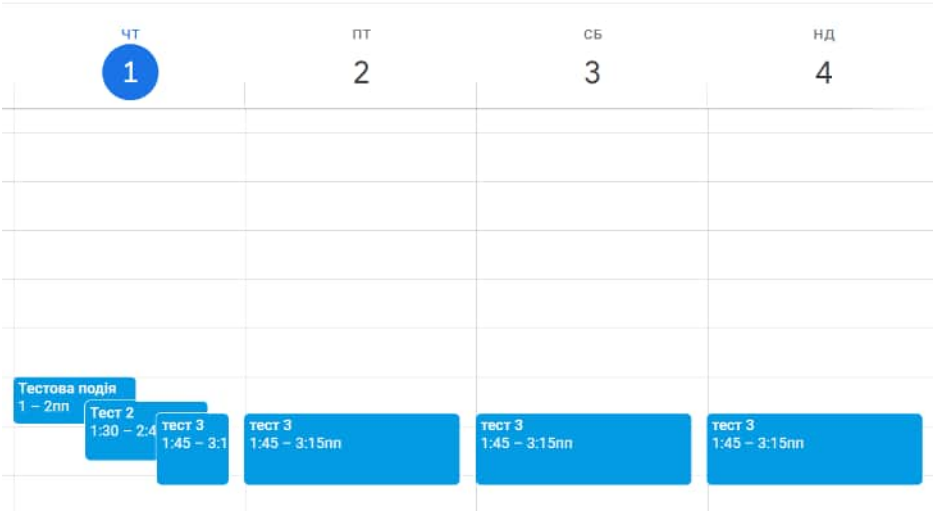

Рисунок 5.47 Дві звичайні й одна подія, що повторюється в Google Календарі

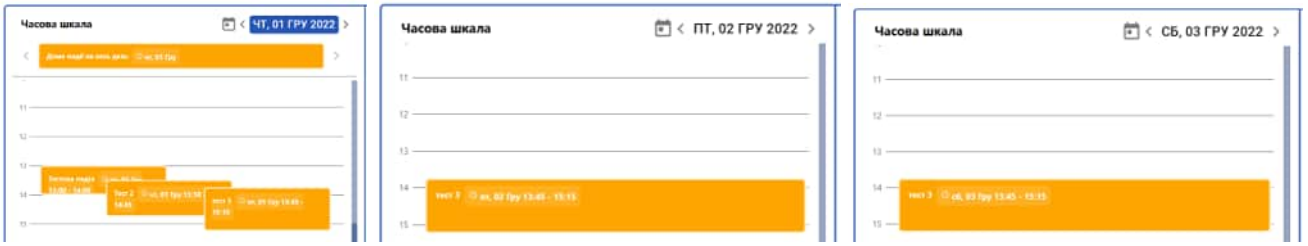

Рисунок 5.48 Події імпортовано з Google Календаря

Події, що були імпортовані з гугл календаря зберігають унікальний ідентифікатор і при наступній синхронізації не створюють копії, а оновлюються.

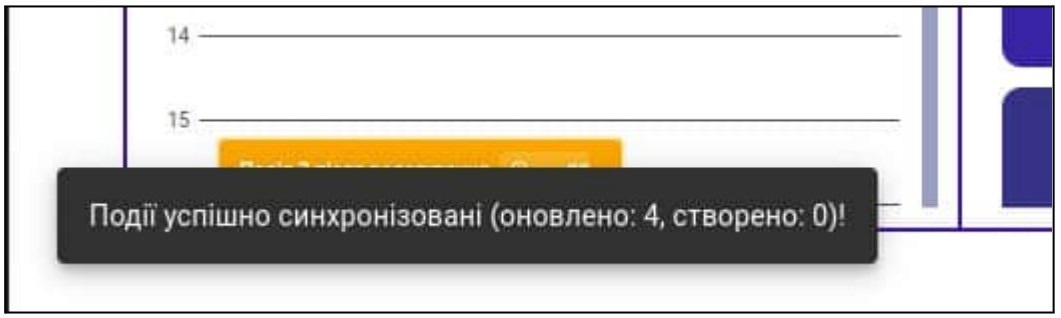

Рисунок 5.49 Сповіщення після успішної синхронізації

Нехай у нас є дві активні події, які ми хочемо експортувати на свій Google Календар.

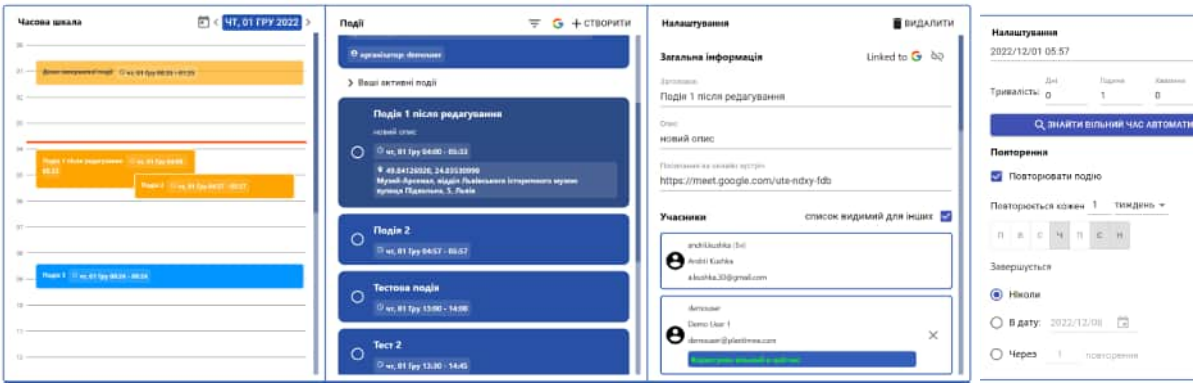

Рисунок 5.50 Події обрані для експорту

У меню синхронізації вибираємо бажані фільтри по даті та натискаємо кнопку: "Експортувати події". Після чого ці дві події можна побачити на Google Календарі, а у застосунку появиться сповіщення про успішну синхронізацію.

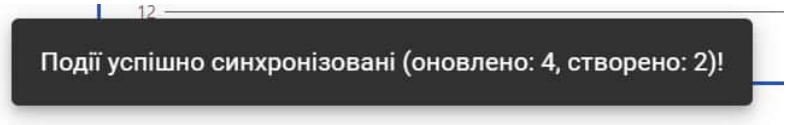

Рисунок 5.51 Сповіщення після успішної синхронізації

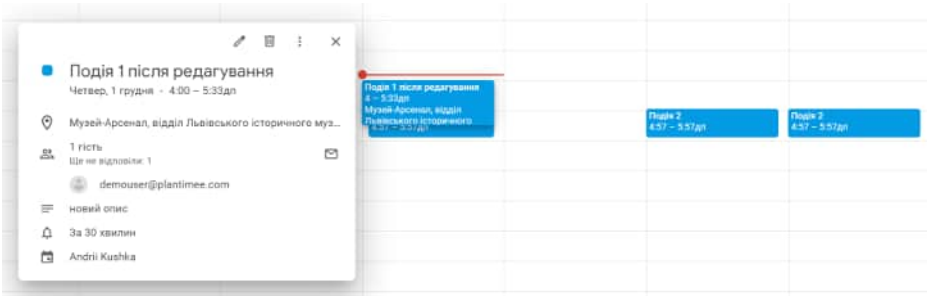

Рисунок 5.52 Експортовані події у Google Календарі

Як можна бачити при експорті збереглись назви подій, опис, місце проведення, періодичність, а також учасники події, які також отримали запрошення взяти участь у події на Google Календарі.

Як уже було сказано у попередньому розділі, використані технології дозволили реалізувати вебзастосунок, яким однаково зручно користуватися як на стаціонарному комп’ютері чи ноутбуці, так і на мобільних пристроях.

Розглянемо як виглядають різні сторінки застосунку на двох видах мобільних пристроїв: смартфон Pixel 2 XL та планшет iPad. Для цього використаємо симуляцію пристроїв в інструментах розробника браузера Gooogle Chrome.

Смартфон:

Рис. 5.53 Домашня сторінка на екрані мобільного пристрою

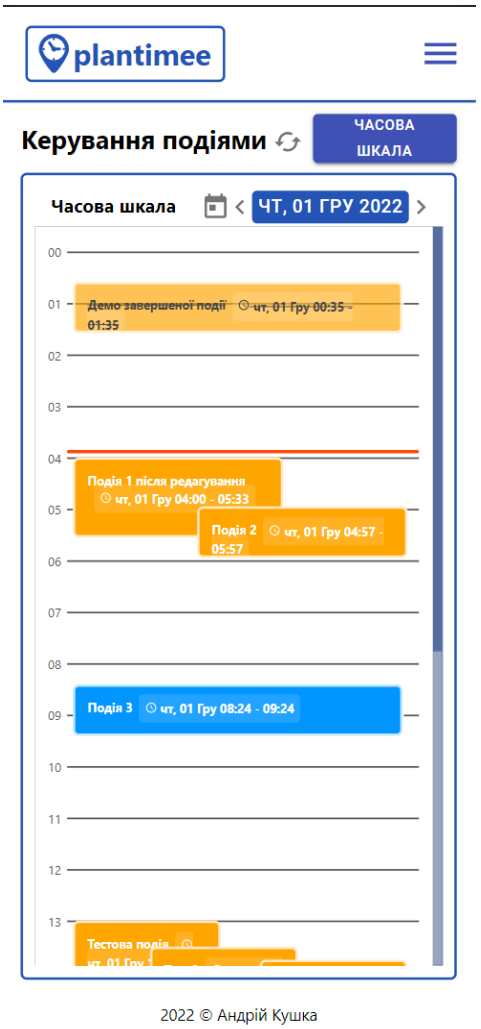

Рис 5.54 Сторінка з часовою шкалою на екрані мобільного пристрою

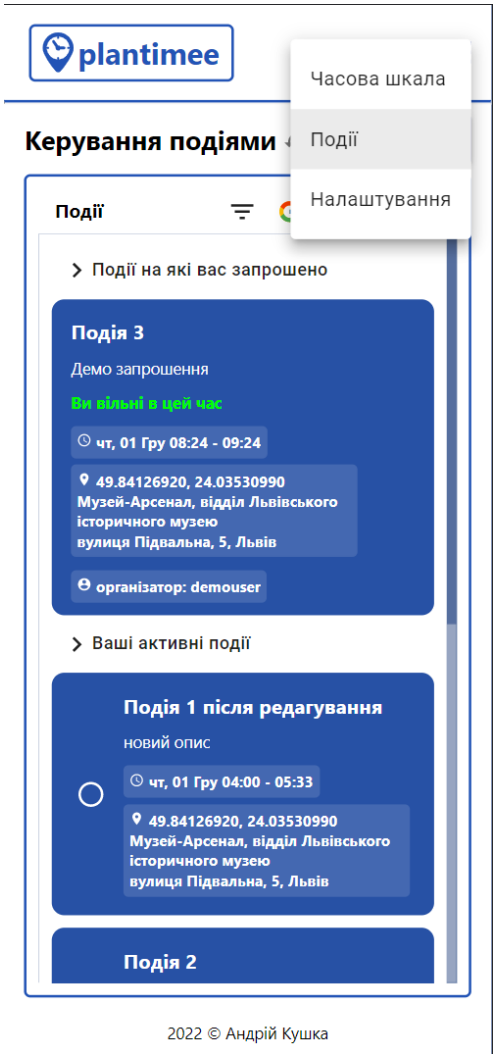

Рис. 5.55 Список подій з відкритим меню панелі керування

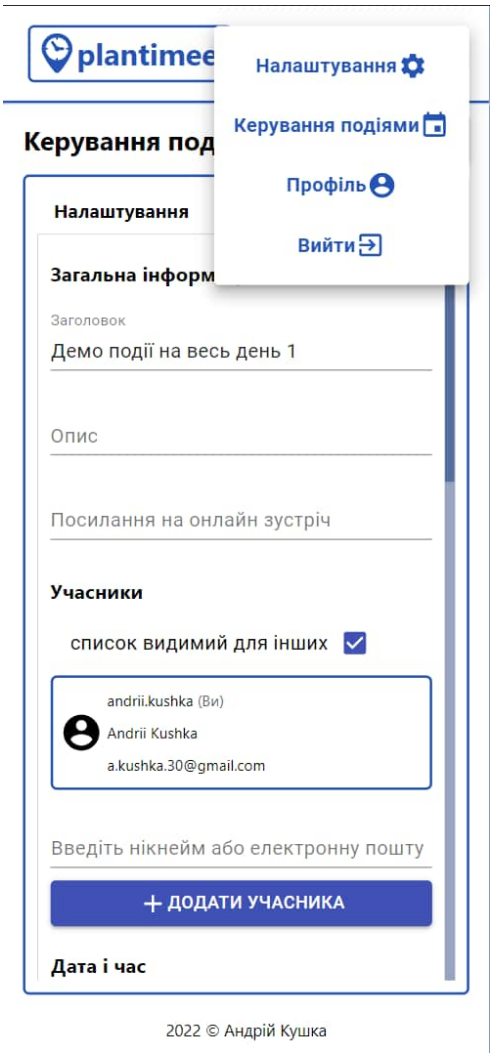

Рис. 5.56 Налаштування події з відкритим загальним меню

Планшет:

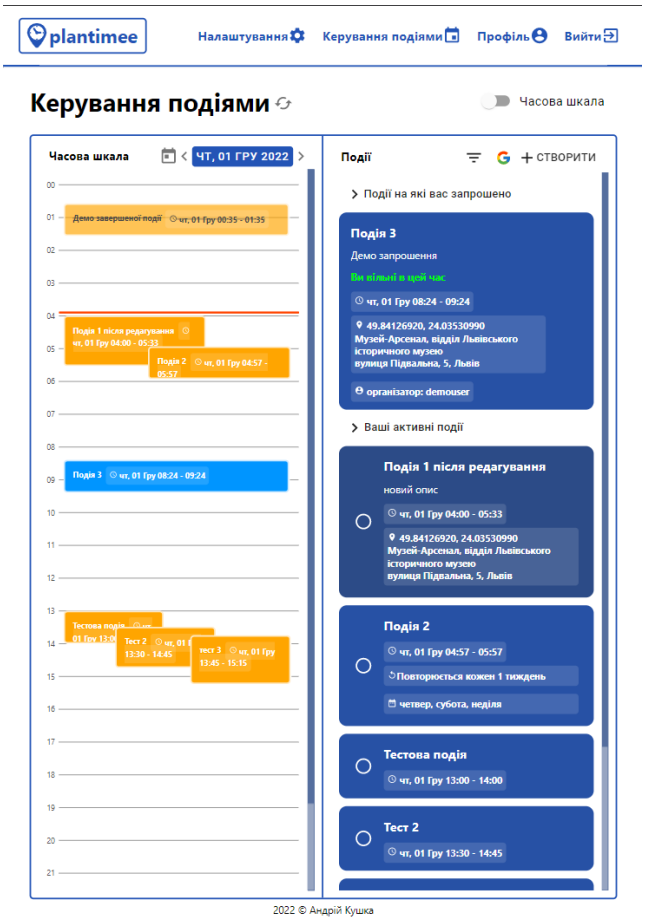

Рисунок 5.57 Часова шкала та список подій на екрані планшета

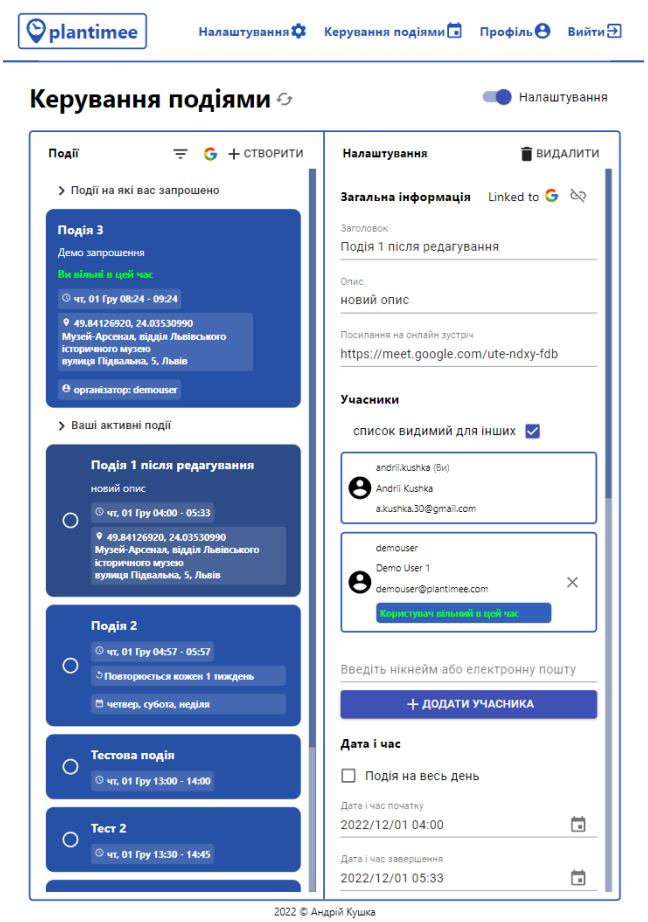

Рисунок 5.58 Список подій та меню налаштувань на екрані планшета

ВИСНОВКИ

У цій роботі ми дослідили наявні програми планери (Google Календар, Oulook Calendar і Microsoft Project) та визначили їхні переваги й недоліки. На основі отриманої інформації побудували власну систему, яка поєднує у собі простоту додатка для особистого використання і важливий функціонал системи для планування великих проєктів – автоматичне планування завдань.

Основними перевагами даного застосунку є такі:

а) можливість запланувати час звичайної події автоматично;

б) можливість запланувати автоматично час події, яка повторюється;

в) можливість автоматично запланувати місце проведення події використовуючи один з наявних алгоритмів.

Отже, розроблена система у вигляді вебдодатку значно полегшує користувачу планування подій, адже автоматичне планування дозволяє визначити найбільш відповідний час чи місце ледь не за одне натискання кнопки. Також при цьому зберігається конфіденційність розпорядків дня та місцеперебування гостей події, оскільки про них знає лише алгоритм.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Браян Трейсі. Зроби це зараз. 21 чудовий спосіб зробити більше за менший час. Харків: Книжковий Клуб «Клуб Сімейного Дозвілля», 2016. 146 с.
2. Ryder Carroll. The Bullet Journal Method: Track the Past, Order the Present, Design the Future. New York: Portfolio/Penguin, 2018. 320 p.
3. Malik Ghallab, Dana Nau and Paolo Traverso. Automated Planning: Theory & Practice. Trento: Morgan Kauffman Publishers, 2004. 609 p.
4. Google Календар. URL: https://calendar.google.com/calendar (дата звернення: 20.09.2022)
5. Calendar - Outlook URL: https://outlook.live.com/calendar (дата звернення: 23.09.2022)
6. Microsoft Project. URL: https://www.microsoft.com/uk-ua/microsoft-365/project/project-management-software (дата звернення: 19.03.2022)
7. Center of mass - Wikipedia URL: https://en.wikipedia.org/wiki/Center_of_mass (дата звернення: 04.10.2022)
8. Geometric median - Wikipedia URL: https://en.wikipedia.org/wiki/Geometric_median (дата звернення: 06.10.2022)
9. Посібник: знайомство з React. React JavaScript-бібліотека для створення користувацьких інтерфейсів. URL: https://uk.reactjs.org/tutorial/tutorial.html (дата звернення: 27.04.2022)
10. Односторінковий застосунок. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Односторінковий_застосунок (дата звернення: 29.10. 2022)
11. React components for faster and easier web development. Build your own design system, or start with Material Design. Material-UI. URL: https://material-ui.com/ (дата звернення: 03.11. 2022)
12. Getting Started. Promise based HTTP client for the browser and node.js. Axios. URL: https://axios-http.com/docs/intro (дата звернення: 13.10. 2022)

13. Про Node.js®. Nodejs. URL: https://nodejs.org/uk/about/ (дата звернення: 29.09. 2022)
14. Fast, unopinionated, minimalist web framework for Node.js. Express. URL: https://expressjs.com/ (дата звернення: 25.10. 2022)
15. What is PostgreSQL? PostgreSQL. URL: https://www.postgresql.org/about/ (дата звернення: 11.11. 2022)
16. Plantimee. URL: https://plantimee.onrender.com (дата звернення: 01.12.2022)
17. What is CI/CD? Continuous Integration & Continuous Delivery in 2020, Praful Dhabekar Medium URL: https://medium.com/devops-dudes/what-is-ci-cd-continuous-integration-continuous-delivery-in-2020-988765f5d116 (дата звернення: 30.11.2022)
18. About Render, Cloud Hosting for Developers | Render URL: https://render.com/about (дата звернення: 27.10. 2022)
19. Google Maps Platform - Location and Mapping Solutions URL: https://mapsplatform.google.com (дата звернення: 15.11.2022)
20. Google Calendar API - Google Developers URL: https://developers.google.com/calendar/api (дата звернення: 20.11.2022)

Додаток А. Код функції автоматичного планування часу простої події

```
const { attendees } = event;

let start = new Date();
let { dayEnd: end } = getDayBounds(start);
if (fromDate) start = fromDate;
if (toDate) end = toDate;

let raw_tasks = [];

const addTask = (eventData, startDate, endDate) => {
 if (eventData.startTime && eventData.endTime &&
!eventData.isFullDay && start < endDate && startDate <
endDate && startDate < end) {
   let taskStart = startDate;
   let taskEnd = endDate;
   if (startDate < start) taskStart = start;
   if (end < endDate) taskEnd = end;
   raw_tasks.push([taskStart.getTime(),
taskEnd.getTime()]);
 }
}

const collectTask = (eventData) => {
 if (eventData.id === event.id ||
eventData.recurrentEventId === event.id) return;
 let startDate = new Date(eventData.startTime);
 let endDate = new Date(eventData.endTime);

 addTask(eventData, startDate, endDate);
 if (eventData.repeatEnabled) {
   let from = new Date(start);
   while (from < end) {
     if (checkIfDayHasEventOccurrence(from, eventData)) {
       ({ startTime: startDate, endTime: endDate } =
getEventOccurrence(event, from));

       addTask(eventData, startDate, endDate);
     }
     from.setDate(from.getDate() + 1);
   }
 }
}
```

```
events.forEach(collectTask);
own_events.forEach(collectTask);
attendees.forEach(attendee => {
 attendee.events.forEach(collectTask);
 attendee.own_events.forEach(collectTask);
})

const dayTimeLimitingTasks = (fromTime && toTime) ? (
 getDayTimeLimitingTasks(start, end, fromTime, toTime)
) : [];

const tasks = _.uniqBy([
 ...raw_tasks,
 ...dayTimeLimitingTasks
], item => JSON.stringify(item));

let segments = [[start.getTime(), end.getTime()]];

tasks.forEach(task => {
 segments = segments.reduce((result, segment) => {
   if (task[1] <= segment[0] || segment[1] <= task[0]) {
     result.push([segment[0], segment[1]]);
   } else {
     if (segment[0] < task[0] && task[0] < segment[1]) {
       result.push([segment[0], task[0]]);
     }
     if (segment[0] < task[1] && task[1] < segment[1]) {
       result.push([task[1], segment[1]]);
     }
   }

   return result;
 }, []);
});

const sortedSegments = _.sortBy(segments, seg => seg[0]);

const segmentToInsertIn = sortedSegments.find(seg =>
msDuration < (seg[1] - seg[0]+0.05));
if (segmentToInsertIn) {
 event.startTime = new Date(segmentToInsertIn[0]);
 event.endTime = new Date(segmentToInsertIn[0] +
msDuration);
 const savedEvent = await event.save();
```

Додаток Б. Код функції автоматичного планування часу події, що повторюється

```
if (event.repeatEnabled) {
 segmentToInsertIn[0] = segmentToInsertIn[0] + msDuration;

 await db.Event.destroy({
   where: {
     recurrentEventId: event.id,
     startTime: {
       [Op.gt]: start,
     },
     endTime: {
       [Op.lt]: end,
     }
   }
 });

 let newEventOccurrences = [];
 let from = new Date(start);
 while (from < end) {
   const { dayStart: possibleDayStart, dayEnd:
possibleDayEnd } = getDayBounds(from);
   const dayStart = Math.max(start.getTime(),
possibleDayStart.getTime());
   const dayEnd = Math.min(possibleDayEnd.getTime(),
end.getTime());
   if (checkIfDayHasEventOccurrence(from, event)) {
     const { startTime: startDate, endTime: endDate } =
getEventOccurrence(event, from);

     let prevSegment = null;
     let foundStartTime = null;
     for (const seg of sortedSegments) {
       const segStart = Math.max(seg[0], dayStart);
       const segEnd = Math.min(seg[1], dayEnd);
       if ((segEnd - segStart) < msDuration) {}
       else if (segStart < startDate && endDate < segEnd) {
         foundStartTime = startDate.getTime();
         break;
       } else if (segEnd < startDate) {
         prevSegment = [segStart, segEnd];
       } else if (endDate < segStart) {
         if (prevSegment) {
           const diffToPrev = startDate - prevSegment[1];
```

```
            const diffToCurr = segStart - endDate;
            if (diffToPrev < diffToCurr) {
              foundStartTime = prevSegment[0];
              break;
            } else {
              foundStartTime = segStart;
              break;
            }
          } else {
            foundStartTime = segStart;
            break;
          }
        } else if (segStart < startDate || endDate < segEnd)
{
          foundStartTime = segStart;
          break;
        }
      }

      if (foundStartTime) {
        if (foundStartTime !== startDate.getTime()) {
          newEventOccurrences.push(
            buildSeparateInstance(event, startDate, new
Date(foundStartTime), new Date(foundStartTime +
msDuration))
          );
        }
      } else {
        newEventOccurrences.push(
          buildSeparateInstance(event, startDate, new
Date(startDate), new Date(startDate))
        );
      }
    }
    from.setDate(from.getDate() + 1);
  }

  if (newEventOccurrences.length) {
    try {
      await db.Event.bulkCreate(newEventOccurrences);
    } catch (e) {
      console.error(e);
    }
  }
}
```

Додаток В. Код функції автоматичного планування місця події

```
event.attendees.push(user)
const coords = event.attendees.map(({ latitude, longitude
}) => (
 latitude && longitude && ({ lat: latitude, lng: longitude
})
));
const mean = coords.reduce((result, point) => {
 result.lat += point.lat;
 result.lng += point.lng;
 return result;
}, { lat: 0, lng: 0 });
mean.lat /= coords.length || 1;
mean.lng /= coords.length || 1;

if (algorithm === 'MEAN') {
 return res.send(mean);
}

const min = coords.reduce((result, point) => {
 if (point.lat < result.lat) result.lat = point.lat;
 if (point.lng < result.lng) result.lng = point.lng;
 return result;
}, { ...coords[0] });
const max = coords.reduce((result, point) => {
 if (point.lat > result.lat) result.lat = point.lat;
 if (point.lng > result.lng) result.lng = point.lng;
 return result;
}, { ...coords[0] });
const stepPoint = {
 lat: (max.lat - min.lat),
 lng: (max.lng - min.lng),
};
const eps = 0.000001;

const distance = (a, b) => Math.sqrt((b.lat-a.lat) ** 2 +
(b.lng-a.lng) ** 2);
const distanceSum = (point, otherPoints) => (
 otherPoints.reduce((result, p) => result + distance(p,
point), 0)
);
const move = (point, diff, direction) => {
 const result = { ...point };
 switch (direction) {
   case 'up':
```

```
      result.lat += diff.lat;
      break;
    case 'down':
      result.lat -= diff.lat;
      break;
    case 'left':
      result.lng -= diff.lng;
      break;
    case 'right':
      result.lng += diff.lng;
      break;
  }
  return result;
}

const currPoint = { ...mean };

while(stepPoint.lat > eps && stepPoint.lng > eps) {
  const nextOptions = [
    currPoint,
    move(currPoint, stepPoint, 'up'),
    move(currPoint, stepPoint, 'down'),
    move(currPoint, stepPoint, 'left'),
    move(currPoint, stepPoint, 'right'),
  ];

  const distanceSums = nextOptions.map(point =>
distanceSum(point, coords));
  const bestOption = distanceSums.reduce((result, sum,
index) => {
    if (sum < result.value) {
      result = { index, value: sum };
    }
    return result;
  }, { index: 0, value: distanceSums[0] });
  currPoint.lat = nextOptions[bestOption.index].lat;
  currPoint.lng = nextOptions[bestOption.index].lng;

  stepPoint.lat = stepPoint.lat / 2;
  stepPoint.lng = stepPoint.lng / 2;
}

return res.send(currPoint);
```

Додаток Г. Код функції перевірки наявності повторення події в межах вибраного дня

```
let until;
if (event.repeatUntil) {
 ({ dayEnd: until } = getDayBounds(new
Date(event.repeatUntil)));
} else if (event.repeatCount) {
 const startDate = new Date(event.startTime);
 switch (event.repeatFreq) {
   case REPEAT_FREQ.DAILY:
     startDate.setDate(startDate.getDate() +
(event.repeatInterval || 1) * event.repeatCount);
     break;
   case REPEAT_FREQ.WEEKLY:
     startDate.setDate(startDate.getDate() +
(event.repeatInterval || 1) * 7 * event.repeatCount);
     break;
   case REPEAT_FREQ.MONTHLY:
     startDate.setMonth(startDate.getMonth() +
(event.repeatInterval || 1) * event.repeatCount);
     break;
   case REPEAT_FREQ.YEARLY:
     startDate.setFullYear(startDate.getFullYear() +
(event.repeatInterval || 1) * event.repeatCount);
     break;
   default: break;
 }
 ({ dayEnd: until } = getDayBounds(startDate));
}

const since = new Date(event.startTime);
switch (event.repeatFreq) {
 case REPEAT_FREQ.DAILY:
   since.setDate(since.getDate() + 1);
   break;
 case REPEAT_FREQ.WEEKLY:
   since.setDate(since.getDate() + 1);
   break;
 case REPEAT_FREQ.MONTHLY:
   since.setDate(since.getDate() + 1);
   since.setMonth(since.getMonth() + (event.repeatInterval
|| 1) - 1);
   break;
 case REPEAT_FREQ.YEARLY:
```

```
    since.setDate(since.getDate() + 1);
    since.setFullYear(since.getFullYear() +
(event.repeatInterval || 1) - 1);
    break;
  default: break;
}

const { dayStart, dayEnd } = getDayBounds(new Date(date));
const { dayStart: sinceStart } = getDayBounds(since);

if ((until && until < dayStart) || (dayEnd < sinceStart))
return false;

const startTime = new Date(event.startTime);
switch (event.repeatFreq) {
  case REPEAT_FREQ.DAILY:
    return checkRepeatIntervalMatch(startTime,
event.repeatInterval, event.repeatFreq, dayStart)
  case REPEAT_FREQ.WEEKLY:
    const daysOfWeek = event.repeatByDay &&
event.repeatByDay.split(',').map(dayTxt => (
      EnglishDays.findIndex(el => dayTxt ===
el.slice(0,2).toUpperCase())
    )) || [];
    daysOfWeek.sort();
    return daysOfWeek.includes(getWeekdayNumber(dayStart))
&& (
      checkRepeatIntervalMatch(startTime,
event.repeatInterval, event.repeatFreq, dayStart)
    );
  case REPEAT_FREQ.MONTHLY:
    if (!event.repeatByDay) {
      return dayStart.getDate() === startTime.getDate() && (
        checkRepeatIntervalMatch(startTime,
event.repeatInterval, event.repeatFreq, dayStart)
      );
    } else {
      const [weekdayOfMonth, dayOfWeek] =
getDayOfMonth(event.repeatByDay);
      const chosenDayOfWeek = getWeekdayNumber(dayStart);

      if (dayOfWeek !== chosenDayOfWeek) return false;

      return checkRepeatIntervalMatch(startTime,
event.repeatInterval, event.repeatFreq, dayStart) && ((
```

```
          weekdayOfMonth === -1 &&
countCertainDayTillEndOfMonth(dayOfWeek, dayStart) === 0
        ) || (
          countCertainDaySinceStartOfMonth(dayOfWeek,
dayStart) === weekdayOfMonth
        ));
    }
 case REPEAT_FREQ.YEARLY:
    return dayStart.getMonth() === startTime.getMonth() &&
dayStart.getDate() === startTime.getDate() && (
      checkRepeatIntervalMatch(startTime,
event.repeatInterval, event.repeatFreq, dayStart)
    );
 default:
    return false;
}
```