

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

**Пояснювальна записка**

до дипломного проекту (роботи)

бакалавра

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ

(назва теми)

ПОШУКУ І ОРГАНІЗАЦІЇ ЗБОРІВ. SOFTWARE TO

SEARCH FOR AND ORGANIZE GATHERINGS

Виконав(ла): студент(ка) 4 курсу, групи КНТ-110

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ДУБОВІКОВ Д.С.

(ПРІЗВИЩЕ та ініціали)

Керівник ЛЬОВКІН В.М.

(ПРІЗВИЩЕ та ініціали)

Рецензент ГОЛУБ Т.В.

(ПРІЗВИЩЕ та ініціали)

2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
Національний університет «Запорізька політехніка»

Факультет \_\_\_\_\_ КНТ \_\_\_\_\_  
Кафедра \_\_\_\_\_ програмних засобів \_\_\_\_\_  
Ступінь вищої освіти \_\_\_\_\_ бакалавр \_\_\_\_\_  
Спеціальність \_\_\_\_\_ 121 Інженерія програмного забезпечення \_\_\_\_\_  
(код і найменування)  
Освітня програма (спеціалізація) \_\_\_\_\_ Інженерія програмного забезпечення \_\_\_\_\_  
(назва освітньої програми (спеціалізації))

**ЗАТВЕРДЖУЮ**

Завідувач кафедри ПЗ, д.т.н, проф. \_\_\_\_\_  
Сергій СУББОТІН  
« \_\_\_\_\_ » \_\_\_\_\_ 2024 року

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

ДУБОВІКОВА Дмитра Сергійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення для пошуку і організації зборів  
Software to Search for and Organize Gatherings

керівник проєкту (роботи) к.т.н., доцент ЛЬОВКІН Валерій Миколайович,  
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від « 24 » квітня 2024 року № 168

2. Строк подання студентом проєкту (роботи) 10 червня 2024 року

3. Вихідні дані до проєкту (роботи) технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Проєктування програми. 3. Розробка програми. 4. Експлуатація та тестування програми.

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

Слайди презентації

## 6. Консультанти розділів проєкту (роботи)

| Розділ              | ПРІЗВИЩЕ, ініціали та посада консультанта | Підпис, дата   |                           |
|---------------------|-------------------------------------------|----------------|---------------------------|
|                     |                                           | завдання видав | прийняв виконане завдання |
| 1-4 Основна частина | ЛЬОВКІН В.М., доцент                      |                |                           |
| Нормоконтроль       | ДЕЙНЕГА Л.Ю., ст. викладач                |                |                           |
|                     |                                           |                |                           |
|                     |                                           |                |                           |
|                     |                                           |                |                           |
|                     |                                           |                |                           |
|                     |                                           |                |                           |

7. Дата видачі завдання « 15 » квітня 2024 року.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломного проєкту (роботи)                                             | Срок виконання етапів проєкту ( роботи ) | Примітка     |
|-------|--------------------------------------------------------------------------------------|------------------------------------------|--------------|
| 1     | Постановка завдання роботи                                                           | 1 тиждень                                | Завдання, ТЗ |
| 2     | Аналіз предметної області                                                            | 2 тиждень                                | Розділ 1     |
| 3     | Розробка архітектури програми                                                        | 3 тиждень                                | Розділ 2     |
| 4     | Розробка програми                                                                    | 4-5 тижні                                | Розділ 3     |
| 5     | Тестування та експериментальне дослідження програми                                  | 6 тиждень                                | Розділ 4     |
| 6     | Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування | 7 тиждень                                | Додатки      |
| 7     | Захист роботи                                                                        | 8 тиждень                                |              |
|       |                                                                                      |                                          |              |
|       |                                                                                      |                                          |              |
|       |                                                                                      |                                          |              |
|       |                                                                                      |                                          |              |
|       |                                                                                      |                                          |              |
|       |                                                                                      |                                          |              |
|       |                                                                                      |                                          |              |
|       |                                                                                      |                                          |              |

Студент(ка)

\_\_\_\_\_ Дмитро ДУБОВІКОВ  
( підпис ) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

\_\_\_\_\_ Валерій ЛЬОВКІН  
( підпис ) (Ім'я ПРІЗВИЩЕ)

## РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 90 с., 26 рис., 3 табл., 3 дод., 10 джерел.

ОРГАНІЗАЦІЯ ЗБОРІВ, ПОШУК ЗБОРІВ, СПІЛЬНОТА, ОРГАНІЗАТОРИ, ВІДВІДУВАЧІ, ВЕБДОДАТОК.

Об'єкт роботи – процеси пошуку і організації зборів людей. Предмет роботи – програмне забезпечення пошуку і організації зборів. Мета роботи – розробити програмне забезпечення пошуку і організації зборів для сприяння живій або дистанційній комунікації у спільнотах.

Матеріали, методи та технічні засоби: мова Python разом з фреймворком Django, шаблон проектування Model-template-view, засоби роботи з базою даних на основі системи PostgreSQL.

Результати. Розроблена програма пошуку і організації зборів надає можливості створення спільнот відвідувачів зборів, створення серій зборів та планування окремих зборів, додавання учасників до спільнот та реєстрації для проведення зборів, у тому числі створюючи черги очікування за обмеження кількості учасників та необхідності підтвердження участі, оцінювання зборів, що дозволяє визначати оцінки для серій зборів, демонстрування зборів, на які зареєструвався користувач, в яких він є організатором, створення персональних запитів після спільного відвідування зборів, коментування зборів і спільнот, організовуючи таким чином спілкування.

Висновки. Програму розроблено з врахуванням функціональних можливостей і особливостей аналогів, включаючи програми Meetup і Community, з виділенням відмінностей розробки, використано технології, які адекватно дозволили втілити запропоновані рішення в межах роботи.

Галузь використання. Інформування про організовані збори людей, в тому числі в межах спільнот.

## ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 90 pages, 26 figures, 3 tables, 3 appendixes, 10 sources.

GATHERING ORGANIZATION, GATHERING SEARCH,  
COMMUNITY, ORGANIZERS, VISITORS, WEB APPLICATION.

The object of the work is the process of finding and organizing gatherings of people. The subject of work is software for finding and organizing gatherings. The goal of the work is to develop software for finding and organizing gatherings to facilitate live or remote communication in communities.

Materials, methods and technical means: the Python language together with the Django framework, the Model-template-view design template, tools for working with a database based on the PostgreSQL system.

The results. The developed program for finding and organizing gatherings provides opportunities for creating communities, creating series of gatherings and planning individual gatherings, adding participants to communities and registering for gatherings, including creating waiting lines for limiting the number of participants and the need for confirmation of participation, evaluating meetings that allows participants to define ratings for a series of gatherings, show the gatherings a user has registered for, in which he is an organizer, create personal requests after attending gatherings together, comment on gatherings and communities, thus organizing communication.

Conclusions. The program was developed taking into account the functionality and features of analogues, including the Meetup and Kommunity programs, highlighting the differences in development, using technologies that adequately allowed the proposed solutions to be implemented within the scope of the work.

Field of use. Information about organized gatherings of people, including within communities.

## ЗМІСТ

|                                                                            |    |
|----------------------------------------------------------------------------|----|
|                                                                            | С. |
| Перелік скорочень та умовних познач.....                                   | 7  |
| Вступ.....                                                                 | 8  |
| 1 Аналіз предметної області.....                                           | 10 |
| 1.1 Огляд програм пошуку і організації зборів.....                         | 10 |
| 1.2 Порівняння аналогів для визначення характеристик власної розробки..... | 15 |
| 1.3 Висновки за розділом 1 .....                                           | 16 |
| 2 Проєктування програми.....                                               | 18 |
| 2.1 Вибір засобів проєктування і розробки .....                            | 18 |
| 2.2 Моделювання взаємодії користувача з програмою .....                    | 19 |
| 2.3 Проєктування структури бази даних.....                                 | 22 |
| 2.4 Висновки за розділом 2 .....                                           | 29 |
| 3 Розробка програми .....                                                  | 30 |
| 3.1 Визначення послідовності роботи користувача з програмою .....          | 30 |
| 3.2 Особливості програмної реалізації.....                                 | 31 |
| 3.3 Висновки за розділом 3 .....                                           | 47 |
| 4 Експлуатація та тестування програми .....                                | 48 |
| 4.1 Призначення програми .....                                             | 48 |
| 4.2 Умови виконання програми .....                                         | 48 |
| 4.3 Експлуатація програми.....                                             | 48 |
| 4.4 Тестування програми .....                                              | 52 |
| 4.5 Висновки за розділом 4 .....                                           | 55 |
| Висновки .....                                                             | 56 |
| Перелік джерел посилання .....                                             | 57 |
| Додаток А Технічне завдання .....                                          | 58 |
| Додаток Б Текст програми .....                                             | 63 |
| Додаток В Слайди презентації.....                                          | 84 |

## **ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК**

MTV – Model-template-view;

БД – база даних.

## ВСТУП

Сучасні умови комунікації потребують інших засобів надання інформації про збори людей, пошуку таких зборів. У великому місті може відбуватися велика кількість зборів, через що знайти потрібний достатньо важко.

Концертні заходи мають зазвичай власні вебсайти для пошуку і придбання квитків на такі заходи. Те ж стосується театральних подій, наприклад. Проте інформація про них дублюється також і в соціальних мережах, вебсайтах, що інформують про події у містах. У той же час через це некомерційні збори людей втрачають увагу. Знайти таку інформацію стає достатньо складно. Вона може бути представлена, наприклад, у соціальній мережі Facebook, або в іншій соціальній мережі. Проте віднайти потрібні збори достатньо складно у випадках, коли таких подій у місті відбувається багато. Окрім того в поточних умовах нерідко збори мають ще й дистанційну форму.

Усе це потребує певного впорядкування, а відповідно і окремого програмного інструменту, який би дозволив як виносити інформацію про свої збори, так і керувати ними, долучати учасників, виконувати пошук таких зборів за різними параметрами, які можуть включати не тільки назву, конкретне або загальне (місто) місце проведення зборів.

Усе це з одного боку вказує на важливість даної проблеми та створення відповідного програмного забезпечення. З іншого боку підкреслює, що вносити інформацію про збори повинні не тільки певні адміністратори або менеджери, а потенційно будь-які користувачів. При цьому потрібно враховувати, що збори організовуються різними спільнотами, а тому організація таких спільнот має визначатися у програмному забезпеченні, що розробляється.

Таким чином, метою роботи є розробити програмне забезпечення пошуку і організації зборів для сприяння живій або дистанційній комунікації

у спільнотах.

Завдання, винесені у підсумку для виконання в роботі:

- проаналізувати проблему розробки програми пошуку і організації зборів;
- спроектувати програмне забезпечення для пошуку і організації зборів;
- реалізувати програмне забезпечення для пошуку і організації зборів та виконати його тестування.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Огляд програм пошуку і організації зборів

Найбільш використовуваним програмним забезпеченням для надання інформації про збори та пошуку таких зборів є Meetup (рис. 1.1) [1].

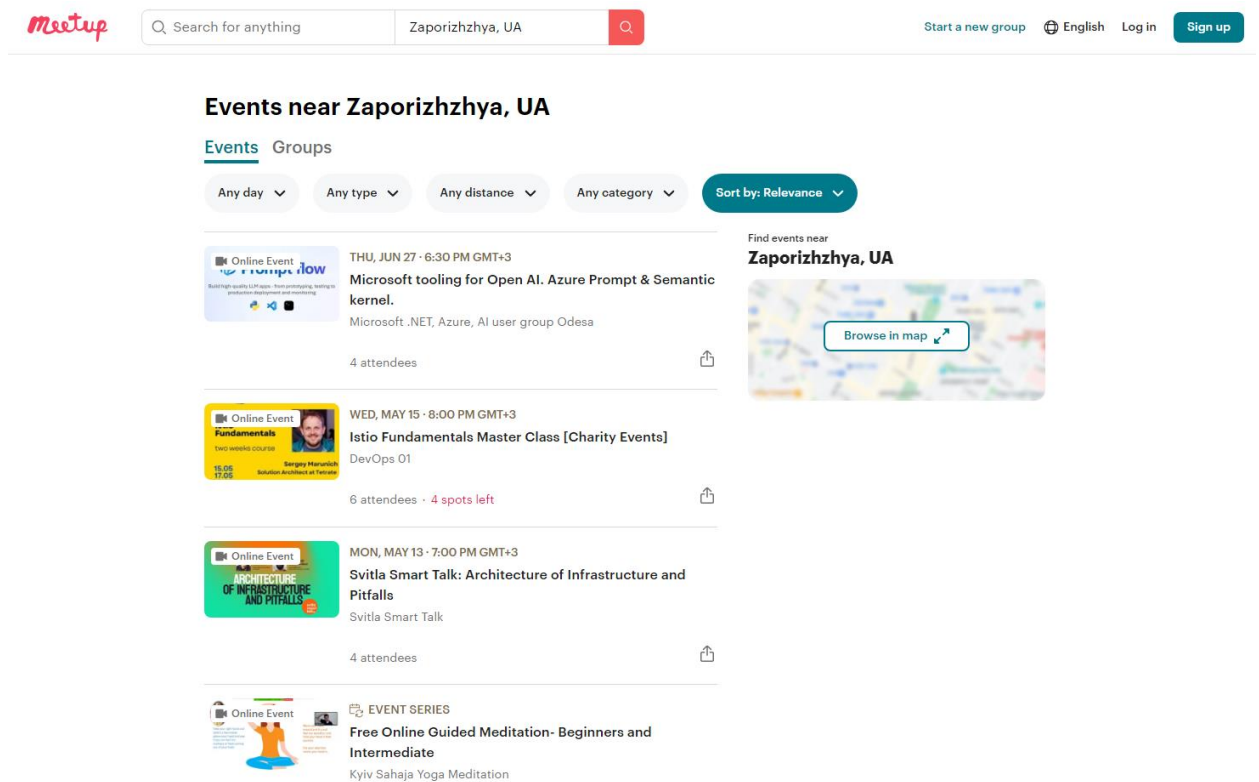


Рисунок 1.1 – Пошук зборів у вебдодатку Meetup

Meetup має багато різних функціональних інструментів для роботи зі зборами, проте стосовно створення власних зборів людей цей додаток вимагає сплати за роботу. Така підписка вимагає приблизно 25 американських доларів на місяць.

Основний пошук зборів відбувається, задаючи назву події та місце проведення. Місце проведення задається обов'язково. Після виконання авторизації збори можна не тільки шукати, але і переглядати збори, на які користувач зареєструвався, фільтруючи їх відповідним чином. Наприклад, за

датами календаря, за поточним тижнем, наступним тижнем, найближчими вихідними.

Збори можна зберегти одразу з переліку, але також і зі сторінки, де вони детально описуються (рис. 1.2).

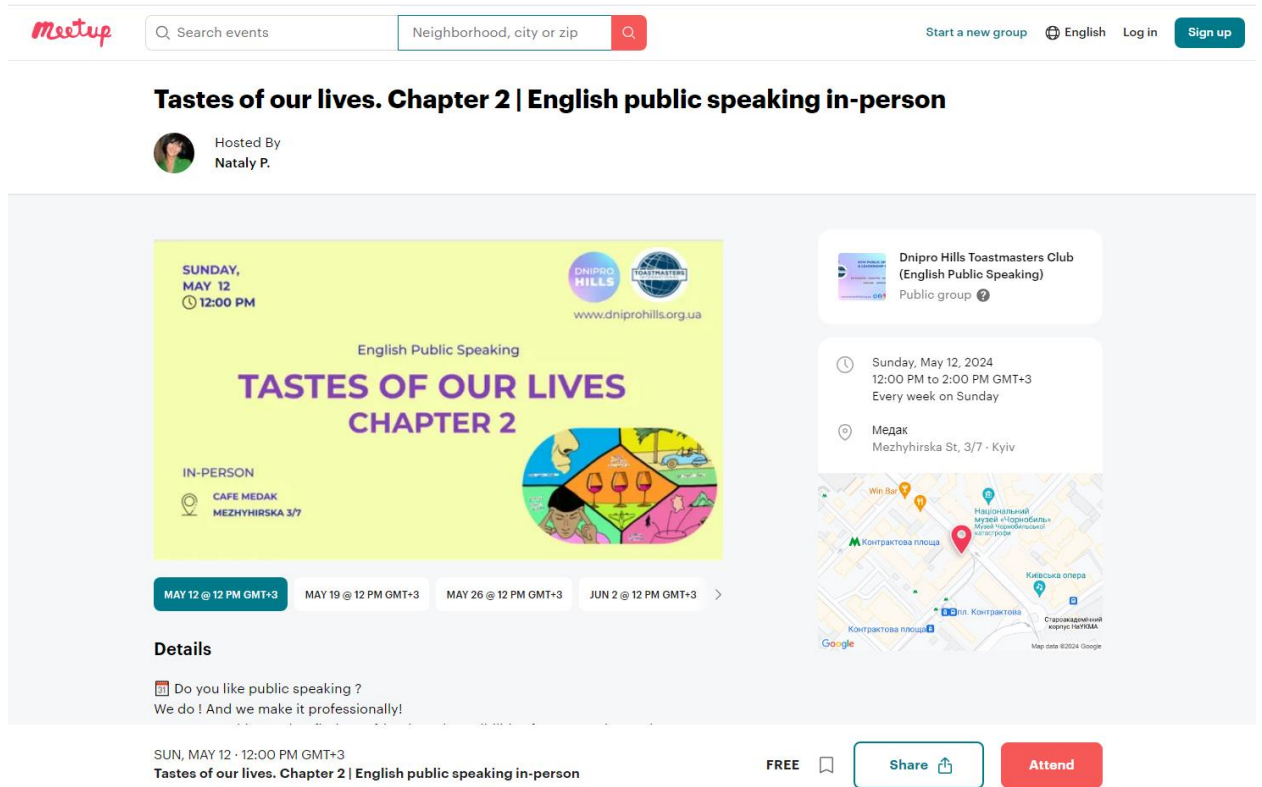


Рисунок 1.2 – Перегляд зборів у вебдодатку Meetup

На такій сторінці не тільки детально описуються збори, але і приводяться відвідувачі. Якщо не потрібна додаткова реєстрація (з чергою очікування), то приводиться єдиний перелік відвідувачів. Біля відвідувача також вказується окрема позначка, якщо людина є не тільки відвідувачем, але і організатором.

Після авторизації при переході до власних зборів можна обрати, чи це збори, які користувач має відвідати, чи це ті, які він сам проводить, чи це ті, які він зберіг, чи це ті, які вже раніше відбулись.

З меню можна з'єднатися з іншими людьми, з якими раніше були відвідані разом певні збори (рис. 1.3). Проте з'єднання можна встановити тільки протягом 24 годин після завершення зборів.

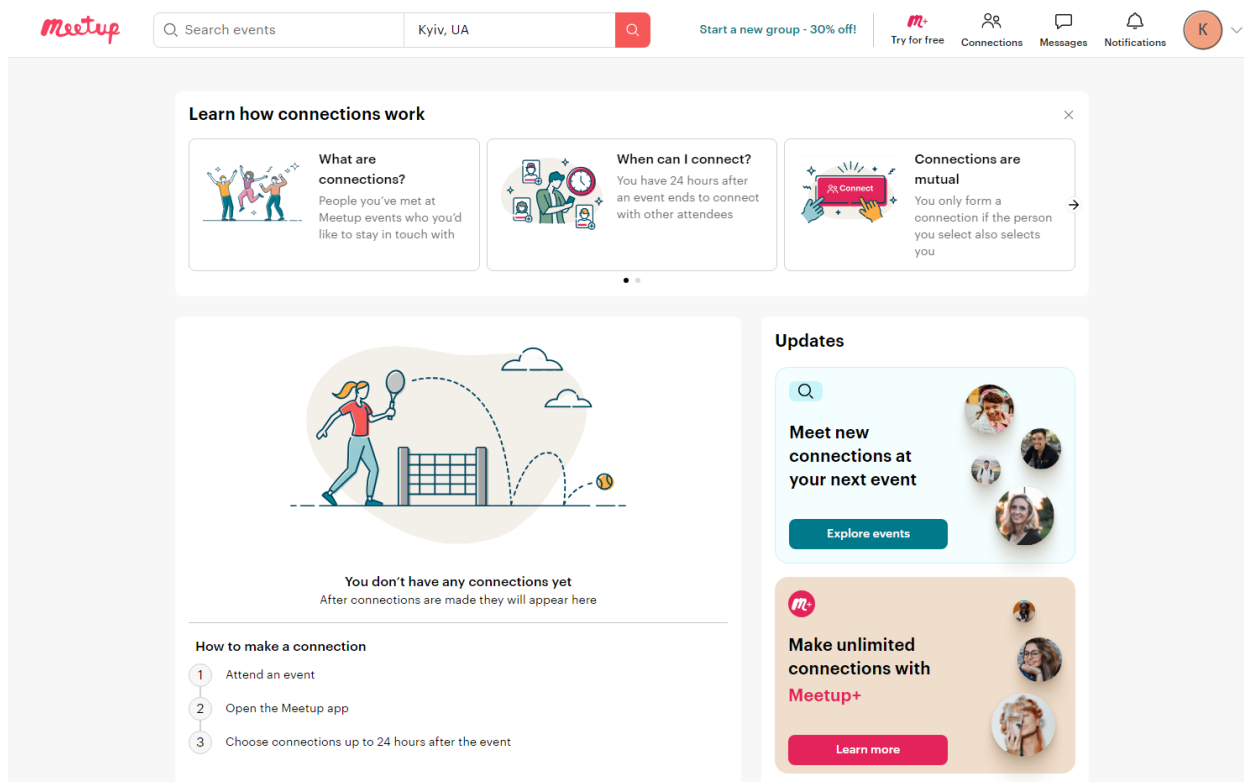


Рисунок 1.3 – Встановлення зв'язків після зборів у вебдодатку Meetup

Користувачі можуть додаватися до груп (рис. 1.4). Сторінка групи містить опис, перелік зборів, які за цією групою реалізуються, перелік учасників, спілкування фактично у вигляді коментарів. До груп можна додаватися, а за потреби залишати.

Групи фактично дозволяють об'єднуватися за інтересами, створювати певні спільноти, щоби таким чином швидше і результативніше розповсюджувати інформацію про подібні, потенційно цікаві збори в подальшому.

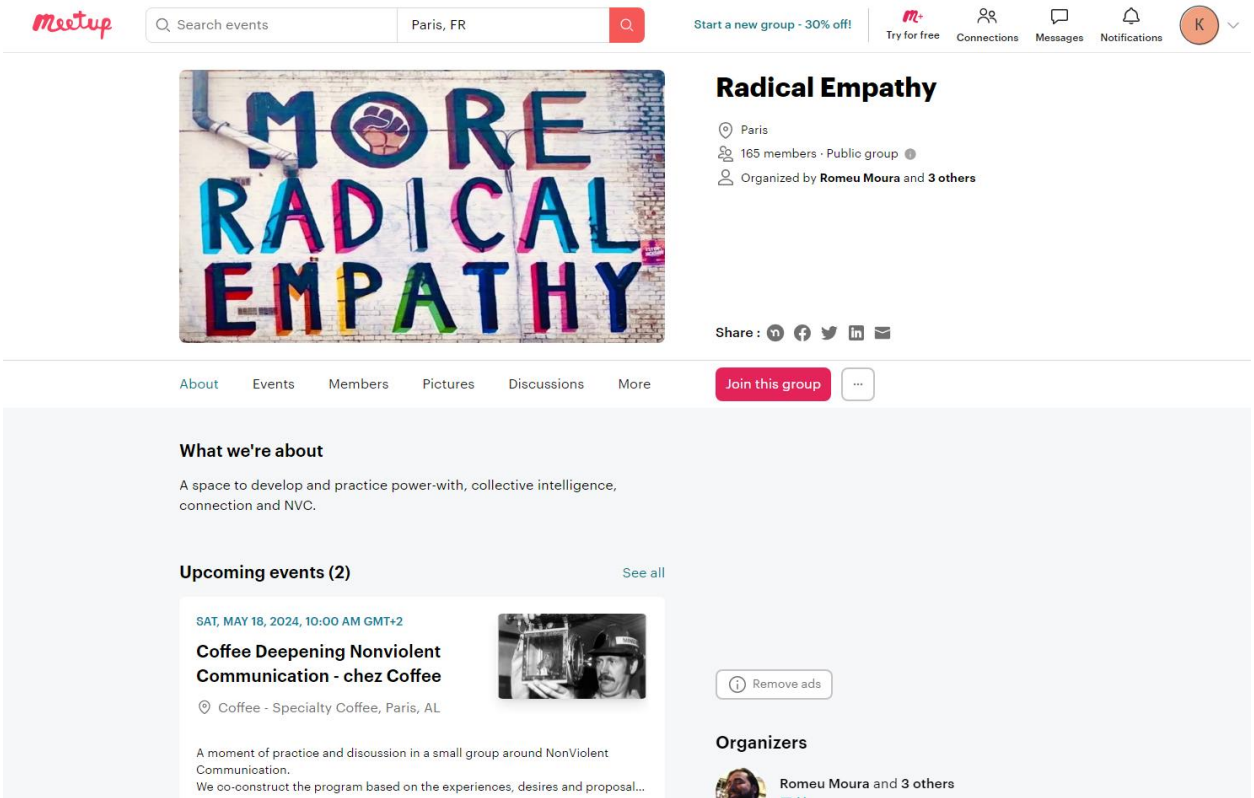


Рисунок 1.4 – Перегляд групи у вебдодатку Meetup

Альтернативним додатком є Community (рис. 1.5) [2]. Пошук зборів відбувається за мовою проведення. Відповідно не можна розподілити збори за місцем проведення саме під час виконання пошуку. Збори можна відфільтрувати або ті, які проводяться онлайн, або ті, які проводяться в конкретному місті, яке визначається автоматично за локацією.

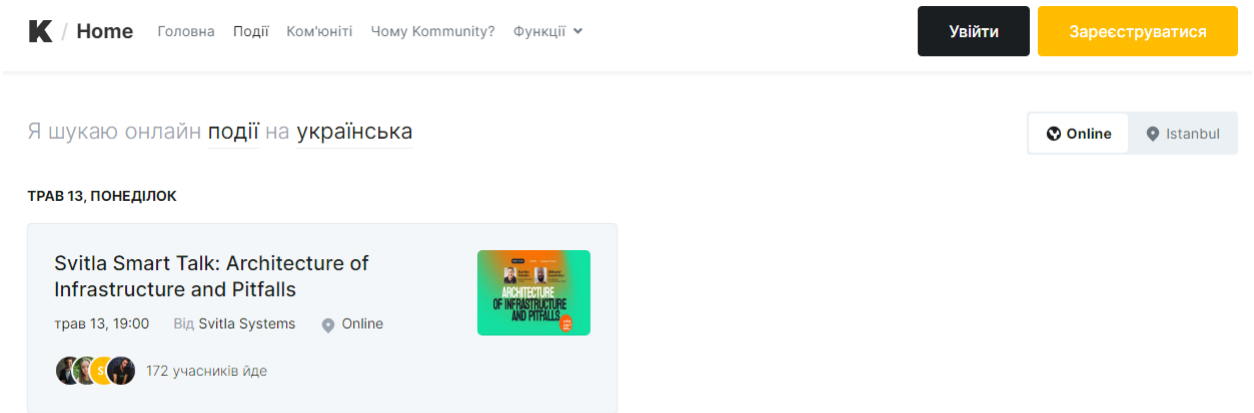


Рисунок 1.5 – Пошук зборів у вебдодатку Community

На сторінці зборів надається опис, інформація про учасників зборів, організатора, час та місце проведення, надається можливість переглянути створені раніше коментарі та одразу створити власний (рис. 1.6).

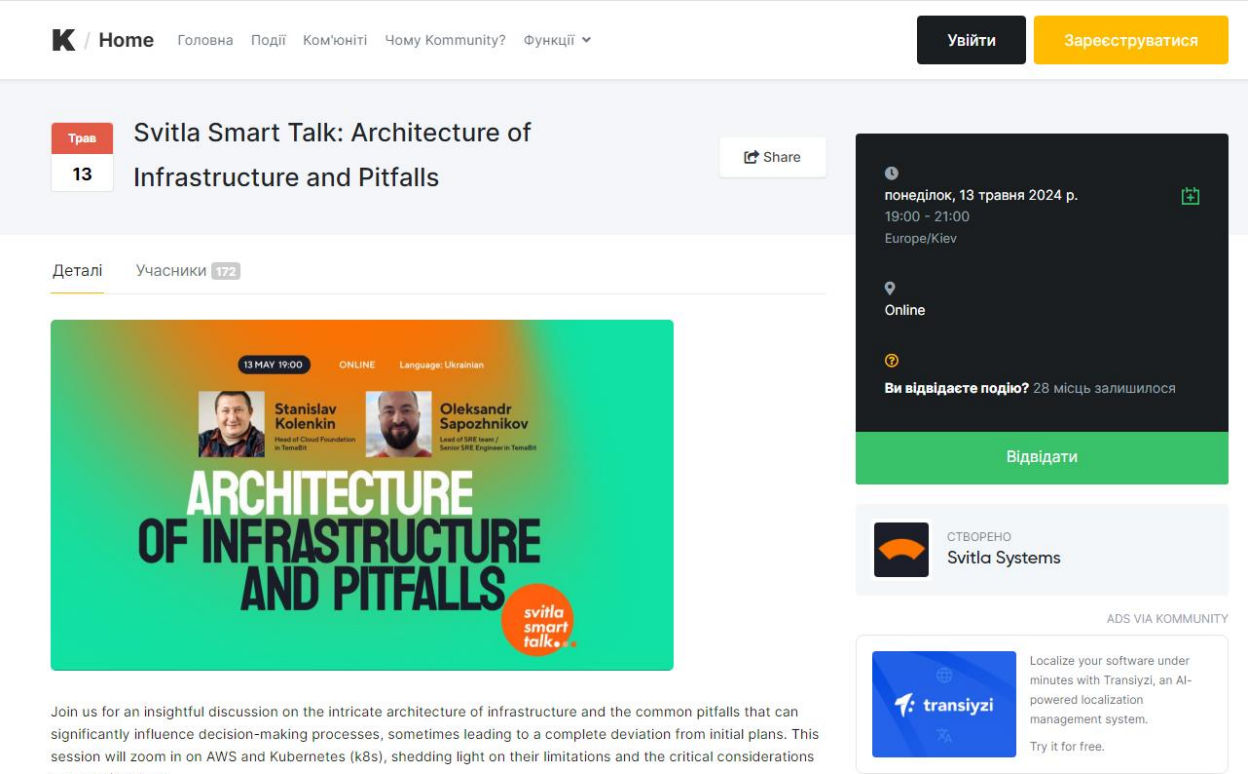


Рисунок 1.6 – Перегляд зборів у вебдодатку Kommunity

У додатку також присутні спільноти, які вже розподіляються за містами, а виконати пошук спільнот можна, вказуючи конкретне місто. При цьому пошук зборів за датами неможливий, а для спільнот недоцільний. Тобто за великої кількості зборів достатньо складно буде знаходити потрібні.

На сторінці спільноти (рис. 1.7) надається її опис, можна приєднатися до спільноти, наявна стіна з повідомлень, а також перелік зборів (розподілені тільки на заплановані та минулі) та учасників спільноти.

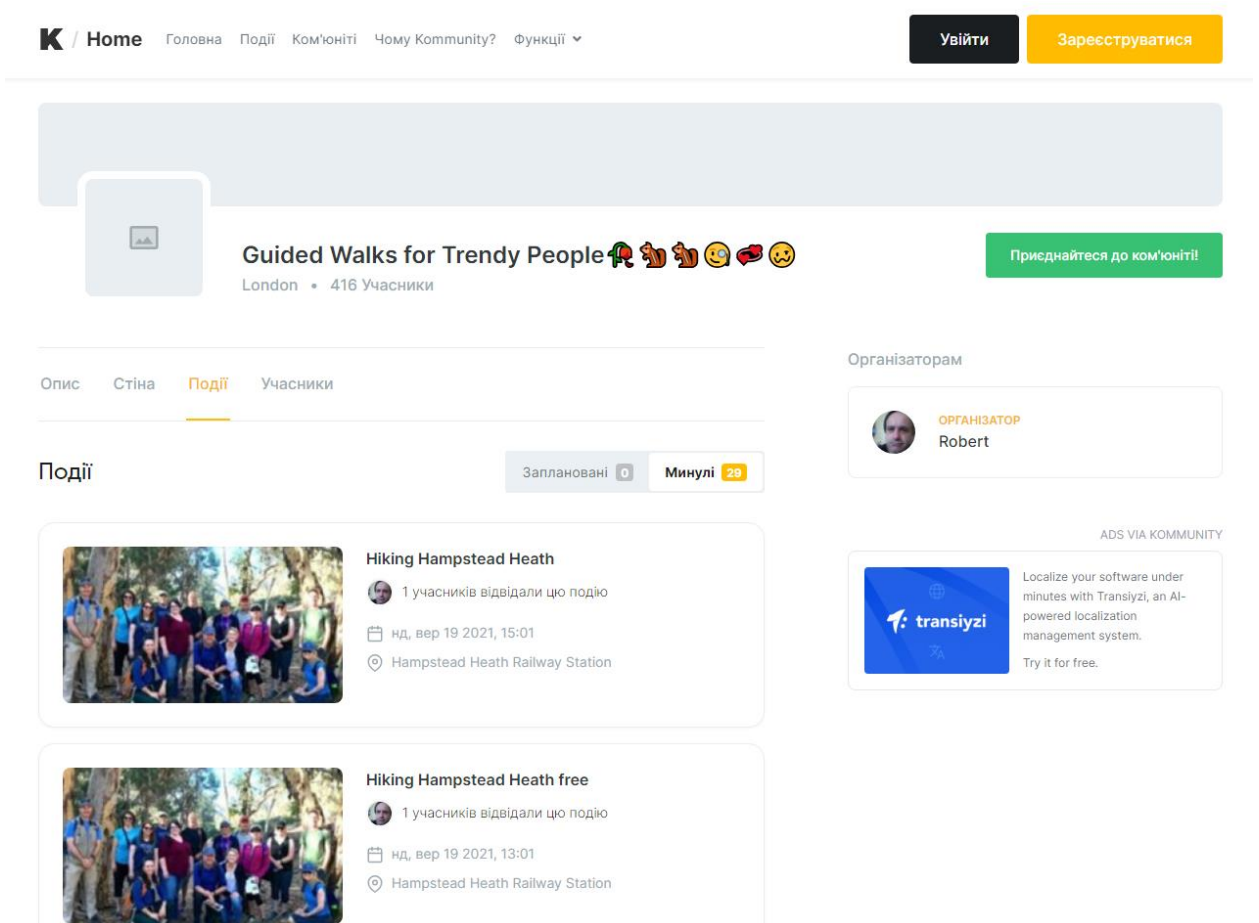


Рисунок 1.7 – Перегляд спільнот у вебдодатку Kommunity

Kommunity надає можливості роботи за меншу вартість на місяць, але створення тільки першого збору є безкоштовним. Тобто цей додаток також є платним.

## 1.2 Порівняння аналогів для визначення характеристик власної розробки

Порівнюючи Meetup, Kommunity з розробкою, що виконується в роботі, слід визначити, що функції планування зборів, створення спільнот, визначення організаторів спільнот, створення серії зборів, створення черги очікування є важливими і потрібними для реалізації, щоб задовольнити базові вимоги користувачів (табл. 1.1).

Таблиця 1.1 – Порівняння існуючих аналогів та розроблюваної програми для пошуку і організації зборів

| Характеристика                    | Meetup | Kommunity | Розробка |
|-----------------------------------|--------|-----------|----------|
| Планування зборів                 | Так    | Так       | Так      |
| Створення спільнот                | Так    | Так       | Так      |
| Визначення організаторів спільнот | Так    | Так       | Так      |
| Створення серій зборів            | Так    | Так       | Так      |
| Створення черги очікування        | Так    | Так       | Так      |
| Оцінювання зборів у серії         | Так    | Так       | Так      |
| Створення контактів після зборів  | Так    | Ні        | Так      |
| Плата за використання             | Так    | Так       | Ні       |

При цьому ці вимоги не є єдиними. Аналоги не мають можливостей оцінювання зборів, що є важливим для серії зборів. Середня оцінка за попередні збори визначає сприйняття користувачами наступних зборів з серії. Тому дана функція є важливою для реалізації.

Окрім того важливим є те, щоб розробка була безкоштовною, адже аналоги вимагають щомісячної сплати за роботу над створенням власних зборів.

Також важливо реалізувати особливість Meetup порівняно з Kommunity. Ця особливість пов'язана зі створенням контактів після зборів. Це приваблює користувачів повертатися до роботи з програмою після проведення збору. Тому така особливість має бути реалізована і в розробці.

### 1.3 Висновки за розділом 1

У розділі розглянуто програми Meetup, Kommunity, що можуть використовуватися для пошуку і організації зборів. Визначено ті

функціональні можливості, які можуть використовуватися в даній сфері. Основні такі можливості виділені для реалізації в розробці, а також визначено її особливості.

## 2 ПРОЄКТУВАННЯ ПРОГРАМИ

### 2.1 Вибір засобів проєктування і розробки

Мова Python [3] є потужною мовою сучасної розробки, яка втім як один з варіантів застосування визначає веброзробку. Для ефективної веброзробки потрібен відповідний фреймворк. Фреймворки бувають фактично двох груп. Як приклад кожної з цих груп виділено Django [4] і Falcon [5]-[6] (табл. 2.1). Falcon має дуже обмежений набір засобів роботи за замовчуванням, що сприяє швидкості роботи, однак вимагає більше часу на інтеграцію цих засобів. Django має достатньо якісні засоби одразу після встановлення і створення додатку.

Обрано було фреймворк Django з мовою Python. Оскільки фреймворк має певні правила створення додатків щонайменше за архітектурою і відповідною структурою, то це також визначило побудову архітектури програми в подальшому на основі шаблону проєктування Model-template-view (MTV) [7].

Таблиця 2.1 – Порівняння фреймворків, необхідних для створення вебдодатку

| Характеристика                        | Falcon                                      | Django                     |
|---------------------------------------|---------------------------------------------|----------------------------|
| Створення бази даних з коду           | Не має базових засобів, потребує інтеграції | Присутнє за замовчуванням  |
| Керування даними з базової панелі     | Не має базових засобів, потребує інтеграції | Присутнє за замовчуванням  |
| Робота з формами                      | Присутня за замовчуванням                   | Присутня за замовчуванням  |
| Засоби авторизації                    | Не має базових засобів, потребує інтеграції | Присутні за замовчуванням  |
| Рушій для роботи з шаблонами сторінок | Не має базових засобів, потребує інтеграції | Присутній за замовчуванням |

## 2.2 Моделювання взаємодії користувача з програмою

Процес моделювання взаємодії користувача з програмою за допомогою діаграм прецедентів було розпочато з визначення акторів. Було вирішено виділити тільки одного загального актора – користувача, який одночасно може бути і відвідувачем зборів або учасником спільнот, так і власником, організатором спільноти, зборів. Усе це відрізняється в залежності від того, з якими саме зборами або спільнотами відбувається робота. Тому для користувача основним потоком сценаріїв для роботи є наступний:

- створення спільноти відвідувачів зборів, альтернативним сценарієм для якого є редагування інформації про власну спільноту відвідувачів зборів, коли спільнота створена, проте потрібно змінити інформацію про неї;

- додавання учасника до спільноти, альтернативним сценарієм для якого є вилучення або блокування учасника в спільноті, якщо користувач вже є учасником спільноти відвідувачів зборів;

- демонстрування переліку спільнот, учасниками яких є користувач;

- демонстрування інформації про спільноту відвідувачів зборів;

- демонстрування переліку зборів, які організовані спільнотою;

- демонстрування переліку серій зборів, які організовані спільнотою;

- додавання організатора власної спільноти відвідувачів зборів, альтернативним сценарієм для якого є скасування організатора спільноти відвідувачів зборів, якщо користувач вже є організатором спільноти відвідувачів зборів;

- створення серії зборів власної спільноти, альтернативним сценарієм для якого є редагування серії зборів, якщо серія вже створена, але потрібно змінити інформацію про неї;

- створення запису про заплановані збори у власній спільноті, альтернативним сценарієм для якого є редагування інформації про заплановані збори, скасування проведення зборів;

- демонстрування інформації про збори;

- демонстрування учасників зборів;
- реєстрація участі в зборах, альтернативним сценарієм для якого є скасування участі в зборах, якщо користувач вже додався до участі в зборах;
- додавання до переліку очікування зборів, альтернативним сценарієм для якого є вилучення з переліку очікування зборів, якщо користувач вже раніше додався до переліку очікування зборів;
- підтвердження участі в зборах власної спільноти з переліку очікування, альтернативним сценарієм для якого є відхилення участі в зборах власної спільноти з переліку очікування, якщо власник спільноти (користувач, який її створив, або який є одним з організаторів) не надає дозвіл на участь;
- пошук зборів;
- демонстрування переліку зборів з серії;
- демонстрування переліку зборів, на які зареєструвався користувач;
- демонстрування переліку зборів, що організовуються спільнотами, в яких є учасником користувач;
- демонстрування переліку створених користувачем зборів;
- демонстрування інформації про користувача, альтернативним сценарієм для якого є редагування даних користувацького запису;
- збереження зборів, альтернативним сценарієм для якого є скасування збереження зборів, якщо збори було збережено тим самим користувачем раніше;
- демонстрування переліку збережених зборів;
- створення запиту після спільного відвідування зборів, альтернативним сценарієм для якого є скасування запиту, якщо користувач, якому він адресований, ще не підтвердив або не відхилив його, а користувач, який створив запит, не має в ньому більше потреби;
- підтвердження адресованого користувачу запиту після спільного відвідування зборів, альтернативним сценарієм для якого є відхилення адресованого користувачу запиту після спільного відвідування зборів (рис. 2.1).

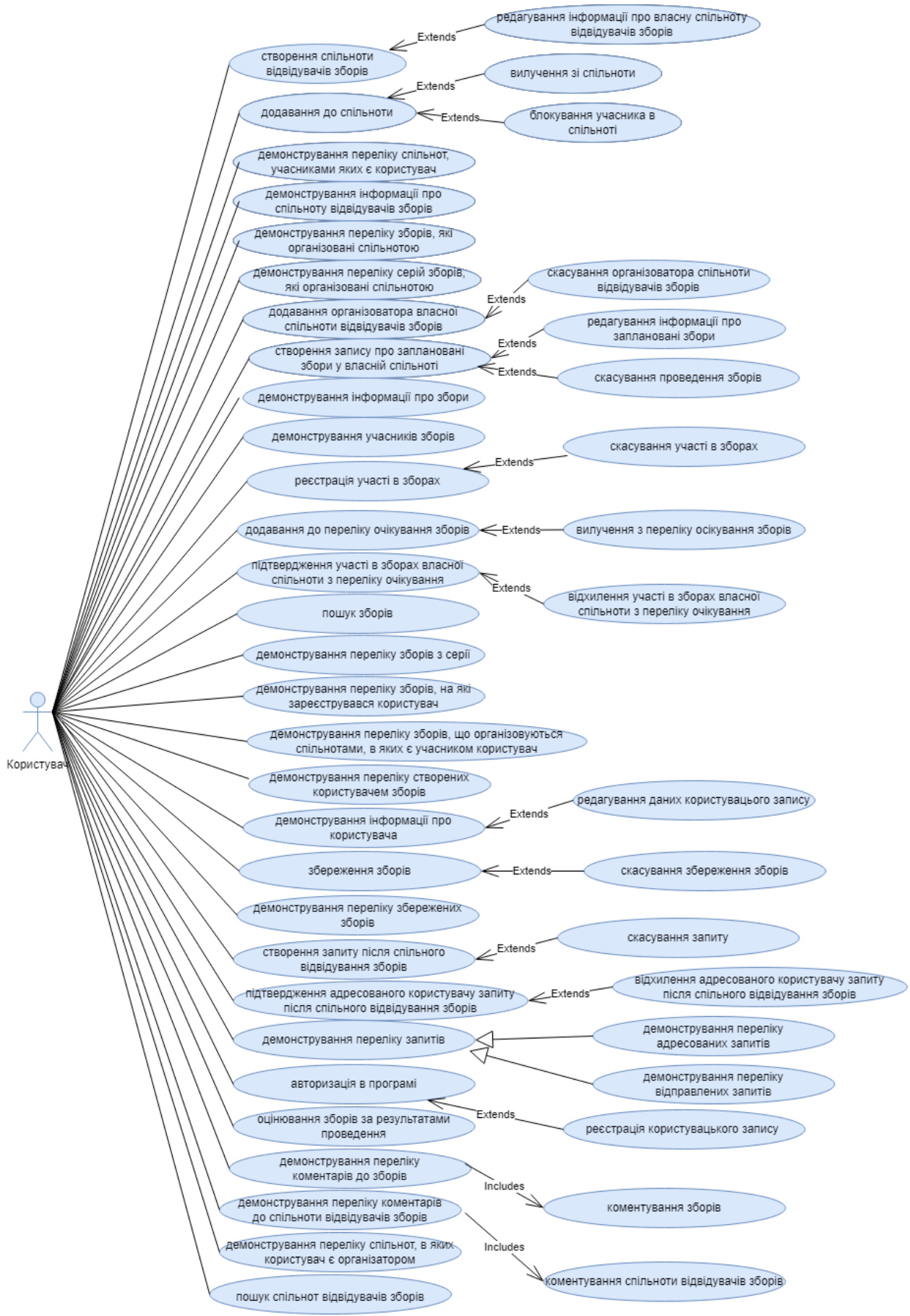


Рисунок 2.1 – Діаграма прецедентів програмного забезпечення для пошуку і організації зборів

До основного потоку належать ще сценарії:

- демонстрування переліку запитів, що узагальнює сценарії демонстрування переліку адресованих або відправлених запитів;
- авторизація в програмі, альтернативним сценарієм для якого є реєстрація користувачького запису для ситуації відсутності такого запису;
- оцінювання зборів за результатами проведення;
- демонстрування переліку коментарів до зборів, що включає сценарій коментування зборів;
- демонстрування переліку коментарів до спільноти відвідувачів зборів, що включає сценарій коментування спільноти відвідувачів зборів;
- демонстрування переліку спільнот, в яких користувач є організатором;
- пошук спільнот відвідувачів зборів.

## 2.3 Проєктування структури бази даних

Однією з передових на даний момент реляційних баз даних (БД), яка окрім того є й безкоштовною, є PostgreSQL [8]-[9], яку зважаючи на все це, було обрано для проєктування і забезпечення роботи з БД.

Проєктування було завершено виділенням потрібних для реалізації підтримки основних сутностей таблиць, що описані далі.

Таблиця `gathapp_customeruser` з відповідними полями представляє користувачький запис у програмі:

- `id` – номер, що унікально ідентифікує користувачький запис;
- `username` – логін, що текстово унікально ідентифікує користувачький запис;
- `email` – електронна пошта, що використовується власником цього користувачького запису;
- `firstname` – ім'я власника цього користувачького запису;
- `lastname` – прізвище власника цього користувачького запису;

- password – пароль власника цього користувацького запису;
- active – поле, що визначає, чи досі є активним цей користувацький запис, чи він більше не має використовуватись.

Самі користувацькі записи використовуються, щоб ідентифікувати користувачів програми, а далі вже вони або створюють свої збори, або долучаються до створених іншими. Проте збори недоцільно створювати всі як окремі. Їх доцільно об'єднати в певні спільноти. Це обумовлено тим, що збори людей нерідко не є відокремленими подіями, адже мають стандартних організаторів, а розглядати їх відокремлено некоректно. Відокремлено такі події важче шукати. З іншого боку, якщо люди об'єднані в спільноти, то далі вони можуть чітко виділяти ті збори, які для них можуть бути потенційно цікавими, адже проводяться такою спільнотою.

Таблиця `gathapp_community` з відповідними полями представляє спільноти:

- `id` – номер, що унікально ідентифікує цю спільноту відвідувачів зборів;
- `fullcaption` – повна назва спільноти відвідувачів зборів;
- `gathercityfk` – місто, де функціонує ця спільнота відвідувачів зборів (визначається як зовнішній ключ з таблиці міст);
- `information` – деталізований опис спільноти відвідувачів зборів, щоб потенційні учасники могли визначити, чи цікавою є для них потенційно ця спільнота;
- `postphoto` – фотографія, яка як постер використовується для ілюстрації спільноти відвідувачів зборів;
- `creatorfk` – користувацький запис, за допомогою якого було створено цю спільноту відвідувачів зборів з самого початку (визначається як зовнішній ключ з таблиці користувацьких записів);
- `visible` – поле, що визначає чи це стандартна спільнота, чи прихована: приховані спільноти потрібні зокрема тоді, коли користувач створює збори людей від себе самого, не потребує виділення спільноти, але для використання

стандартної структури роботи, що визначена цією БД, має бути виділено спільноту, а тоді вже зі спільнотою співвіднесено збори, тож спільнота має бути за замовчуванням, а тому за відсутності такої для кожного користувача, який запланував збори людей, виділяється спільнота за замовчуванням, яка є стандартною, не потрібна іншим людям, а тому визначається як прихована.

Таблиця `gathapp_communityorganizer` з відповідними полями представляє організаторів спільнот відвідувачів зборів:

- `id` – номер, що унікально ідентифікує організатора спільноти відвідувачів зборів;
- `userfk` – користувацький запис, за яким працює організатор цієї спільноти відвідувачів зборів (визначається як зовнішній ключ з таблиці користувацьких записів);
- `communityfk` – спільнота відвідувачів зборів, для якої є організатором цей користувач (визначається як зовнішній ключ з таблиці спільнот відвідувачів зборів).

Першим організатором є користувач, який створив спільноту, але вже далі він, а згодом і інші організатори можуть додавати інших користувачів як організаторів.

Основною функцією таблиці `gathapp_communityorganizer` є забезпечення зв'язку багато-до-багатьох між таблицями спільнот відвідувачів зборів та користувацьких записів.

Таблиця `gathapp_communityvisitor` з відповідними полями представляє користувачів, які є учасниками спільноти відвідувачів зборів:

- `id` – номер, що унікально ідентифікує учасника спільноти відвідувачів зборів;
- `userfk` – користувацький запис, за яким працює учасник цієї спільноти відвідувачів зборів (визначається як зовнішній ключ з таблиці користувацьких записів);

– communityfk – спільнота відвідувачів зборів, в якій приймає участь цей користувач (визначається як зовнішній ключ з таблиці спільнот відвідувачів зборів);

– block – поле, що визначає блокування доступу користувача до спільноти відвідувачів зборів, якщо таке було здійснено, або відповідно вільний доступ.

Таблиця gathapp\_gathercountry з відповідними полями представляє країни:

- id – номер, що унікально ідентифікує країну;
- fullcaption – назва країни.

Таблиця gathapp\_gathercity з відповідними полями представляє міста:

- id – номер, що унікально ідентифікує місто;
- countryfk – країна, де є це місто (визначається як зовнішній ключ з таблиці країн);
- fullcaption – назва міста.

Таблиця gathapp\_gatheringseries з відповідними полями представляє серії зборів:

- id – номер, що унікально ідентифікує серію зборів;
- communityfk – спільнота відвідувачів зборів, у межах якої організовується ця серія зборів (визначається як зовнішній ключ з таблиці спільнот відвідувачів зборів);
- fullcaption – повна назва серії зборів;
- avscore – середня оцінка, яку отримали збори, що відбувались як частина цієї серії;
- visible – поле, що визначає чи це стандартна серія зборів, чи прихована: приховані серії зборів для зручності використовуються організаторами, але не відображаються користувачам.

Таблиця gathapp\_gathering з відповідними полями представляє самі збори людей:

- id – номер, що унікально ідентифікує ці збори;

- gatheringseriesfk – серія зборів, в межах якої реалізуються ці окремі збори (визначається як зовнішній ключ з таблиці серій зборів);
- fullcaption – повна назва зборів;
- postphoto – фотографія, яка як постер використовується для ілюстрації цих зборів людей;
- gathercityfk – місто, де відбуваються ці збори людей (визначається як зовнішній ключ з таблиці міст);
- gatheraddress – адреса, за якою в заданому місті відбуваються збори людей;
- link – посилання на конференцію для дистанційних зборів;
- ticket – необхідність відвідування зборів тільки за придбання квитків (позитивне значення встановлює вартість квитка, нульове – зборів не передбачають придбання квитків);
- control – контроль відвідувачів зборів зі сторони організаторів: якщо контроль здійснюється, то обмежується максимальна кількість відвідувачів, а організатор має підтвердити або скасувати участь відвідувача після того, як він додався до черги очікування;
- maxplaces – максимальна кількість людей що можуть зібратися на цих зборах;
- creatorfk – користувацький запис, за допомогою якого було створено ці збори (визначається як зовнішній ключ з таблиці користувацьких записів);
- starttime – дата і час, коли мають розпочатися збори;
- longtime – запланована тривалість зборів.

Таблиця gathapp\_gatheringvisitor з відповідними полями представляє відвідувачів зборів:

- id – номер, що унікально ідентифікує участь відвідувача у зборах;
- gatheringfk – збори, в яких приймає участь відвідувач (визначається як зовнішній ключ з таблиці зборів);
- userfk – користувацький запис, за яким працює цей відвідувач зборів (визначається як зовнішній ключ з таблиці користувацьких записів);

– variant – поточний варіант участі (zareєстрований, в очікуванні або скасований).

Таблиця gathapp\_gatheringsave з відповідними полями представляє збереження зборів:

- id – номер, що унікально ідентифікує збереження зборів;
- gatheringfk – збори, які збережено (визначається як зовнішній ключ з таблиці зборів);
- userfk – користувацький запис, який зберіг збори (визначається як зовнішній ключ з таблиці користувацьких записів).

Таблиця gathapp\_visitorrequest з відповідними полями представляє запити між користувацькими записами після відвідування зборів:

- id – номер, що унікально ідентифікує запит між користувацькими записами після відвідування зборів;
- userfk – користувацький запис, який створив запит між користувацькими записами після відвідування зборів (визначається як зовнішній ключ з таблиці користувацьких записів);
- usertext – повідомлення, яке в запиті було направлено автором, що створив запит;
- usertofk – користувацький запис, якому адресовано запит між користувацькими записами після відвідування зборів (визначається як зовнішній ключ з таблиці користувацьких записів);
- usertotext – повідомлення, яке у відповіді на запит було направлено користувачем, який відповів на запит;
- variant – стан відповіді на запит (без відповіді, підтверджено, скасовано).

Таблиця gathapp\_gathercomment з відповідними полями представляє коментар до зборів:

- id – номер, що унікально ідентифікує коментар до зборів;
- userfk – користувацький запис, який створив коментар (визначається як зовнішній ключ з таблиці користувацьких записів);



- userfk – користувацький запис, який виконав оцінювання (визначається як зовнішній ключ з таблиці користувацьких записів);
- gatheringfk – збори, які оцінено (визначається як зовнішній ключ з таблиці зборів);
- mark – виставлена оцінка зборам;
- scoretime – час оцінювання зборів.

Таблиця gathapp\_communitycomment з відповідними полями представляє коментар до спільноти відвідувачів зборів:

- id – номер, що унікально ідентифікує коментар до спільноти відвідувачів зборів;
- userfk – користувацький запис, який створив коментар (визначається як зовнішній ключ з таблиці користувацьких записів);
- communityfk – спільнота відвідувачів зборів, до якої адресовано цей коментар (визначається як зовнішній ключ з таблиці спільноти відвідувачів зборів);
- commtime – час створення коментаря до спільноти відвідувачів зборів;
- commtext – текст коментаря до спільноти відвідувачів зборів.

## 2.4 Висновки за розділом 2

Спільне використання мови Python та фреймворку Django сприяє швидкій розробці в межах дипломної кваліфікаційної роботи з виконанням розробки в дуже обмежені строки, сприяє забезпеченню функціонування потужного додатку. З використанням системи PostgreSQL виконано проєктування і забезпечення роботи з БД. Прецеденти відповідних діаграм визначені для моделювання програмного забезпечення.

### 3 РОЗРОБКА ПРОГРАМИ

#### 3.1 Визначення послідовності роботи користувача з програмою

Для визначення послідовності роботи користувача з програмою було створено схему функціонування програми (рис. 3.1), що розпочинається з відображення вхідної сторінки, а продовжується вибором відповідних розділів, підрозділів меню для виконання наступних дій на основі функціональних можливостей, перелічених у розділі 2.

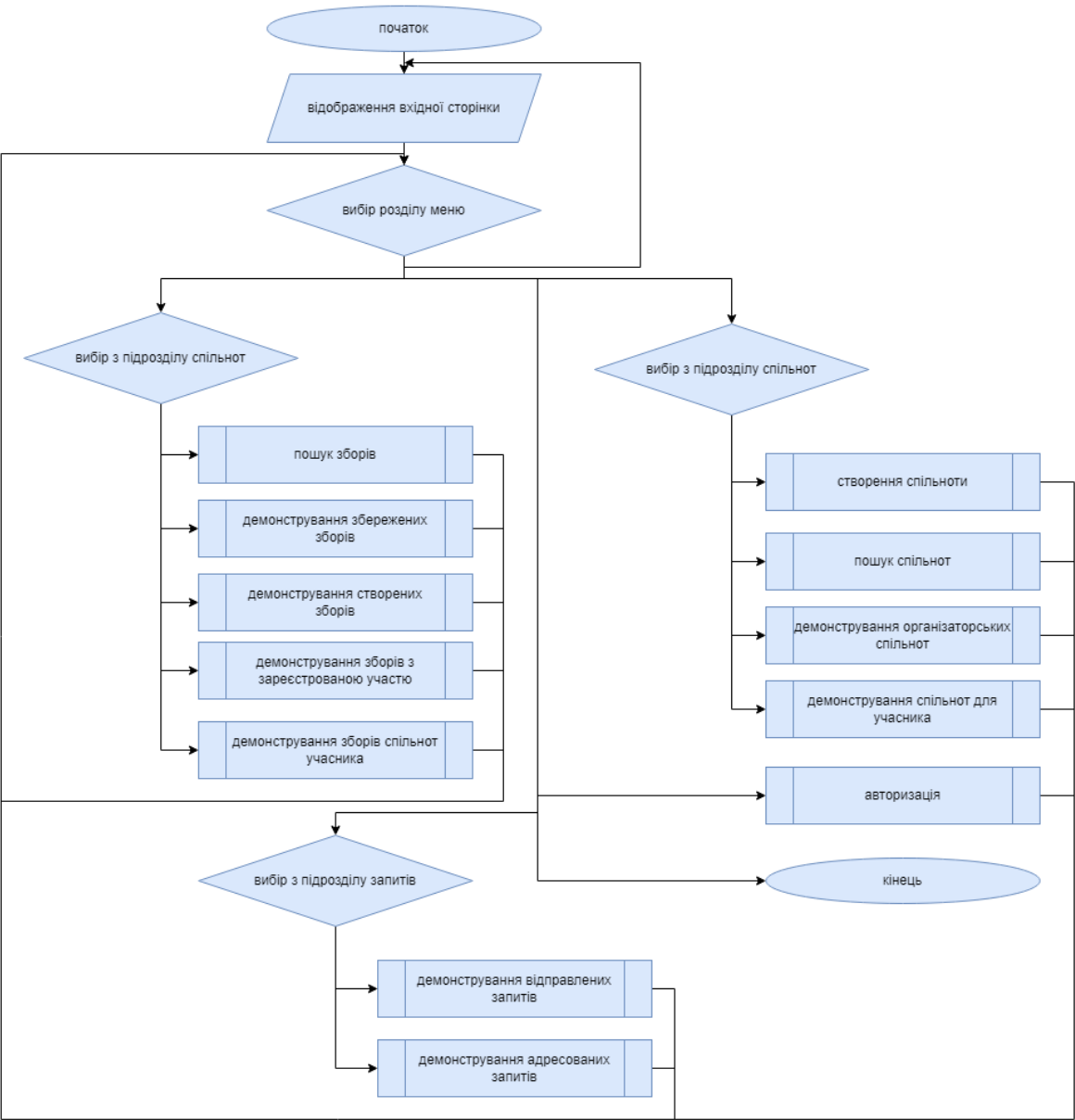


Рисунок 3.1 – Схема функціонування програмного забезпечення для пошуку і організації зборів

### 3.2 Особливості програмної реалізації

Програмна реалізація виконана зі створенням структури додатку за шаблоном MTV.

Програмно реалізовані описані нижче моделі.

Модель Customeruser реалізує в складі програмного додатку користувацький запис у програмі зі структурою:

- поле id програмно реалізоване з типом AutoField, представляє номер, що унікально ідентифікує користувацький запис;
- поле username програмно реалізоване з типом CharField, представляє логін, що текстово унікально ідентифікує користувацький запис;
- поле email програмно реалізоване з типом EmailField, представляє електронну пошту, що використовується власником цього користувацького запису;
- поле firstname програмно реалізоване з типом CharField, представляє ім'я власника цього користувацького запису;
- поле lastname програмно реалізоване з типом CharField, представляє прізвище власника цього користувацького запису;
- поле password програмно реалізоване з типом CharField, представляє пароль власника цього користувацького запису;
- поле active програмно реалізоване з типом BooleanField, представляє поле, що визначає, чи досі є активним цей користувацький запис, чи він більше не має використовуватись.

Модель Community реалізує в складі програмного додатку спільноти зі структурою:

- поле id програмно реалізоване з типом AutoField, представляє номер, що унікально ідентифікує цю спільноту відвідувачів зборів;
- поле fullcaption програмно реалізоване з типом CharField, представляє повну назву спільноти відвідувачів зборів;

– поле `gathercityfk` програмно реалізоване з типом `ForeignKey`, представляє місто, де функціонує ця спільнота відвідувачів зборів (визначається як зовнішній ключ з моделі міст `Gathercity`);

– поле `information` програмно реалізоване з типом `TextField`, представляє деталізований опис спільноти відвідувачів зборів, щоб потенційні учасники могли визначити, чи цікавою є для них потенційно ця спільнота;

– поле `postphoto` програмно реалізоване з типом `ImageField`, представляє фотографію, яка як постер використовується для ілюстрації спільноти відвідувачів зборів;

– поле `creatorfk` програмно реалізоване з типом `ForeignKey`, представляє користувацький запис, за допомогою якого було створено цю спільноту відвідувачів зборів з самого початку (визначається як зовнішній ключ з моделі користувацьких записів `Customeruser`);

– поле `visible` програмно реалізоване з типом `BooleanField`, представляє поле, що визначає чи це стандартна спільнота, чи прихована;

– поле `organizers` програмно реалізоване з типом `ManyToManyField`, представляє організаторів спільнот відвідувачів зборів через зв'язок багато-до-багатьох.

Модель `Communityvisitor` реалізує в складі програмного додатку користувачів, які є учасниками спільноти відвідувачів зборів, зі структурою:

– поле `id` програмно реалізоване з типом `AutoField`, представляє номер, що унікально ідентифікує учасника спільноти відвідувачів зборів;

– поле `userfk` програмно реалізоване з типом `ForeignKey`, представляє користувацький запис, за яким працює учасник цієї спільноти відвідувачів зборів (визначається як зовнішній ключ з моделі користувацьких записів `Customeruser`);

– поле `communityfk` програмно реалізоване з типом `ForeignKey`, представляє спільноту відвідувачів зборів, в якій приймає участь цей користувач (визначається як зовнішній ключ з моделі спільнот відвідувачів зборів `Community`);

– поле `block` програмно реалізоване з типом `BooleanField`, представляє поле, що визначає блокування доступу користувача до спільноти відвідувачів зборів, якщо таке було здійснено, або відповідно вільний доступ.

Модель `Gathercountry` реалізує в складі програмного додатку країни зі структурою:

– поле `id` програмно реалізоване з типом `AutoField`, представляє номер, що унікально ідентифікує країну;

– поле `fullcaption` програмно реалізоване з типом `CharField`, представляє назву країни.

Таблиця `Gathercity` реалізує в складі програмного додатку міста зі структурою:

– поле `id` програмно реалізоване з типом `AutoField`, представляє номер, що унікально ідентифікує місто;

– поле `countryfk` програмно реалізоване з типом `ForeignKey`, представляє країну, де є це місто (визначається як зовнішній ключ з моделі країн `Gathercountry`);

– поле `fullcaption` програмно реалізоване з типом `CharField`, представляє назву міста.

Модель `Gatheringseries` реалізує в складі програмного додатку серії зборів зі структурою:

– поле `id` програмно реалізоване з типом `AutoField`, представляє номер, що унікально ідентифікує серію зборів;

– поле `communityfk` програмно реалізоване з типом `ForeignKey`, представляє спільноту відвідувачів зборів, у межах якої організовується ця серія зборів (визначається як зовнішній ключ з моделі спільнот відвідувачів зборів `Community`);

– поле `fullcaption` програмно реалізоване з типом `CharField`, представляє повну назву серії зборів;

– поле `avscore` програмно реалізоване з типом `FloatField`, представляє середню оцінку, яку отримали збори, що відбувались як частина цієї серії;

– поле `visible` програмно реалізоване з типом `BooleanField`, представляє поле, що визначає чи це стандартна серія зборів, чи прихована.

Модель `Gathering` реалізує в складі програмного додатку збори людей зі структурою:

– поле `id` програмно реалізоване з типом `AutoField`, представляє номер, що унікально ідентифікує ці збори;

– поле `gatheringseriesfk` програмно реалізоване з типом `ForeignKey`, представляє серію зборів, в межах якої реалізуються ці окремі збори (визначається як зовнішній ключ з моделі серій зборів `Gatheringseries`);

– поле `fullcaption` програмно реалізоване з типом `CharField`, представляє повну назву зборів;

– поле `postphoto` програмно реалізоване з типом `ImageField`, представляє фотографію, яка як постер використовується для ілюстрації цих зборів людей;

– поле `gathercityfk` програмно реалізоване з типом `ForeignKey`, представляє місто, де відбуваються ці збори людей (визначається як зовнішній ключ з моделі міст `Gathercity`);

– поле `gatheraddress` програмно реалізоване з типом `CharField`, представляє адресу, за якою в заданому місті відбуваються збори людей;

– поле `link` програмно реалізоване з типом `URLField`, представляє посилання на конференцію для дистанційних зборів;

– поле `ticket` програмно реалізоване з типом `BooleanField`, представляє необхідність відвідування зборів тільки за придбання квитків (позитивне значення встановлює вартість квитка, нульове – зборів не передбачають придбання квитків);

– поле `control` програмно реалізоване з типом `BooleanField`, представляє контроль відвідувачів зборів зі сторони організаторів: якщо контроль здійснюється, то обмежується максимальна кількість відвідувачів, а організатор має підтвердити або скасувати участь відвідувача після того, як він додався до черги очікування;

- поле `maxplaces` програмно реалізоване з типом `IntegerField`, представляє максимальну кількість людей що можуть зібратися на цих зборах;
- поле `creatorfk` програмно реалізоване з типом `ForeignKey`, представляє користувацький запис, за допомогою якого було створено ці збори (визначається як зовнішній ключ з моделі користувацьких записів `Customeruser`);
- поле `starttime` програмно реалізоване з типом `DateTimeField`, представляє дату і час, коли мають розпочатися збори;
- поле `longtime` програмно реалізоване з типом `TimeField`, представляє заплановану тривалість зборів.

Модель `Gatheringvisitor` реалізує в складі програмного додатку відвідувачів зборів зі структурою:

- поле `id` програмно реалізоване з типом `AutoField`, представляє номер, що унікально ідентифікує участь відвідувача у зборах;
- поле `gatheringfk` програмно реалізоване з типом, представляє збори, в яких приймає участь відвідувач (визначається як зовнішній ключ з таблиці зборів);
- поле `userfk` програмно реалізоване з типом `ForeignKey`, представляє користувацький запис, за яким працює цей відвідувач зборів (визначається як зовнішній ключ з моделі користувацьких записів `Customeruser`);
- поле `variant` програмно реалізоване з типом `SmallIntegerField`, представляє поточний варіант участі (zareєстрований, в очікуванні, скасований).

Модель `Gatheringsave` реалізує в складі програмного додатку збереження зборів зі структурою:

- поле `id` програмно реалізоване з типом `AutoField`, представляє номер, що унікально ідентифікує збереження зборів;
- поле `gatheringfk` програмно реалізоване з типом `ForeignKey`, представляє збори, які збережено (визначається як зовнішній ключ з моделі зборів `Gathering`);

– поле `userfk` програмно реалізоване з типом `ForeignKey`, представляє користувацький запис, який зберіг збори (визначається як зовнішній ключ з моделі користувацьких записів `Customeruser`).

Модель `Visitorrequest` реалізує в складі програмного додатку запити між користувацькими записами після відвідування зборів зі структурою:

– поле `id` програмно реалізоване з типом `AutoField`, представляє номер, що унікально ідентифікує запит між користувацькими записами після відвідування зборів;

– поле `userfk` програмно реалізоване з типом `ForeignKey`, представляє користувацький запис, який створив запит між користувацькими записами після відвідування зборів (визначається як зовнішній ключ з моделі користувацьких записів `Customeruser`);

– поле `usertext` програмно реалізоване з типом `CharField`, представляє повідомлення, яке в запиті було направлено автором, що створив запит;

– поле `usertofk` програмно реалізоване з типом `ForeignKey`, представляє користувацький запис, якому адресовано запит між користувацькими записами після відвідування зборів (визначається як зовнішній ключ з моделі користувацьких записів `Customeruser`);

– поле `usertotext` програмно реалізоване з типом `CharField`, представляє повідомлення, яке у відповіді на запит було направлено користувачем, який відповів на запит;

– поле `variant` програмно реалізоване з типом `SmallIntegerField`, представляє стан відповіді на запит (без відповіді, підтверджено, скасовано).

Модель `Gathercomment` реалізує в складі програмного додатку коментар до зборів зі структурою:

– поле `id` програмно реалізоване з типом `AutoField`, представляє номер, що унікально ідентифікує коментар до зборів;

– поле `userfk` програмно реалізоване з типом `ForeignKey`, представляє користувацький запис, який створив коментар (визначається як зовнішній ключ з моделі користувацьких записів `Customeruser`);

- поле `gatheringfk` програмно реалізоване з типом `ForeignKey`, представляє збори, до яких створено цей коментар (визначається як зовнішній ключ з моделі зборів `Gathering`);

- поле `commtime` програмно реалізоване з типом `DateTimeField`, представляє час створення коментаря до зборів;

- поле `commtext` програмно реалізоване з типом `CharField`, представляє текст коментаря до зборів.

Модель `Gatherscore` реалізує в складі програмного додатку оцінювання зборів зі структурою:

- поле `id` програмно реалізоване з типом `AutoField`, представляє номер, що унікально ідентифікує оцінювання зборів;

- поле `userfk` програмно реалізоване з типом `ForeignKey`, представляє користувацький запис, який виконав оцінювання (визначається як зовнішній ключ з моделі користувацьких записів `Customeruser`);

- поле `gatheringfk` програмно реалізоване з типом `ForeignKey`, представляє збори, які оцінено (визначається як зовнішній ключ з моделі зборів `Gathering`);

- поле `mark` програмно реалізоване з типом `SmallIntegerField`, представляє виставлену оцінку зборам;

- поле `scoretime` програмно реалізоване з типом `DateTimeField`, представляє час оцінювання зборів.

Модель `Communitycomment` реалізує в складі програмного додатку коментар до спільноти відвідувачів зборів зі структурою:

- поле `id` програмно реалізоване з типом `AutoField`, представляє номер, що унікально ідентифікує коментар коментар до спільноти відвідувачів зборів;

- поле `userfk` програмно реалізоване з типом `ForeignKey`, представляє користувацький запис, який створив коментар (визначається як зовнішній ключ з моделі користувацьких записів `Customeruser`);

- поле `communityfk` програмно реалізоване з типом `ForeignKey`, представляє спільноту відвідувачів зборів, до якої адресовано цей коментар

(визначається як зовнішній ключ з моделі спільноти відвідувачів зборів Community);

- поле `commtime` програмно реалізоване з типом `DateTimeField`, представляє час створення коментаря до спільноти відвідувачів зборів;

- поле `commtext` програмно реалізоване з типом `CharField`, представляє текст коментаря до спільноти відвідувачів зборів.

Програмно реалізовані описані нижче представлення (view) додатку переважно з заповненням шаблонів вебсторінок (template), для яких визначена розмітка за шаблоном [10].

Демонстрування переліку збережених зборів забезпечено функцією представлення `saved_gatherings_view`. Параметр `registered` значенням -1 вказує на необхідність визначення всіх зборів, а значення від 0 і більше (цілочисельне) вказує на потрібний тип реєстрації (zareєстрований, в очікуванні, скасований). Параметр `variant` задає потрібний набір зборів: значення 0 відповідає новим зборам (впорядковуються від найпізніших), 1 відповідає зборам, що мають відбутися не раніше поточної дати, 2 – архівним зборам, 3 – зборам, які мають відбутися протягом найближчого тижня з поточного часу. Якщо користувач авторизувався, то для кожного збору також у атрибуті спільноти `auth_organizer` встановлюється, чи він є організатором зборів, `auth_visitor` – чи він є учасником зборів.

Створення спільноти відвідувачів зборів забезпечено функцією представлення `new_gathering_community_view` на основі створеного класу форми `CommunityForm` та шаблону `create_gathering_community.html`, куди така форма передається для заповнення даних спільноти.

Редагування інформації про власну спільноту відвідувачів зборів забезпечено функцією представлення `edit_gathering_community_view`, номер спільноти задано параметром `gath_community_id`. Підтримка також виконується через форму `CommunityForm`, але збереження внесених через форму даних виконується тільки в тому випадку, якщо користувач, який авторизувався, є власником (автором) спільноти або організатором спільноти.

Демонстрування переліку спільнот, учасниками яких є користувач, що авторизувався, забезпечено функцією представлення `user_visitor_communities_view`. Вибираються всі спільноти, до яких входить як відвідувач користувач, проте до яких доступ користувача окремо не заблокований. Спільноти впорядковуються за назвою за алфавітом.

Демонстрування переліку зборів, що організовуються спільнотами, в яких є учасником користувач, що авторизувався, забезпечено функцією представлення `reg_community_gatherings_view`. Параметр `registered` значенням `-1` вказує на необхідність визначення всіх зборів, а значення від `0` і більше (цілочисельне) вказує на потрібний тип реєстрації (zareestrovaniy, в очікуванні, скасований). Параметр `variant` задає потрібний набір зборів: значення `0` відповідає новим зборам (впорядковуються від найпізніших), `1` відповідає зборам, що мають відбутися не раніше поточної дати, `2` – архівним зборам, `3` – зборам, які мають відбутися протягом найближчого тижня з поточного часу. Якщо користувач авторизувався, то для кожного збору також у атрибуті спільноти `auth_organizer` встановлюється, чи він є організатором зборів, `auth_visitor` – чи він є учасником зборів, `auth_save` визначає, чи збережено збір користувачем.

Демонстрування переліку спільнот, в яких користувач, що авторизувався, є організатором забезпечено функцією представлення `user_organizer_communities_view`. Вибираються всі спільноти, в яких користувач або є організатором або які він самостійно створив у програмі, при цьому відкидаються спільноти, які є прихованими. Спільноти впорядковуються за назвою за алфавітом.

Демонстрування інформації про спільноту відвідувачів зборів забезпечено функцією представлення `gather_community_view`. Номер самої спільноти відвідувачів зборів визначається параметром цієї функції `gath_community_id`. Вибираються дані про спільноту відвідувачів зборів, серії зборів у цій спільноті відвідувачів зборів, збори у цій спільноті відвідувачів зборів. Якщо користувач авторизувався, то для нього також у атрибуті

спільноти `auth_organizer` встановлюється, чи він є організатором спільноти відвідувачів зборів, `auth_visitor` – чи він є учасником спільноти відвідувачів зборів, `number_comments` – кількість створених коментарів щодо цієї спільноти відвідувачів зборів. Усі дані заповнюють шаблон вебсторінки `gathercommunity.html`.

Демонстрування переліку зборів, які організовані спільнотою, забезпечено функцією представлення `community_gatherings_view`. Номер спільноти визначається параметром цієї функції `gath_community_id`. Визначаються всі збори, які реалізуються саме заданою спільнотою. Параметр `variant` задає потрібний набір зборів: значення 0 відповідає зборам, що мають відбутися не раніше поточної дати, 1 – архівним зборам, 2 – новим зборам (впорядковуються від найпізніших), 3 – зборам, які мають відбутися протягом найближчого тижня з поточного часу. Якщо користувач авторизувався, то для кожного збору також у атрибуті спільноти `auth_organizer` встановлюється, чи він є організатором спільноти відвідувачів зборів, `auth_visitor` – чи він є учасником спільноти відвідувачів зборів, `number_comments` – кількість створених коментарів щодо цієї спільноти відвідувачів зборів. Усі дані заповнюють шаблон вебсторінки `community_gatherings.html`.

Демонстрування переліку серій зборів, які організовані спільнотою, забезпечено функцією представлення `community_series_view`. Номер спільноти визначається параметром цієї функції `gath_community_id`. Визначаються всі серії збори, які реалізуються саме заданою спільнотою. Параметр `variant` визначає потрібний варіант сортування серій: 0 – спершу найкращі, 1 – спершу найгірші, 2 – за алфавітом за назвами серій, 3 – у протилежному порядку за алфавітом за назвами серій. Результатом роботи виступає заповнений вказаними даними шаблон вебсторінки `community_series.html`.

Демонстрування інформації про збори забезпечено функцією представлення `gathering_view`, де номер самих зборів визначається

параметром `gathering_id`. До результатів на сторінці `gathering.html` входить об'єкт самих зборів `gathering` та перелік відвідувачів зборів `visitors`. Якщо користувач авторизувався, то для кожного збору також у атрибуті спільноти `auth_organizer` встановлюється, чи він є організатором зборів, `auth_visitor` – чи він є учасником зборів, `auth_save` – відмітка про збереження цих зборів користувачем. Якщо менше 1 доби пройшло з моменту початку зборів, то в атрибуті `auth_friend` істинне значення визначає можливість встановлення контакту з іншим відвідувачем цих зборів.

Демонстрування переліку зборів з серії забезпечено функцією представлення `series_gatherings_view`. Параметр `gatheringseries_id` визначає номер самої серії зборів. Параметр `registered` значенням `-1` вказує на необхідність визначення всіх зборів, а значення від `0` і більше (цілочисельне) вказує на потрібний тип реєстрації (zareєстрований, в очікуванні, скасований). Параметр `variant` задає потрібний набір зборів: значення `0` відповідає новим зборам (впорядковуються від найпізніших), `1` відповідає зборам, що мають відбутися не раніше поточної дати, `2` – архівним зборам, `3` – зборам, які мають відбутися протягом найближчого тижня з поточного часу. Якщо користувач авторизувався, то для кожного збору також у атрибуті спільноти `auth_organizer` встановлюється, чи він є організатором зборів, `auth_visitor` – чи він є учасником зборів, `auth_save` визначає, чи збережено збір користувачем.

Демонстрування переліку коментарів до зборів забезпечено функцією представлення `gathering_comments_view`. Параметр `gathering_id` необхідний для встановлення номеру зборів. Окрім формування переліку, який передається у шаблон вебсторінки `gatheringcomments.html`, туди ж також передається форма для створення коментаря. За збором також встановлюється, чи є поточний користувач організатором, відвідувачем зборів, чи зберіг він ці збори, а також виставлена оцінка або можливість встановлення контакту з іншими користувачами (якщо пройшов час не більший за добу з моменту початку зборів).

Демонстрування переліку коментарів до спільноти відвідувачів зборів забезпечено функцією представлення `community_comments_view`. Параметр `gath_community_id` необхідний для встановлення номеру спільноти відвідувачів зборів. Окрім формування переліку, який передається у шаблон вебсторінки `commcomments.html`, туди ж також передається форма для створення коментаря. За спільнотою також встановлюється, чи є поточний користувач організатором, учасником спільноти.

Демонстрування учасників зборів забезпечено функцією представлення `gathering_visitors_view`. Параметр `gathering_id` необхідний для встановлення номеру зборів. Якщо не більше ніж доба пройшла з часу початку зборів, то всі учасники зборів аналізуються щодо того, чи було авторизованим під час роботи з представленням користувачем відправлено запит або отримано запит на встановлення контакту від відповідного учасника зборів. Окрім того за користувачем, який авторизувався, перевіряється, чи він є організатором або учасником цих зборів, або зберіг ці збори для себе. Також встановлюється, чи виставив оцінку цим зборам користувач, який авторизувався: якщо виставив, то визначається, яка саме, та повертається для об'єкту зборів у атрибуті `auth_score`.

Демонстрування інформації про користувача забезпечено функцією представлення `user_presentation_view`. Логін такого користувацького запису визначається параметром `presented_username`. За користувацьким записом визначаються спільноти, до яких він входить як організатор, спільноти, до яких він входить як учасник, збори, на які він зареєструвався. Якщо користувач, який авторизувався, має той самий обліковий запис, що запитується, то також надається форма для редагування даних користувацького запису.

Демонстрування переліку адресованих або відправлених запитів забезпечено функцією представлення `requests_after_gatherings_view`. Параметр `variant` визначає тип переліку: значення 0 відповідає відправленим користувачам запитам, значення 1 відповідає адресованим користувачам

запитам. Параметр `state` повинен мати значення `-1`, коли потрібно повернути всі запити, або конкретне значення, що відповідає стану запитів.

Демонстрування переліку зборів, на які зареєструвався користувач забезпечено функцією представлення `registered_gatherings_view`. Параметр `registered` значенням `-1` вказує на необхідність визначення всіх зборів, а значення від `0` і більше (цілочисельне) вказує на потрібний тип реєстрації (зареєстрований, в очікуванні, скасований). Параметр `variant` задає потрібний набір зборів: значення `0` відповідає новим зборам (впорядковуються від найпізніших), `1` відповідає зборам, що мають відбутися не раніше поточної дати, `2` – архівним зборам, `3` – зборам, які мають відбутися протягом найближчого тижня з поточного часу. Якщо користувач авторизувався, то для кожного збору також у атрибуті спільноти `auth_organizer` встановлюється, чи він є організатором зборів, `auth_visitor` – чи він є учасником зборів, `auth_save` визначає, чи збережено збір користувачем. Усі дані заповнюють шаблон вебсторінки `reggatherings.html`.

Додавання учасника до спільноти забезпечено функцією представлення `add_community_visitor_view`. Параметр `gath_community_id` визначає номер спільноти відвідувачів зборів. Користувацький запис визначається з авторизації. У випадку, якщо учасник вже доданий до спільноти, то він вилючається. Параметр `block` за замовчуванням має значення `False`. Якщо значення `True`, то користувач не додається, а блокується. Це стосується і випадків, коли він вже є учасником спільноти, і випадків, коли він ще не був учасником.

Додавання організатора власної спільноти відвідувачів зборів забезпечено функцією представлення `add_community_organizer_view`. Параметр `gath_community_id` визначає номер спільноти відвідувачів зборів. Параметр `organizer_username` визначає логін користувацького запису, який має стати новим організатором спільноти відвідувачів зборів. Перевіряється те, що користувач, який авторизувався, є власником спільноти або її діючим організатором. Якщо це так, то користувач додається до організаторів. Якщо

він вже є організатором, то в результаті його організаторська роль скасовується.

Демонстрування переліку створених користувачем зборів забезпечено функцією представлення `creator_gatherings_view`. Параметр `variant` задає потрібний набір зборів: значення 0 відповідає новим зборам (впорядковуються від найпізніших), 1 відповідає зборам, що мають відбутися не раніше поточної дати, 2 – архівним зборам, 3 – зборам, які мають відбутися протягом найближчого тижня з поточного часу. Якщо користувач авторизувався, то для кожного збору також у атрибуті спільноти `auth_organizer` встановлюється, чи він є організатором зборів, `auth_visitor` – чи він є учасником зборів. Усі дані заповнюють шаблон вебсторінки `creatorgatherings.html`.

Створення серії зборів власної спільноти забезпечено функцією представлення `create_series_view` на основі створеного класу форми `SeriesForm` та шаблону `create_series.html`, куди така форма передається для заповнення даних серії зборів. Форма повертається тільки тоді, коли користувач, що авторизувався, є власником або організатором даної спільноти. Номер спільноти задано параметром `gath_community_id`.

Редагування серії зборів забезпечено функцією представлення `edit_series_view`. Номер серії зборів задано параметром `series_id`. Підтримка також виконується через форму `SeriesForm`, але збереження внесених через форму даних виконується тільки в тому випадку, якщо користувач, який авторизувався, є власником (автором) спільноти або організатором спільноти.

Створення запису про заплановані збори у власній спільноті забезпечено функцією представлення `create_gathering_view` на основі створеного класу форми `GatheringForm` та шаблону `create_gathering.html`, куди така форма передається для заповнення даних зборів. Форма повертається тільки тоді, коли користувач, що авторизувався, є власником або організатором спільноти. Номер спільноти задано параметром `gath_community_id`.

Редагування запису про заплановані збори у власній спільноті забезпечено функцією представлення `edit_gathering_view`. Номер зборів задано параметром `gathering_id`. Підтримка також виконується через форму `GatheringForm`, але збереження внесених через форму даних виконується тільки в тому випадку, якщо користувач, який авторизувався, є власником (автором) спільноти або організатором спільноти, до якої належать ці збори.

Реєстрацію участі користувача в зборах разом з додаванням до переліку очікування зборів забезпечено функцією представлення `add_gathering_visitor_view`. Параметр `gathering_id` визначає номер зборів. Користувацький запис визначається з авторизації. Якщо кількість місць на зборах обмежена і вона вже заповнена, то реєстрація не відбувається. Якщо за доступом відбувається контроль, то користувач додається в чергу очікування. У випадку, якщо учасник вже доданий до зборів, то він вилючається. Те ж саме стосується і перебування в черзі очікування. Параметр `block` за замовчуванням має значення `False`. Якщо значення `True`, то користувач не додається, а блокується. Це стосується і випадків, коли він вже є учасником зборів, і випадків, коли він ще не був учасником.

Підтвердження участі в зборах власної спільноти з переліку очікування забезпечено функцією представлення `gathering_wait_submit_view`. Номер участі користувача в зборах спільноти задається параметром `gath_id`. Відбувається перевірка того, що користувач, який авторизувався, є власником або організатором спільноти, до якої належать збори. Якщо параметр `variant` має значення `False` (за замовчуванням `True`), то відбувається не підтвердження, а відхилення.

Пошук зборів забезпечено функцією представлення `gathering_search_view`. Пошук відбувається тільки серед зборів, які ще не розпочалися. Серед параметрів задається місто проведення, проміжок дат та тип проведення (онлайн або офлайн) або варіант доступу (вільні або з реєстрацією, або з придбанням квитків).

Пошук спільнот відвідувачів зборів забезпечено функцією представлення `community_search_view`. Серед параметрів задається місто, в якому функціонує ця спільнота відвідувачів зборів. Пошук виконується тільки серед видимих спільнот відвідувачів зборів.

Збереження зборів забезпечено функцією представлення `gathering_save_view`. Параметр `gathering_id` визначає номер зборів. Якщо збори цим користувачем збережено, то збереження вилючається.

Оцінювання зборів за результатами проведення забезпечено функцією представлення `gathering_mark_view`. Параметр `gathering_id` визначає номер зборів. Перед виставленням оцінки перевіряється, що користувач є серед учасників зборів, при чому не серед тих, хто залишився у черзі очікування або участь якого була відхилена.

Підтвердження адресованого користувачу запиту після спільного відвідування зборів забезпечено функцією представлення `gather_request_approve_view`. Номер запиту визначає параметр `gather_request_id`. Перевіряється, що запит адресований тому користувачу, який авторизувався і працює з представленням. Параметр `variant` за замовчуванням має значення `True`. Якщо значення `False`, то відбувається відхилення. В основі працює форма `RequestForm`.

Створення запиту після спільного відвідування зборів забезпечено функцією представлення `create_visitor_request_view` на основі створеного класу форми `RequestForm`. Дана форма в підсумку повертається у складі відповідної сторінки тільки тоді, коли користувач, що авторизувався, взяв участь у тому зборі, до учасника якого адресовано його запит, а до того ж ці збори відбулись не пізніше ніж 1 добу тому (фактично почались за 24 години до поточного моменту). Номер участі користувача в зборах задано параметром `gath_visitor_id`.

### **3.3 Висновки за розділом 3**

Програмне забезпечення пошуку і організації зборів за шаблоном MTV визначає відповідні структурні одиниці, необхідні для забезпечення функціональних можливостей, що були визначені за результатами проєктування.

## **4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ**

### **4.1 Призначення програми**

Створена програма призначена для надання комплексу функціональних можливостей організації зборів, що включають надання інформації про них, керування учасниками, для організаторів зборів, пошуку зборів, що включають пошук зборів за розширеними критеріями, визначення власного відвідування зборів.

### **4.2 Умови виконання програми**

Мінімальні технічні засоби визначені на рівні, що має забезпечувати приблизно 50 запитів на секунду:

- процесор не менше Intel Xeon E5440 з частотою 2.83 ГГц;
- оперативна пам'ять – 12 Гб;
- жорсткий диск – 40 Гб простору.

Експлуатація можлива тільки за використання сервера, на якому можливо встановити і в подальшому встановлено засоби, які включають інтерпретатор Python версії 3, засоби створення вебдодатків на основі Django, забезпечено доступ до системи керування базами даних PostgreSQL з можливістю програмного створення бази даних засобами Django.

### **4.3 Експлуатація програми**

Підхід до роботи з програмою однаковий для користувачів з будь-якими наявними правами, хоч користувач є власником чи організатором спільноти, зборів, хоч він є лише відвідувачем зборів. У будь-якому випадку користувач має всі можливості, доступні через головне меню додатку.

Якщо не вказується конкретна сторінка, а тільки вказується адреса вебсервера, за якою розташований додаток, то відкривається сторінка з описом можливостей програми.

Для того щоб створити нову спільноту відвідувачів зборів, потрібно обрати з меню “Спільноти” підрозділ “Створити”. Скористатись цими можливостями може будь-який користувач. Після цього отримати доступ до переліку всіх спільнот відвідувачів зборів, де користувач є організатором, можна через меню “Спільноти” і підрозділ “Організатор”. Якщо користувач бажає переглянути перелік тих спільнот відвідувачів зборів, де він є тільки учасником (не організатором, власником), то підрозділ меню “Участь” дозволяє відкрити саме такий перелік. Зазвичай при пошуку спільнот потрібно використовувати підрозділ меню “Пошук” з розділу “Спільноти”. З результатів можна відкрити будь-яку спільноту на окремій сторінці (рис. 4.1).

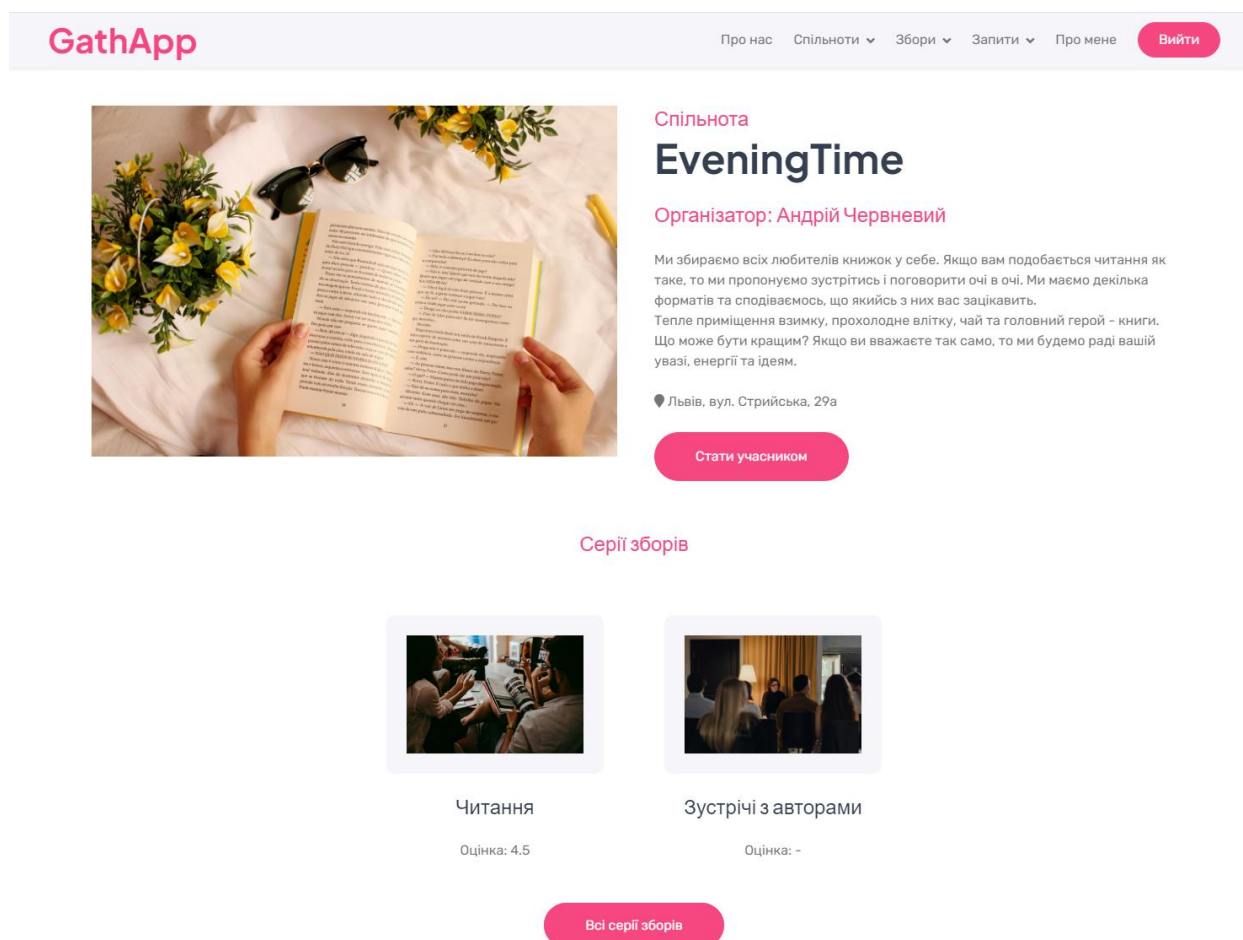


Рисунок 4.1 – Демонстрування інформації про спільноту відвідувачів зборів

На сторінці спільноти відвідувачів зборів знаходиться її назва, опис, інформація про місце проведення, фотографія. Якщо користувач, який переглядає спільноту відвідувачів зборів, не є учасником її, то відображається кнопка “Стати учасником”. У протилежному випадку відображається кнопка “Залишити спільноту”. Далі наводиться перелік серій зборів. Кожну таку серію можна відкрити на окремій сторінці з переліком конкретних зборів, які до цієї серії відносяться. Перейти до відкриття сторінки з усіма серіями зборів цієї спільноти відвідувачів зборів можна за допомогою кнопки “Всі серії зборів”. Далі знаходиться перелік конкретних зборів. Відображаються тільки 8, а до всіх зборів спільноти можна перейти за допомогою кнопки “Всі збори”. Далі знаходиться перелік учасників спільноти. Відображаються тільки 8, перейти до всіх можна за допомогою кнопок “Всі організатори” та “Всі учасники”. Якщо користувач є організатором спільноти, то йому також відображається кнопка “Додати організатора”. Далі знаходиться перелік останніх коментарів. Перейти до переліку всіх коментарів до спільноти можна за допомогою кнопки “Усі коментарі”.

Розділ меню “Збори” дозволяє перейти до перегляду збережених користувачем зборів з підрозділу “Збережені”, до перегляду зборів, на які зареєструвався як учасник користувач, з підрозділу “Реєстрації”, до перегляду створених цим користувачем зборів з підрозділу “Створені” (рис. 4.2), до пошуку зборів з підрозділу “Пошук”, до перегляду зборів, які створені спільнотами, учасником яких є користувач, з підрозділу “Спільнотські”.

Зі сторінки створених користувачем зборів можна перейти до редагування кожного збору. Також можна зберегти збір (на фотографії у правому нижньому куті відображається або порожнє кола, коли збір не був збережений, або галочка у колі, коли збір збережений, відповідно лівіше показується позначка плюс, коли користувач є відвідувачем зборів, або мінус, коли навпаки).

Пошук зборів передбачає подібне відображення зборів (рис. 4.3).

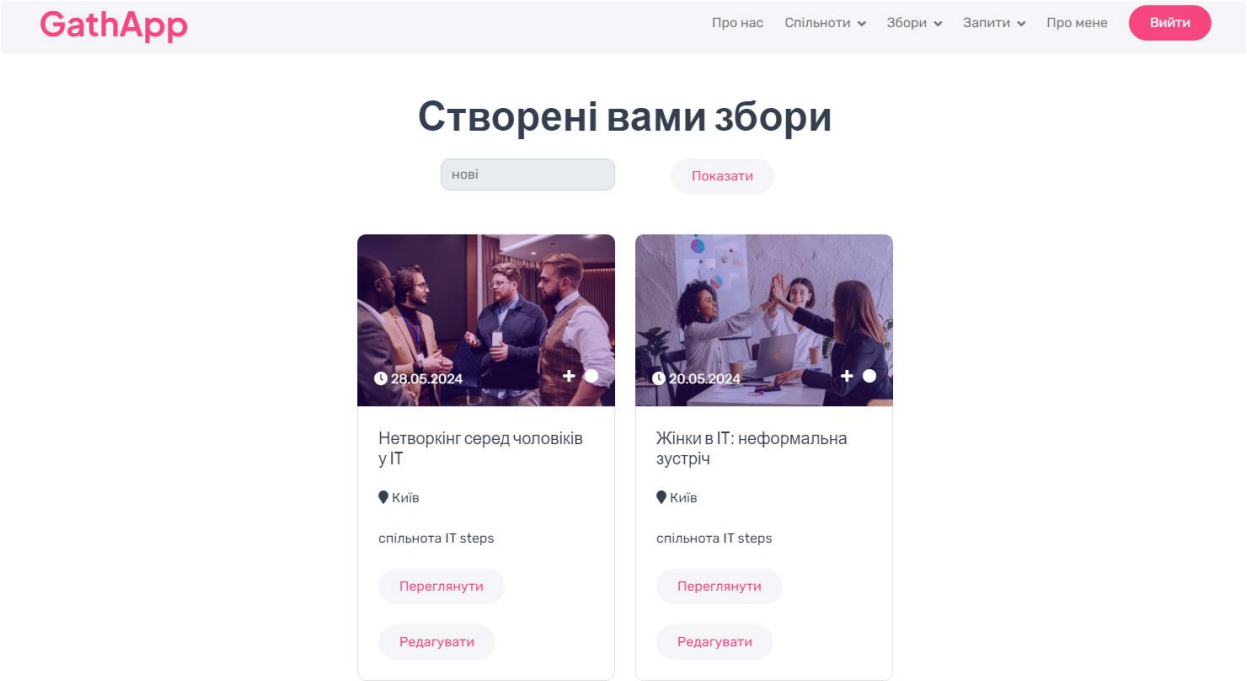


Рисунок 4.2 – Демонстрування створених користувачем зборів

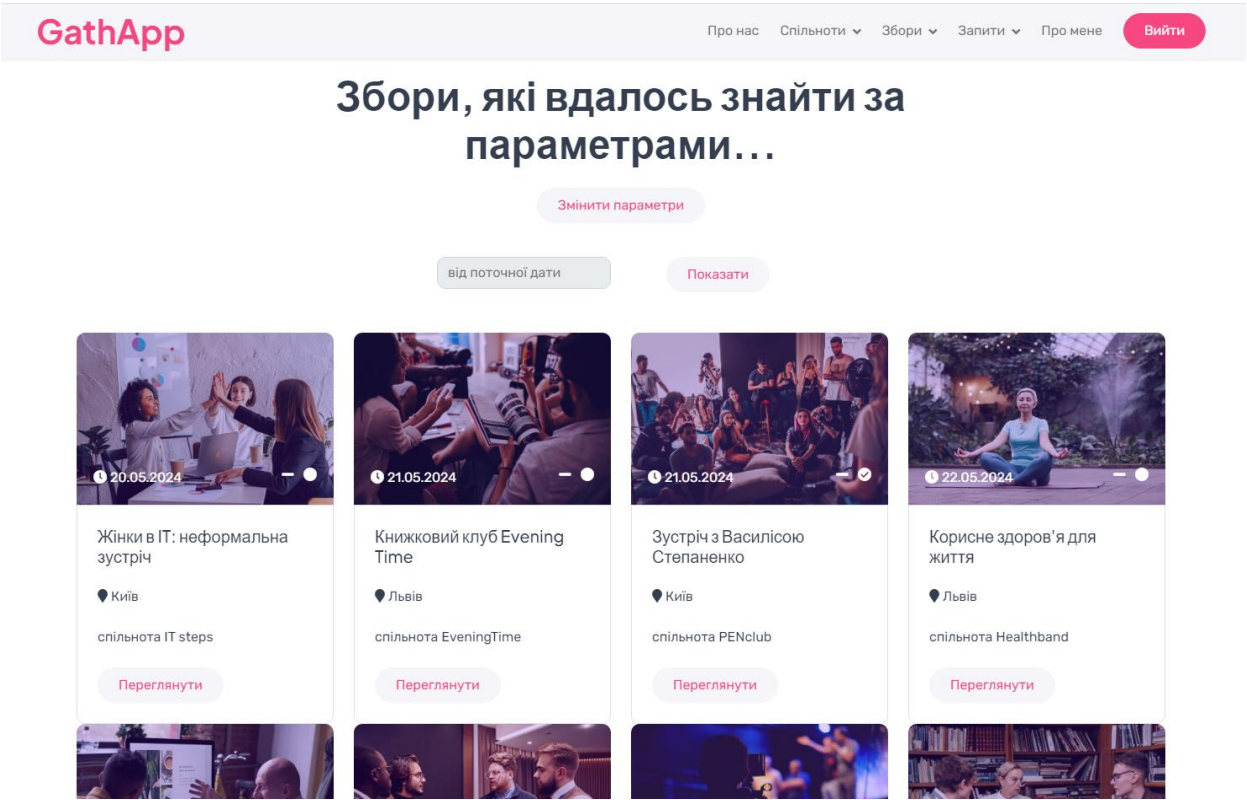


Рисунок 4.3 – Пошук зборів

Якщо користувач відкриває перелік всіх відвідувачів, і він є організатором спільноти, в якій збори організовуються, то йому

демонструється перелік відвідувачів з можливістю керування. Якщо створюється черга очікування, то з неї він за допомогою позначок плюс або мінус на фотографіях користувачів, які додалися до черги очікування, може надавати дозвіл або навпаки забороняти користувачу для відвідування зборів (рис. 4.4).

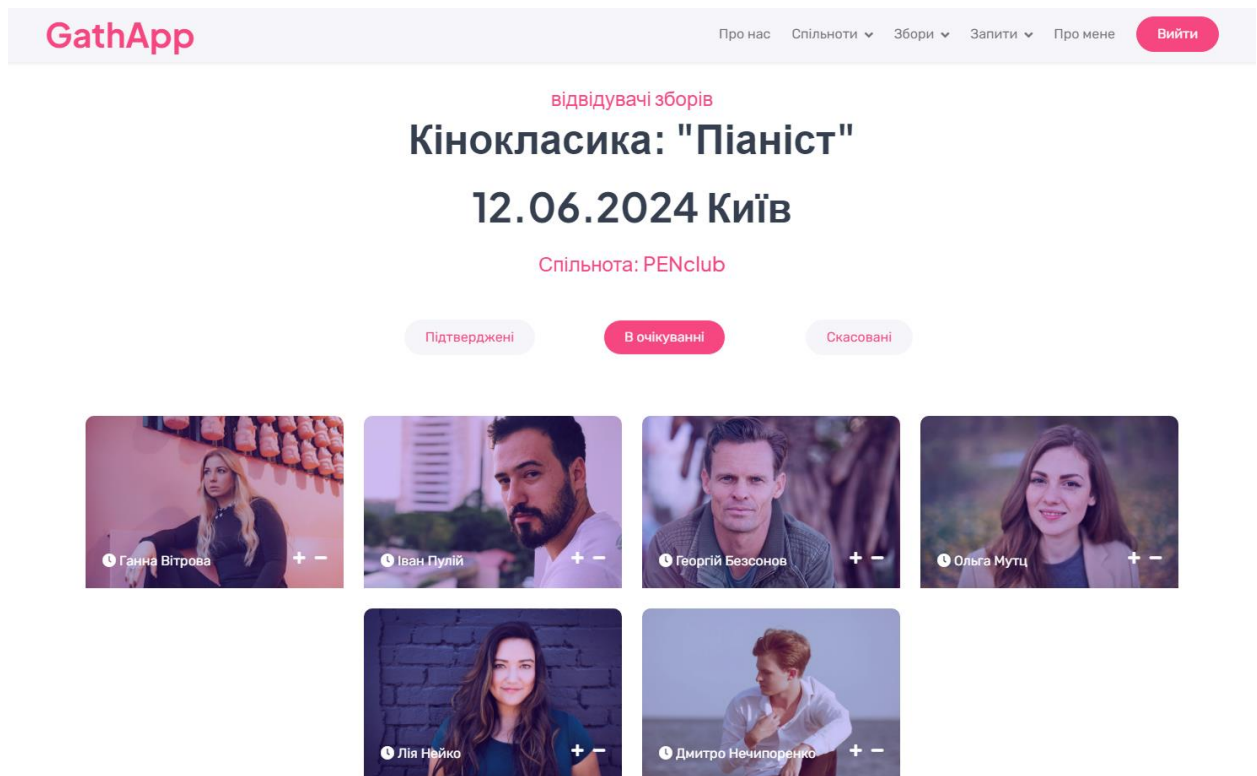


Рисунок 4.4 – Керування чергою очікування відвідувачів зборів

#### 4.4 Тестування програми

Тестування програми забезпечувалось через розміщення програми на сервері та виконання звернень за тими сценаріями, які були передбачені результатами моделювання (табл. 4.1). Кожен сценарій перевірявся визначенням відповідного звернення, виконанням дій за сценарієм, а тоді співставленням очікуваного результату з тим, що було отримано. Тестування було успішно завершено.

Таблиця 4.1 – Результати тестування програми пошуку і організації зборів

| Функціональна можливість                                          | Результат |
|-------------------------------------------------------------------|-----------|
| 1                                                                 | 2         |
| Створення спільноти відвідувачів зборів                           | Успіх     |
| Редагування інформації про власну спільноту відвідувачів зборів   | Успіх     |
| Додавання учасника до спільноти відвідувачів зборів               | Успіх     |
| Вилучення учасника зі спільноти відвідувачів зборів               | Успіх     |
| Блокування учасника в спільноті відвідувачів зборів               | Успіх     |
| Демонстрування переліку спільнот, учасниками яких є користувач    | Успіх     |
| Демонстрування інформації про спільноту відвідувачів зборів       | Успіх     |
| Демонстрування переліку зборів, які організовані спільнотою       | Успіх     |
| Демонстрування переліку серій зборів, які організовані спільнотою | Успіх     |
| Додавання організатора власної спільноти відвідувачів зборів      | Успіх     |
| Скасування організатора спільноти відвідувачів зборів             | Успіх     |
| Створення серії зборів власної спільноти                          | Успіх     |
| Редагування серії зборів                                          | Успіх     |
| Створення запису про заплановані збори у власній спільноті        | Успіх     |
| Редагування інформації про заплановані збори                      | Успіх     |
| Скасування проведення зборів                                      | Успіх     |
| Демонстрування інформації про збори                               | Успіх     |
| Демонстрування учасників зборів                                   | Успіх     |

Продовження таблиці 4.1

| 1                                                                                             | 2     |
|-----------------------------------------------------------------------------------------------|-------|
| Реєстрація участі в зборах                                                                    | Успіх |
| Скасування участі в зборах                                                                    | Успіх |
| Додавання до переліку очікування зборів                                                       | Успіх |
| Вилучення з переліку очікування зборів                                                        | Успіх |
| Підтвердження участі в зборах власної спільноти з переліку очікування                         | Успіх |
| Відхилення участі в зборах власної спільноти з переліку очікування                            | Успіх |
| Пошук зборів                                                                                  | Успіх |
| Демонстрування переліку зборів з серії                                                        | Успіх |
| Демонстрування переліку зборів, на які зареєструвався користувач                              | Успіх |
| Демонстрування переліку зборів, що організовуються спільнотами, в яких є учасником користувач | Успіх |
| Демонстрування переліку створених користувачем зборів                                         | Успіх |
| Демонстрування інформації про користувача                                                     | Успіх |
| Редагування даних користувацького запису                                                      | Успіх |
| Збереження зборів                                                                             | Успіх |
| Демонстрування переліку збережених зборів                                                     | Успіх |
| Створення запиту після спільного відвідування зборів                                          | Успіх |
| Скасування запиту після спільного відвідування зборів                                         | Успіх |
| Підтвердження адресованого користувачу запиту після спільного відвідування зборів             | Успіх |
| Відхилення адресованого користувачу запиту після спільного відвідування зборів                | Успіх |
| Демонстрування переліку адресованих запитів                                                   | Успіх |
| Демонстрування переліку відправлених запитів                                                  | Успіх |

Продовження таблиці 4.1

| 1                                                                   | 2     |
|---------------------------------------------------------------------|-------|
| Авторизація в програмі                                              | Успіх |
| Реєстрація користувачького запису                                   | Успіх |
| Оцінювання зборів за результатами проведення                        | Успіх |
| Демонстрування переліку коментарів до зборів                        | Успіх |
| Демонстрування переліку коментарів до спільноти відвідувачів зборів | Успіх |
| Демонстрування переліку спільнот, в яких користувач є організатором | Успіх |
| Пошук спільнот відвідувачів зборів                                  | Успіх |

#### 4.5 Висновки за розділом 4

Програма пошуку і організації зборів була успішно перевірена щодо реалізації тих сценаріїв, які після моделювання роботи користувача мали бути реалізовані. Усі функції працюють.

## ВИСНОВКИ

Програму розроблено з врахуванням функціональних можливостей і особливостей аналогів, включаючи програми Meetup і Kommunity, з виділенням відмінностей розробки, використано технології, які адекватно дозволили втілити запропоновані рішення в межах роботи.

Спільне використання мови Python та фреймворку Django сприяє швидкій розробці в межах дипломної кваліфікаційної роботи з виконанням розробки в дуже обмежені строки, сприяє забезпеченню функціонування потужного додатку. З використанням системи PostgreSQL виконано проєктування і забезпечення роботи з БД. Прецеденти відповідних діаграм визначені для моделювання програмного забезпечення.

Програму реалізовано повністю стосовно тих сценаріїв використання, які були визначені разом з альтернативними сценаріями та представлені на діаграмі прецедентів. Виконано відповідне тестування, що дозволило перевірити реалізацію цих здатностей.

Розроблена програма пошуку і організації зборів надає можливості створення спільнот відвідувачів зборів, створення серій зборів та планування окремих зборів, додавання учасників до спільнот та реєстрації для проведення зборів, у тому числі створюючи черги очікування за обмеження кількості учасників та необхідності підтвердження участі, оцінювання зборів, що дозволяє визначати оцінки для серій зборів, демонстрування зборів, на які зареєструвався користувач, в яких він є організатором, створення персональних запитів після спільного відвідування зборів, коментування зборів і спільнот, організовуючи таким чином спілкування.

У роботі досягнуто мету, яка полягала в тому, щоб розробити програмне забезпечення пошуку і організації зборів для сприяння живій або дистанційній комунікації у спільнотах.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Meetup. Find Local Groups, Events, and Activities Near You [Electronic resource]. – Access mode : <https://www.meetup.com/>.
2. Let's share together. Kommunity [Electronic resource]. – Access mode : <https://kommunity.com/>.
3. What Is Python Used For? A Beginner's Guide [Electronic resource]. – Access mode : <https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python>.
4. Dinder, M. Becoming an Enterprise Django Developer [Text] : Discover best practices, tooling, and solutions for writing and organizing Django applications in production / M. Dinder. – Birmingham : Packt Publishing, 2022. – 526 p.
5. User Guide – Falcon 3.1.3 documentation [Electronic resource]. – Access mode : <https://falcon.readthedocs.io/en/stable/user/index.html>.
6. Sridhar, S. The Falcon Web Framework – Should You Use It? [Electronic resource] / S. Sridhar. – Access mode : <https://medium.com/affinityanswers-tech/the-falcon-web-framework-should-you-use-it-df3eaae78679>.
7. Gour, R. Working Structure of Django MTV Architecture [Electronic resource] / R. Gour. – Access mode : <https://towardsdatascience.com/working-structure-of-django-mtv-architecture-a741c8c64082>.
8. Obe, R. PostgreSQL [Text] : Up and Running : A Practical Guide to the Advanced Open Source Database / R. Obe, L. Hsu. – Sebastopol, California : O'Reilly Media, 2017. – 312 p.
9. Ferrari, L. Learn PostgreSQL – Second Edition [Text] : Use, manage and build secure and scalable databases with PostgreSQL 16 / L. Ferrari, E. Pirozzi. – Birmingham : Packt Publishing, 2023. – 744 p.
10. Mailler – SaaS Business Website Template [Electronic resource]. – Access mode : <https://htmlcodex.com/saas-website-template/>.

**ДОДАТОК А**  
**Технічне завдання**

## **Вступ**

Сучасні умови комунікації потребують інших засобів надання інформації про збори людей, пошуку таких зборів. У великому місті може відбуватися велика кількість зборів, через що знайти потрібний достатньо важко. Концертні заходи мають зазвичай власні вебсайти для пошуку і придбання квитків на такі заходи. Те ж стосується театральних подій, наприклад. У той же час через це некомерційні збори людей втрачають увагу. Знайти таку інформацію стає достатньо складно. Вона може бути представлена, наприклад, у Facebook. Проте віднайти потрібні збори достатньо складно у випадках, коли подій у місті відбувається багато. Окрім того в поточних умовах нерідко збори мають ще й дистанційну форму. Усе це потребує певного впорядкування, а відповідно і окремого програмного інструменту, який би дозволив як виносити інформацію про свої збори, так і керувати ними, долучати учасників, виконувати пошук таких зборів за різними параметрами, які можуть включати не тільки назву, конкретне або загальне (місто) місце проведення зборів.

### **A.1 Підстави для розробки**

Наказом № 168 від 24 квітня 2024 р. за Національним університетом «Запорізька політехніка» визначено тему «Програмне забезпечення для пошуку і організації зборів» для виконання дипломної кваліфікаційної роботи.

### **A.2 Призначення розробки**

Виконувана у роботі розробка призначена для надання комплексу функціональних можливостей організації зборів, що включають надання інформації про них, керування учасниками, для організаторів зборів, пошуку

зборів, що включають пошук зборів за розширеними критеріями, визначення власного відвідування зборів.

### **A.3 Основні вимоги до програми**

#### **A.3.1 Вимоги до функціональних характеристик**

Функціональні характеристики програми пошуку і організації зборів повинні включати:

- створення спільноти відвідувачів зборів;
- додавання або видалення учасника до або зі спільноти;
- редагування інформації про спільноту відвідувачів зборів;
- демонстрування переліку спільнот, учасниками яких є користувач;
- демонстрування інформації про спільноту відвідувачів зборів;
- демонстрування переліку зборів, які організовані спільнотою;
- демонстрування переліку серій зборів, які організовані спільнотою;
- додавання або скасування організатора спільноти відвідувачів зборів;
- створення серії зборів або її редагування;
- створення запису про заплановані збори;
- редагування інформації про заплановані збори, включаючи скасування проведення зборів;
- демонстрування інформації про збори;
- демонстрування учасників зборів;
- реєстрація участі в зборах або її скасування;
- додавання до переліку очікування зборів або скасування;
- підтвердження або відхилення участі в зборах з переліку очікування;
- пошук зборів;
- демонстрування переліку зборів з серії;
- демонстрування переліку зборів, на які зареєструвався користувач;
- демонстрування переліку зборів, що організовуються спільнотами, в яких є учасником користувач;

- демонстрування переліку створених користувачем зборів;
- демонстрування інформації про користувача;
- збереження зборів або скасування такого збереження;
- демонстрування переліку збережених зборів;
- створення, підтвердження або відхилення запиту після спільного відвідування зборів;
- демонстрування переліку адресованих або відправлених запитів;
- реєстрація користувацького запису;
- редагування даних користувацького запису;
- авторизація в програмі;
- оцінювання зборів за результатами проведення;
- коментування зборів;
- демонстрування переліку коментарів до зборів;
- коментування спільноти відвідувачів зборів;
- демонстрування переліку коментарів до спільноти відвідувачів зборів;
- демонстрування переліку спільнот, в яких користувач є організатором;
- пошук спільнот відвідувачів зборів.

### **A.3.2 Умови експлуатації**

Експлуатація можлива тільки за використання сервера, на якому можливо встановити і в подальшому встановлено засоби, які включають інтерпретатор Python версії 3, засоби створення вебдодатків на основі Django, забезпечено доступ до системи керування базами даних PostgreSQL з можливістю програмного створення бази даних засобами Django.

### **А.3.3 Вимоги до складу та параметрів технічних засобів**

Технічні засоби визначені на рівні, що має забезпечувати приблизно 50 запитів на секунду:

- процесор не менше Intel Xeon E5440 з частотою 2.83 ГГц;
- оперативна пам'ять – 12 Гб;
- жорсткий диск – 40 Гб простору.

### **А.4 Порядок контролю та приймання**

Виконання дипломної кваліфікаційної роботи бакалавра має бути закінчено у восьми тижневий термін з розподіленням за етапами, що визначено завданням на роботу у відповідному календарному плані. Після виконання даного плану реалізується процедура захисту роботи. Захист відбувається перед екзаменаційною комісією.

**ДОДАТОК Б**  
**Текст програми**

```

from .models import Communityvisitor
from django.shortcuts import render
from .models import Visitorrequest
from .models import Gatheringseries
from django.shortcuts import redirect
from .models import Community
from django.shortcuts import get_object_or_404
from .models import Communitycomment
from .models import Gatheringvisitor
from gathapp.forms import CommunityForm
from .models import Customeruser
from django.db.models import Q
from .models import Gathering
from .models import Gatheringsave
from gathapp.forms import CommunityCommentForm
from .models import Gathercomment
from django.urls import reverse
from .models import GatheringCommentForm

def new_gathering_community_view(request):
    if request.method == "POST":
        gathering_community_form = CommunityForm(request.POST)
        if gathering_community_form.is_valid():
            gathering_community =
gathering_community_form.save(commit=False)
            customer_user =
Customeruser.objects.get(username=request.user.username)
            gathering_community.creatorfk = customer_user
            gathering_community.save()
            return redirect(request.path_info)
        else:
            gathering_community_form = CommunityForm()
            return render(request,
"gathapp/create_gathering_community.html", {"community_form":
gathering_community_form })

def edit_gathering_community_view(request, gath_community_id):
    if request.user.is_active:
        customer_user =
Customeruser.objects.get(username=request.user.username)
        try:
            gathering_community =
Community.objects.prefetch_related("organizers").get(id =
gath_community_id)
        except Community.DoesNotExist:
            gathering_community = None

```

```

        if gathering_community and ((customer_user in
gathering_community.organizers) or (gathering_community.creatorfk ==
customer_user)):
            gathering_community_form =
CommunityForm(request.POST or None, instance = gathering_community)
            if request.method == "POST" and
gathering_community_form.is_valid():
                gathering_community_form.save()
                return redirect('gathcommunityview', id =
gathering_community.id)
            else:
                gathering_community_form = None
            return render(request,
"gathapp/edit_gathering_community.html", {"community_form":
gathering_community_form })

def user_visitor_communities_view(request):
    if request.user.is_active:
        customer_user =
Customeruser.objects.get(username=request.user.username)
        user_visitor_communities = []
        communities = Communityvisitor.objects.filter(userfk =
customer_user, block =
False).select_related('communityfk').values_list('communityfk__id')
        visitor_communities = Community.objects.filter(id__in =
communities)
        visitor_communities =
visitor_communities.order_by("fullcaption")
        return render(request,
"gathapp/visitorcommunities.html", {"communities":
visitor_communities})
    else:
        return render(request,
"gathapp/visitorcommunities.html", {"communities": []})

def user_organizer_communities_view(request):
    if request.user.is_active:
        customer_user =
Customeruser.objects.get(username=request.user.username)
        user_organizer_communities = []
        organizer_communities =
Community.objects.select_related("creatorfk").select_related("gatherci
tyfk").prefetch_related("organizers").filter(Q(organizers__in =
[customer_user]) | Q(creatorfk = customer_user))
        organizer_communities =
organizer_communities.filter(visible = True)

```

```

        organizer_communities =
organizer_communities.order_by("fullcaption")
        return render(request,
"gathapp/organizercommunities.html", {"communities":
organizer_communities})
    else:
        return render(request,
"gathapp/organizercommunities.html", {"communities": []})

def gather_community_view(request, gath_community_id):
    if gath_community_id:
        gathering_community =
Community.objects.prefetch_related("organizers").prefetch_related("vis
itors").select_related("creatorfk").select_related("gathercityfk").get
(id = gath_community_id)
        gathering_community.number_comments =
Communitycomment.objects.filter(communityfk =
gathering_community).count()
        gathering_series =
Gatheringseries.objects.filter(communityfk =
gathering_community).order_by("-avscore")
        import datetime
        gather_time = datetime.datetime.now()
        gatherings =
Gathering.objects.filter(getheringseriesfk__communityfk =
gathering_community, starttime__gte =
gather_time).select_related("gathercityfk").select_related("creatorfk"
).select_related("gatheringseriesfk").order_by("starttime")
        gathering_community.auth_score = 0
        if request.user.is_active:
            customer_user =
Customeruser.objects.get(username=request.user.username)
            if customer_user in gathering_community.organizers
or gathering_community.creatorfk in customer_user:
                gathering_community.auth_organizer = True
            else:
                gathering_community.auth_organizer = False
            if customer_user in gathering_community.visitors:
                gathering_community.auth_visitor = True
            else:
                gathering_community.auth_visitor = False
        else:
            gathering_community.auth_organizer = False
            gathering_community.auth_visitor = False
        return render(request, "gathapp/gathercommunity.html",
{"community": gathering_community, "series": gathering_series,
"gatherings": gatherings })

```

```

    else:
        return render(request, "gathapp/gathercommunity.html",
{"community": None, "series": [], "gatherings": [] })

    def community_gatherings_view(request, gath_community_id,
variant = 0):
        if gath_community_id:
            gathering_community =
Community.objects.prefetch_related("organizers").select_related("creat
ork").select_related("gathercityfk").get(id = gath_community_id)
            import datetime
            gather_time = datetime.datetime.now()
            week_time = datetime.datetime.now() +
datetime.timedelta(days = 7)
            #Від поточної дати
            if variant == 0:
                gatherings =
Gathering.objects.filter(gatheringseriesfk__communityfk =
gathering_community, starttime__gte =
gather_time).prefetch_related("organizers").prefetch_related("visitors
").select_related("gathercityfk").select_related("creatorfk").select_r
elated("gatheringseriesfk").order_by("starttime")
                #архівні
            elif variant == 1:
                gatherings =
Gathering.objects.filter(gatheringseriesfk__communityfk =
gathering_community, starttime__lt =
gather_time).prefetch_related("organizers").prefetch_related("visitors
").select_related("gathercityfk").select_related("creatorfk").select_r
elated("gatheringseriesfk").order_by("starttime")
                #нові
            elif variant == 2:
                gatherings =
Gathering.objects.filter(gatheringseriesfk__communityfk =
gathering_community).prefetch_related("organizers").prefetch_related("
visitors").select_related("gathercityfk").select_related("creatorfk").
select_related("gatheringseriesfk").order_by("-starttime")
                #тиждень
            elif variant == 3:
                gatherings =
Gathering.objects.filter(gatheringseriesfk__communityfk =
gathering_community, starttime__gte = gather_time, starttime__lte =
week_time).prefetch_related("organizers").prefetch_related("visitors")
.select_related("gathercityfk").select_related("creatorfk").select_rel
ated("gatheringseriesfk").order_by("-starttime")
                if request.user.is_active:

```

```

        customer_user =
Customeruser.objects.get(username=request.user.username)
        for gathering in gatherings:
            if customer_user in gathering.organizers or
gathering.creatorfk in customer_user:
                gathering.auth_organizer = True
            else:
                gathering.auth_organizer = False
            if customer_user in gathering.visitors:
                gathering.auth_visitor = True
            else:
                gathering.auth_visitor = False
            gath_saved =
Gatheringsave.objects.filter(gatheringfk = gathering, userfk =
customer_user)

            if gath_saved:
                gathering.auth_save = True
            else:
                gathering.auth_save = False
        return render(request,
"gathapp/community_gatherings.html", {"community":
gathering_community, "gatherings": gatherings })
    else:
        return render(request,
"gathapp/community_gatherings.html", {"community": None, "gatherings":
[] })

def community_series_view(request, gath_community_id, variant =
0):
    if gath_community_id:
        gathering_community =
Community.objects.prefetch_related("organizers").select_related("creat
orfk").select_related("gathercityfk").get(id = gath_community_id)
        #найкращі
        if variant == 0:
            gathering_series =
Gatheringseries.objects.filter(communityfk = gathering_community,
visible = True).order_by("-avscore")
            #найгірші
            elif variant == 1:
                gathering_series =
Gatheringseries.objects.filter(communityfk = gathering_community,
visible = True).order_by("avscore")
            #алфавіт
            elif variant == 2:

```

```

        gathering_series =
Gatheringseries.objects.filter(communityfk = gathering_community,
visible = True).order_by("fullcaption")
        #зворотній алфавіт
        elif variant == 4:
            gathering_series =
Gatheringseries.objects.filter(communityfk = gathering_community,
visible = True).order_by("-fullcaption")
            return render(request, "gathapp/community_series.html",
{"community": gathering_community, "series": gathering_series })
        else:
            return render(request,
"gathapp/community_gatherings.html", {"community": None, "series": []
})

def gathering_view(request, gathering_id):
    if gathering_id:
        gathering =
Gathering.objects.prefetch_related("organizers").prefetch_related("vis
itors").select_related("gatheringseriesfk").select_related("gathercity
fk").select_related("gathercityfk__countryfk").select_related("creator
fk").select_related("gatheringseriesfk__communityfk").get(id =
gathering_id)
        gathering.number_comments =
Gathercomment.objects.filter(gatheringfk = gathering).count()
        gathering_visitors =
Gatheringvisitor.objects.filter(gatheringfk = gathering, variant =
1).order_by("-id")
        gathering.auth_score = 0
        if request.user.is_active:
            customer_user =
Customeruser.objects.get(username=request.user.username)
            if customer_user in gathering.organizers or
gathering.creatorfk in customer_user:
                gathering.auth_organizer = True
            else:
                gathering.auth_organizer = False
            if customer_user in gathering.visitors:
                gathering.auth_visitor = True
            else:
                gathering.auth_visitor = False
            gath_saved =
Gatheringsave.objects.filter(gatheringfk = gathering, userfk =
customer_user)
            if gath_saved:
                gathering.auth_save = True
            else:

```

```

        gathering.auth_save = False
        import datetime
        gath_time = datetime.datetime.now() -
datetime.timedelta(days = 1)
        if gathering.starttime >= gath_time and
gathering.starttime <= datetime.datetime.now():
            gathering.auth_friend = True
        else:
            gathering.auth_friend = False
            scores = Gatherscore.objects.get(userfk =
customer_user, gatheringfk = gathering)
            gathering.auth_score = scores.mark
        else:
            gathering.auth_organizer = False
            gathering.auth_visitor = False
            gathering.auth_save = False
            return render(request, "gathapp/gathering.html",
{"gathering": gathering, "visitors": gathering_visitors })
        else:
            return render(request, "gathapp/gathering.html",
{"gathering": None, "visitors": [] })

def creator_gatherings_view(request, variant = 0):
    if request.user.is_active:
        customer_user =
Customeruser.objects.get(username=request.user.username)
        creator_gatherings = []
        import datetime
        gather_time = datetime.datetime.now()
        week_time = datetime.datetime.now() +
datetime.timedelta(days = 7)
        #НОВІ
        if variant == 0:
            creator_gatherings =
Gathering.objects.prefetch_related("organizers").prefetch_related("vis
itors").select_related("creatorfk").select_related("gatheringseriesfk"
).select_related("gathercityfk").filter(creatorfk =
customer_user).order_by("-starttime")
            #Від поточної дати
        elif variant == 1:
            creator_gatherings =
Gathering.objects.prefetch_related("organizers").prefetch_related("vis
itors").select_related("creatorfk").select_related("gatheringseriesfk"
).select_related("gathercityfk").filter(creatorfk = customer_user,
starttime__gte = gather_time).order_by("-starttime")
            #апріс
        elif variant == 2:

```

```

        creator_gatherings =
Gathering.objects.prefetch_related("organizers").prefetch_related("vis
itors").select_related("creatorfk").select_related("gatheringseriesfk"
).select_related("gathercityfk").filter(creatorfk = customer_user,
starttime__lt = gather_time).order_by("-starttime")
        #тиждень
        elif variant == 3:
            creator_gatherings =
Gathering.objects.prefetch_related("organizers").prefetch_related("vis
itors").select_related("creatorfk").select_related("gatheringseriesfk"
).select_related("gathercityfk").filter(creatorfk = customer_user,
starttime__gte = gather_time, starttime__lte = week_time).order_by("-
starttime")
        for gathering in creator_gatherings:
            if customer_user in gathering.visitors:
                gathering.auth_visitor = True
            else:
                gathering.auth_visitor = False
            gath_saved =
Gatheringsave.objects.filter(gatheringfk = gathering, userfk =
customer_user)
            if gath_saved:
                gathering.auth_save = True
            else:
                gathering.auth_save = False
        return render(request, "gathapp/creatorgatherings.html",
{"gatherings": creator_gatherings })
    else:
        return render(request, "gathapp/creatorgatherings.html",
{"gatherings": []})

def registered_gatherings_view(request, variant = 0, registered
= -1):
    if request.user.is_active:
        customer_user =
Customeruser.objects.get(username=request.user.username)
        registered_gatherings = []
        import datetime
        gather_time = datetime.datetime.now()
        week_time = datetime.datetime.now() +
datetime.timedelta(days = 7)
        if registered == -1:
            gatherings = Gatheringvisitor.objects.filter(userfk=
customer_user).values_list("gatheringfk__id")
        else:
            gatherings = Gatheringvisitor.objects.filter(userfk=
customer_user, variant = registered).values_list("gatheringfk__id")

```

```

        #нові
        if variant == 0:
            registered_gatherings =
Gathering.objects.prefetch_related("organizers").prefetch_related("vis
itors").select_related("creatorfk").select_related("gatheringseriesfk"
).select_related("gathercityfk").filter(id__in =
gatherings).order_by("-starttime")
        #від поточної дати
        elif variant == 1:
            registered_gatherings =
Gathering.objects.prefetch_related("organizers").prefetch_related("vis
itors").select_related("creatorfk").select_related("gatheringseriesfk"
).select_related("gathercityfk").filter(id__in = gatherings,
starttime__gte = gather_time).order_by("-starttime")
        #апдів
        elif variant == 2:
            registered_gatherings =
Gathering.objects.prefetch_related("organizers").prefetch_related("vis
itors").select_related("creatorfk").select_related("gatheringseriesfk"
).select_related("gathercityfk").filter(id__in = gatherings,
starttime__lt = gather_time).order_by("-starttime")
        #тиждень
        elif variant == 2:
            registered_gatherings =
Gathering.objects.prefetch_related("organizers").prefetch_related("vis
itors").select_related("creatorfk").select_related("gatheringseriesfk"
).select_related("gathercityfk").filter(id__in = gatherings,
starttime__gte = gather_time, starttime__lte = week_time).order_by("-
starttime")

        for gathering in registered_gatherings:
            if customer_user in gathering.organizers or
gathering.creatorfk in customer_user:
                gathering.auth_organizer = True
            else:
                gathering.auth_organizer = False
            if customer_user in gathering.visitors:
                gathering.auth_visitor = True
            else:
                gathering.auth_visitor = False
            gath_saved =
Gatheringsave.objects.filter(gatheringfk = gathering, userfk =
customer_user)

            if gath_saved:
                gathering.auth_save = True
            else:
                gathering.auth_save = False

```

```

        return render(request, "gathapp/reggatherings.html",
{"gatherings": registered_gatherings })
    else:
        return render(request, "gathapp/reggatherings.html",
{"gatherings": []})

def reg_community_gatherings_view(request, variant = 0,
registered = -1):
    if request.user.is_active:
        customer_user =
Customeruser.objects.get(username=request.user.username)
        user_visitor_communities = []
        user_visitor_communities =
Communityvisitor.objects.filter(userfk = customer_user, block =
False).select_related('communityfk').values_list('communityfk__id')
        visitor_communities = Community.objects.filter(id__in =
user_visitor_communities)
        registered_gatherings = []
        import datetime
        gather_time = datetime.datetime.now()
        week_time = datetime.datetime.now() +
datetime.timedelta(days = 7)
        if registered == -1:
            gatherings = Gatheringvisitor.objects.filter(userfk=
customer_user, gatheringseriesfk__communityfk__in =
visitor_communities).values_list("gatheringfk__id")
        else:
            gatherings = Gatheringvisitor.objects.filter(userfk=
customer_user, gatheringseriesfk__communityfk__in =
visitor_communities, variant =
registered).values_list("gatheringfk__id")
        #НОВІ
        if variant == 0:
            registered_gatherings =
Gathering.objects.prefetch_related("organizers").prefetch_related("vis
itors").select_related("creatorfk").select_related("gatheringseriesfk"
).select_related("gathercityfk").filter(id__in =
gatherings).order_by("-starttime")
            #Від поточної дати
        elif variant == 1:
            registered_gatherings =
Gathering.objects.prefetch_related("organizers").prefetch_related("vis
itors").select_related("creatorfk").select_related("gatheringseriesfk"
).select_related("gathercityfk").filter(id__in = gatherings,
starttime__gte = gather_time).order_by("-starttime")
            #апріс
        elif variant == 2:

```

```

        registered_gatherings =
Gathering.objects.prefetch_related("organizers").prefetch_related("vis
itors").select_related("creatorfk").select_related("gatheringseriesfk"
).select_related("gathercityfk").filter(id__in = gatherings,
starttime__lt = gather_time).order_by("-starttime")
        #тиждень
        elif variant == 2:
            registered_gatherings =
Gathering.objects.prefetch_related("organizers").prefetch_related("vis
itors").select_related("creatorfk").select_related("gatheringseriesfk"
).select_related("gathercityfk").filter(id__in = gatherings,
starttime__gte = gather_time, starttime__lte = week_time).order_by("-
starttime")
        for gathering in registered_gatherings:
            if customer_user in gathering.organizers or
gathering.creatorfk in customer_user:
                gathering.auth_organizer = True
            else:
                gathering.auth_organizer = False
            if customer_user in gathering.visitors:
                gathering.auth_visitor = True
            else:
                gathering.auth_visitor = False
            gath_saved =
Gatheringsave.objects.filter(gatheringfk = gathering, userfk =
customer_user)
            if gath_saved:
                gathering.auth_save = True
            else:
                gathering.auth_save = False
        return render(request,
"gathapp/regcommunitygatherings.html", {"gatherings":
registered_gatherings })
        else:
            return render(request,
"gathapp/regcommunitygatherings.html", {"gatherings": []})

def saved_gatherings_view(request, variant = 0, registered = -
1):
    if request.user.is_active:
        customer_user =
Customeruser.objects.get(username=request.user.username)
        saved_gatherings = Gatheringsave.objects.filter(userfk =
customer_user).select_related('gatheringfk').values_list('gatheringfk_
_id')

        import datetime
        gather_time = datetime.datetime.now()

```

```

        week_time = datetime.datetime.now() +
datetime.timedelta(days = 7)
        if registered == -1:
            gatherings =
Gatheringvisitor.objects.filter(gatheringfk__in = saved_gatherings,
userfk = customer_user, gatheringseriesfk__communityfk__in =
visitor_communities).values_list("gatheringfk__id")
        else:
            gatherings =
Gatheringvisitor.objects.filter(gatheringfk__in = saved_gatherings,
userfk= customer_user, gatheringseriesfk__communityfk__in =
visitor_communities, variant =
registered).values_list("gatheringfk__id")
            #нові
            if variant == 0:
                registered_gatherings =
Gathering.objects.prefetch_related("organizers").prefetch_related("vis
itors").select_related("creatorfk").select_related("gatheringseriesfk"
).select_related("gathercityfk").filter(id__in =
gatherings).order_by("-starttime")
                #від поточної дати
                elif variant == 1:
                    registered_gatherings =
Gathering.objects.prefetch_related("organizers").prefetch_related("vis
itors").select_related("creatorfk").select_related("gatheringseriesfk"
).select_related("gathercityfk").filter(id__in = gatherings,
starttime__gte = gather_time).order_by("-starttime")
                    #apxib
                    elif variant == 2:
                        registered_gatherings =
Gathering.objects.prefetch_related("organizers").prefetch_related("vis
itors").select_related("creatorfk").select_related("gatheringseriesfk"
).select_related("gathercityfk").filter(id__in = gatherings,
starttime__lt = gather_time).order_by("-starttime")
                        #тиждень
                        elif variant == 2:
                            registered_gatherings =
Gathering.objects.prefetch_related("organizers").prefetch_related("vis
itors").select_related("creatorfk").select_related("gatheringseriesfk"
).select_related("gathercityfk").filter(id__in = gatherings,
starttime__gte = gather_time, starttime__lte = week_time).order_by("-
starttime")
                            for gathering in registered_gatherings:
                                if customer_user in gathering.organizers or
gathering.creatorfk in customer_user:
                                    gathering.auth_organizer = True
                                else:

```

```

        gathering.auth_organizer = False
    if customer_user in gathering.visitors:
        gathering.auth_visitor = True
    else:
        gathering.auth_visitor = False
    return render(request, "gathapp/savedgatherings.html",
{"gatherings": registered_gatherings })
    else:
        return render(request, "gathapp/savedgatherings.html",
{"gatherings": []})

def series_gatherings_view(request, gatheringseries_id, variant
= 0, registered = -1):
    if request.user.is_active:
        customer_user =
Customeruser.objects.get(username=request.user.username)
        series = Gatheringseries.objects.get(id =
gatheringseries_id).select_related("communityfk").select_related("comm
unityfk__gathercityfk").select_related("communityfk__creatorfk")
        series_gatherings = []
        if registered == -1:
            series_gatherings =
Gatheringvisitor.objects.filter(gatheringfk__gatheringseriesfk =
series).values_list("gatheringfk__id")
        else:
            series_gatherings =
Gatheringvisitor.objects.filter(gatheringfk__gatheringseriesfk =
series, variant = registered).values_list("gatheringfk__id")
            import datetime
            gather_time = datetime.datetime.now()
            week_time = datetime.datetime.now() +
datetime.timedelta(days = 7)
            #нові
            if variant == 0:
                serie_gatherings =
Gathering.objects.prefetch_related("organizers").prefetch_related("vis
itors").select_related("creatorfk").select_related("gatheringseriesfk"
).select_related("gathercityfk").filter(id__in =
series_gatherings).order_by("-starttime")
                #від поточної дати
            elif variant == 1:
                serie_gatherings =
Gathering.objects.prefetch_related("organizers").prefetch_related("vis
itors").select_related("creatorfk").select_related("gatheringseriesfk"
).select_related("gathercityfk").filter(id__in = series_gatherings,
starttime__gte = gather_time).order_by("-starttime")
            #apxiб

```

```

        elif variant == 2:
            serie_gatherings =
Gathering.objects.prefetch_related("organizers").prefetch_related("vis
itors").select_related("creatorfk").select_related("gatheringseriesfk"
).select_related("gathercityfk").filter(id__in = series_gatherings,
starttime__lt = gather_time).order_by("-starttime")
            #тиждень
        elif variant == 2:
            serie_gatherings =
Gathering.objects.prefetch_related("organizers").prefetch_related("vis
itors").select_related("creatorfk").select_related("gatheringseriesfk"
).select_related("gathercityfk").filter(id__in = series_gatherings,
starttime__gte = gather_time, starttime__lte = week_time).order_by("-
starttime")
        for gathering in serie_gatherings:
            if customer_user in gathering.organizers or
gathering.creatorfk in customer_user:
                gathering.auth_organizer = True
            else:
                gathering.auth_organizer = False
            if customer_user in gathering.visitors:
                gathering.auth_visitor = True
            else:
                gathering.auth_visitor = False
            gath_saved =
Gatheringsave.objects.filter(gatheringfk = gathering, userfk =
customer_user)
            if gath_saved:
                gathering.auth_save = True
            else:
                gathering.auth_save = False
        return render(request, "gathapp/seriesgatherings.html",
{"gatherings": serie_gatherings })
    else:
        return render(request, "gathapp/seriesgatherings.html",
{"gatherings": []})

def gathering_comments_view(request, gathering_id):
    if gathering_id:
        gathering =
Gathering.objects.prefetch_related("organizers").prefetch_related("vis
itors").select_related("gatheringseriesfk").select_related("gathercity
fk").select_related("gathercityfk__countryfk").select_related("creator
fk").select_related("gatheringseriesfk__communityfk").get(id =
gathering_id)
        gathering.auth_score = 0
        if request.user.is_active:

```

```

        customer_user =
Customeruser.objects.get(username=request.user.username)
        if request.method == "POST":
            gathering_comment_form =
GatheringCommentForm(request.POST)
            if gathering_comment_form.is_valid():
                gathering_comment =
gathering_comment_form.save(commit=False)
                gathering_comment.userfk = customer_user
                gathering_comment.save()
                return redirect(reverse('gathcommentsview',
kwargs={ 'gathering_id': gathering_id })))
            else:
                gathering_comment_form = GatheringCommentForm()
                if customer_user in gathering.organizers or
gathering.creatorfk in customer_user:
                    gathering.auth_organizer = True
                else:
                    gathering.auth_organizer = False
                if customer_user in gathering.visitors:
                    gathering.auth_visitor = True
                else:
                    gathering.auth_visitor = False
                gath_saved =
Gatheringsave.objects.filter(gatheringfk = gathering, userfk =
customer_user)
                if gath_saved:
                    gathering.auth_save = True
                else:
                    gathering.auth_save = False
                import datetime
                gath_time = datetime.datetime.now() -
datetime.timedelta(days = 1)
                if gathering.starttime >= gath_time and
gathering.starttime <= datetime.datetime.now():
                    gathering.auth_friend = True
                else:
                    gathering.auth_friend = False
                scores = Gatherscore.objects.get(userfk =
customer_user, gatheringfk = gathering)
                gathering.auth_score = scores.mark
            else:
                gathering.auth_organizer = False
                gathering.auth_visitor = False
                gathering.auth_save = False

```

```

        gathering_comments =
Gathercomment.objects.filter(gatheringfk =
gathering).order_by("commtime")
        return render(request, "gathapp/gatheringcomments.html",
{"gathering": gathering, "comments": gathering_comments, "form":
gathering_comment_form })
    else:
        return render(request, "gathapp/gatheringcomments.html",
{"gathering": None, "comments": [], "form": None })

def community_comments_view(request, gath_community_id):
    if gath_community_id:
        gathering_community =
Community.objects.prefetch_related("organizers").prefetch_related("vis
itors").select_related("creatorfk").select_related("gathercityfk").get
(id = gath_community_id)
        gathering_community.auth_score = 0
        if request.user.is_active:
            customer_user =
Customeruser.objects.get(username=request.user.username)
            if request.method == "POST":
                community_comment_form =
CommunityCommentForm(request.POST)
                if community_comment_form.is_valid():
                    community_comment =
community_comment_form.save(commit=False)
                    community_comment.userfk = customer_user
                    community_comment.save()
                    return
redirect(reverse('communitycommentsview', kwargs={
'gath_community_id': gath_community_id }))
            else:
                community_comment_form = CommunityCommentForm()
                if customer_user in gathering_community.organizers
or gathering_community.creatorfk in customer_user:
                    gathering_community.auth_organizer = True
                else:
                    gathering_community.auth_organizer = False
                if customer_user in gathering_community.visitors:
                    gathering_community.auth_visitor = True
                else:
                    gathering_community.auth_visitor = False
            else:
                gathering_community.auth_organizer = False
                gathering_community.auth_visitor = False

```

```

        community_comments =
Communitycomment.objects.filter(communityfk =
gathering_community).order_by("commtime")
        return render(request, "gathapp/commcomments.html",
{"gathering": gathering_community, "comments": community_comments,
"form": community_comment_form })
    else:
        return render(request, "gathapp/commcomments.html",
{"gathering": None, "comments": [], "form": None })

def gathering_visitors_view(request, gathering_id, variant):
    if gathering_id:
        gathering =
Gathering.objects.prefetch_related("organizers").prefetch_related("vis
itors").select_related("gatheringseriesfk").select_related("gathercity
fk").select_related("gathercityfk__countryfk").select_related("creator
fk").select_related("gatheringseriesfk__communityfk").get(id =
gathering_id)
        gathering_visitors =
Gatheringvisitor.objects.filter(gatheringfk = gathering, variant =
variant).order_by("-id")
        import datetime
        gath_time = datetime.datetime.now() -
datetime.timedelta(days = 1)
        gathering.auth_score = 0
        if request.user.is_active:
            customer_user =
Customeruser.objects.get(username=request.user.username)
            if gathering.starttime >= gath_time and
gathering.starttime <= datetime.datetime.now():
                auth_friend = True
            else:
                auth_friend = False
            if auth_friend:
                for visitor in gathering_visitors:
                    visitorrequests =
Visitorrequest.objects.filter(userfk = customer_user, usertofk =
visitor)

                    if visitorrequests:
                        visitor.to = True
                    else:
                        visitor.to = False
                    visitorrequests =
Visitorrequest.objects.filter(usertofk = customer_user, userfk =
visitor)

                    if visitorrequests:
                        visitor.from = True

```

```

        else:
            visitor.from = False
            if customer_user in gathering.organizers or
gathering.creatorfk in customer_user:
                gathering.auth_organizer = True
            else:
                gathering.auth_organizer = False
            if customer_user in gathering.visitors:
                gathering.auth_visitor = True
            else:
                gathering.auth_visitor = False
            gath_saved =
Gatheringsave.objects.filter(gatheringfk = gathering, userfk =
customer_user)
            if gath_saved:
                gathering.auth_save = True
            else:
                gathering.auth_save = False
            scores = Gatherscore.objects.get(userfk =
customer_user, gatheringfk = gathering)
            gathering.auth_score = scores.mark
        else:
            gathering.auth_organizer = False
            gathering.auth_visitor = False
            gathering.auth_save = False
            return render(request, "gathapp/gathervisitors.html",
{"gathering": gathering, "visitors": gathering_visitors })
        else:
            return render(request, "gathapp/gathervisitors.html",
{"gathering": None, "visitors": [] })

def user_presentation_view(request, presented_username):
    if presented_username:
        customer_user =
Customeruser.objects.get(username=presented_username)
        user_visitor_communities = []
        communities = Communityvisitor.objects.filter(userfk =
customer_user, block =
False).select_related('communityfk').values_list('communityfk__id')
        visitor_communities = Community.objects.filter(id__in =
communities)
        visitor_communities =
visitor_communities.order_by("fullcaption")
        user_organizer_communities = []
        organizer_communities =
Communityvisitor.objects.select_related("creatorfk").select_related("g

```

```

athercityfk").prefetch_related("organizers").filter(Q(organizers__in =
[customer_user]) | Q(creatorfk = customer_user))
    organizer_communities =
organizer_communities.filter(visible = True)
    organizer_communities =
organizer_communities.order_by("fullcaption")
    creator_gatherings =
Gathering.objects.select_related("gathercityfk").filter(creatorfk =
customer_user).order_by("-starttime")
    gatherings = Gatheringvisitor.objects.filter(userfk=
customer_user, variant = 1).values_list("gatheringfk__id")
    registered_gatherings =
Gathering.objects.select_related("gathercityfk").filter(id__in =
gatherings).order_by("-starttime")
    if request.user.is_active:
        own_customer_user =
Customeruser.objects.get(username=request.user.username)
        if own_customer_user == customer_user:
            customer_user_form =
CustomeruserForm(request.POST or None, instance = customer_user)
            if request.method == "POST" and
customer_user_form.is_valid():
                customer_user_form.save()
                return redirect(reverse('customeruserview',
kwargs={ 'presented_username': presented_username })))
            else:
                customer_user_form = None
                return render(request, "gathapp/user.html", {"user":
customer_user, "communities_visitor": visitor_communities,
"communities_organizer": organizer_communities, "gatherings":
registered_gatherings, "form": customer_user_form })
            else:
                return render(request, "gathapp/user.html", {"user":
None, "communities_visitor": [], "communities_organizer": [],
"gatherings": [], "form": None })

def requests_after_gatherings_view(request, variant = 0, state =
-1):
    if request.user.is_active:
        customer_user =
Customeruser.objects.get(username=request.user.username)
        if not variant and state == -1:
            visitor_requests =
Visitorrequest.objects.filter(userfk =
customer_user).select_related("usertofk").select_related("userfk").ord
er_by("-id")
            elif not variant:

```

```

        visitor_requests =
Visitorrequest.objects.filter(userfk = customer_user, variant =
state).select_related("usertofk").select_related("userfk").order_by("-
id")
        elif variant and state == -1:
            visitor_requests =
Visitorrequest.objects.filter(usertofk =
customer_user).select_related("usertofk").select_related("userfk").ord
er_by("-id")
        else:
            visitor_requests =
Visitorrequest.objects.filter(usertofk = customer_user, variant =
state).select_related("usertofk").select_related("userfk").order_by("-
id")
            return render(request, "gathapp/requests.html",
{"requests": visitor_requests })
        else:
            return render(request, "gathapp/requests.html",
{"requests": [] })

```

**ДОДАТОК В**  
**Слайди презентації**



Рисунок В.1 – Початковий слайд

## ОБ'ЄКТ, ПРЕДМЕТ, МЕТА, ЗАВДАННЯ РОБОТИ

Об'єкт роботи – процеси пошуку і організації зборів людей.

Предмет роботи – програмне забезпечення пошуку і організації зборів.

Мета роботи – розробити програмне забезпечення пошуку і організації зборів для сприяння живій або дистанційній комунікації у спільнотах.

Завдання роботи:

- проаналізувати проблему розробки програми пошуку і організації зборів;
- спроектувати програмне забезпечення для пошуку і організації зборів;
- реалізувати програмне забезпечення для пошуку і організації зборів та виконати його тестування.

Рисунок В.2 – Об'єкт, предмет, мета, завдання роботи

## ПОРІВНЯННЯ ІСНЮЮЧИХ АНАЛОГІВ ТА РОЗРОБЛЮВАНОЇ ПРОГРАМИ ДЛЯ ПОШУКУ І ОРГАНІЗАЦІЇ ЗБОРІВ

| Характеристика                    | Meetup | Kommunity | Розробка |
|-----------------------------------|--------|-----------|----------|
| Планування зборів                 | Так    | Так       | Так      |
| Створення спільнот                | Так    | Так       | Так      |
| Визначення організаторів спільнот | Так    | Так       | Так      |
| Створення серій зборів            | Так    | Так       | Так      |
| Створення черги очікування        | Так    | Так       | Так      |
| Оцінювання зборів у серії         | Так    | Так       | Так      |
| Створення контактів після зборів  | Так    | Ні        | Так      |
| Плата за використання             | Так    | Так       | Ні       |

3

Рисунок В.3 – Порівняння існуючих аналогів та розроблюваної програми для пошуку і організації зборів

## ПОРІВНЯННЯ ФРЕЙМВОРКІВ, НЕОБХІДНИХ ДЛЯ СТВОРЕННЯ ВЕБДОДАТКУ

| Характеристика                        | Falcon                                      | Django                     |
|---------------------------------------|---------------------------------------------|----------------------------|
| Створення бази даних з коду           | Не має базових засобів, потребує інтеграції | Присутнє за замовчуванням  |
| Керування даними з базової панелі     | Не має базових засобів, потребує інтеграції | Присутнє за замовчуванням  |
| Робота з формами                      | Присутня за замовчуванням                   | Присутня за замовчуванням  |
| Засоби авторизації                    | Не має базових засобів, потребує інтеграції | Присутні за замовчуванням  |
| Рушій для роботи з шаблонами сторінок | Не має базових засобів, потребує інтеграції | Присутній за замовчуванням |

4

Рисунок В.4 – Порівняння фреймворків, необхідних для створення вебдодатку

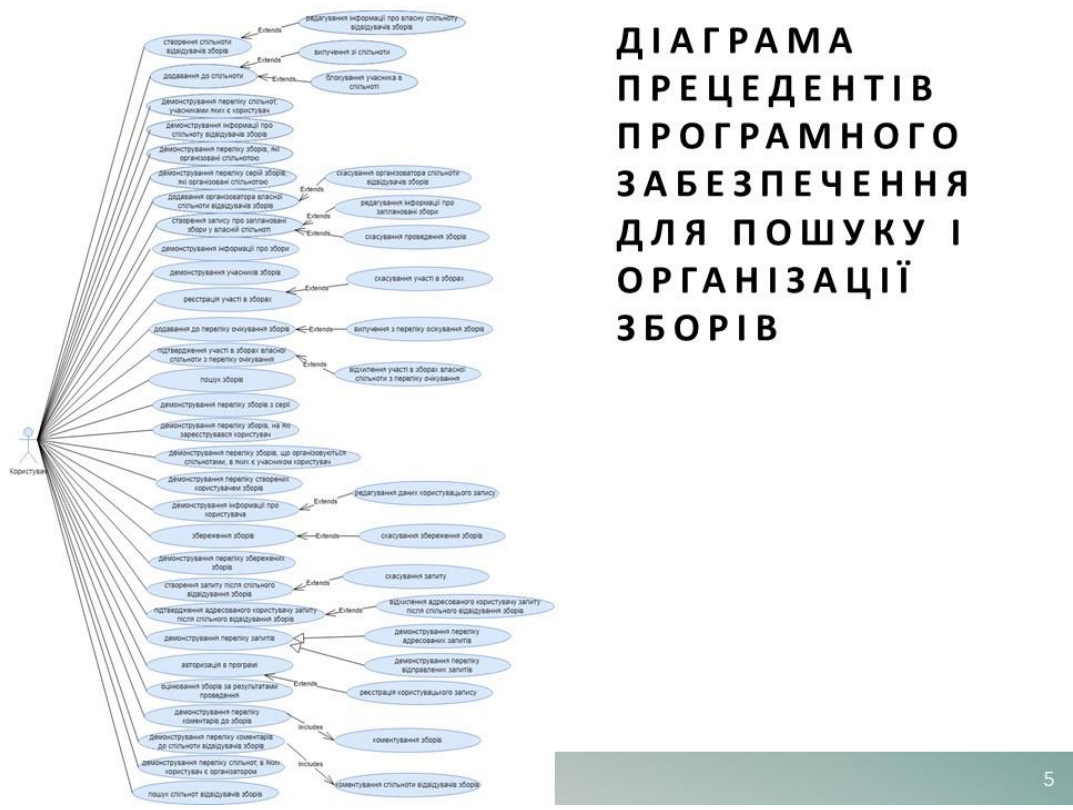


Рисунок В.5 – Діаграма прецедентів програмного забезпечення для пошуку і організації зборів

**СХЕМА БАЗИ ДАНИХ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПОШУКУ І ОРГАНІЗАЦІЇ ЗБОРІВ**

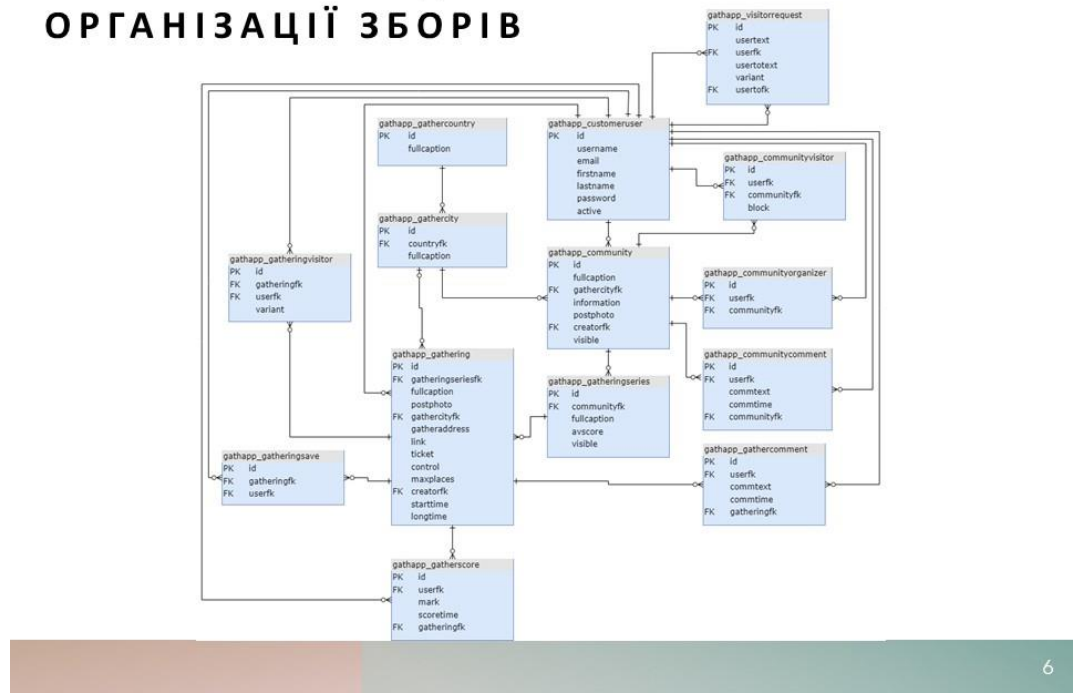


Рисунок В.6 – Схема бази даних програмного забезпечення для пошуку і організації зборів

СХЕМА ФУНКЦІОНУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПОШУКУ І ОРГАНІЗАЦІЇ ЗБОРІВ

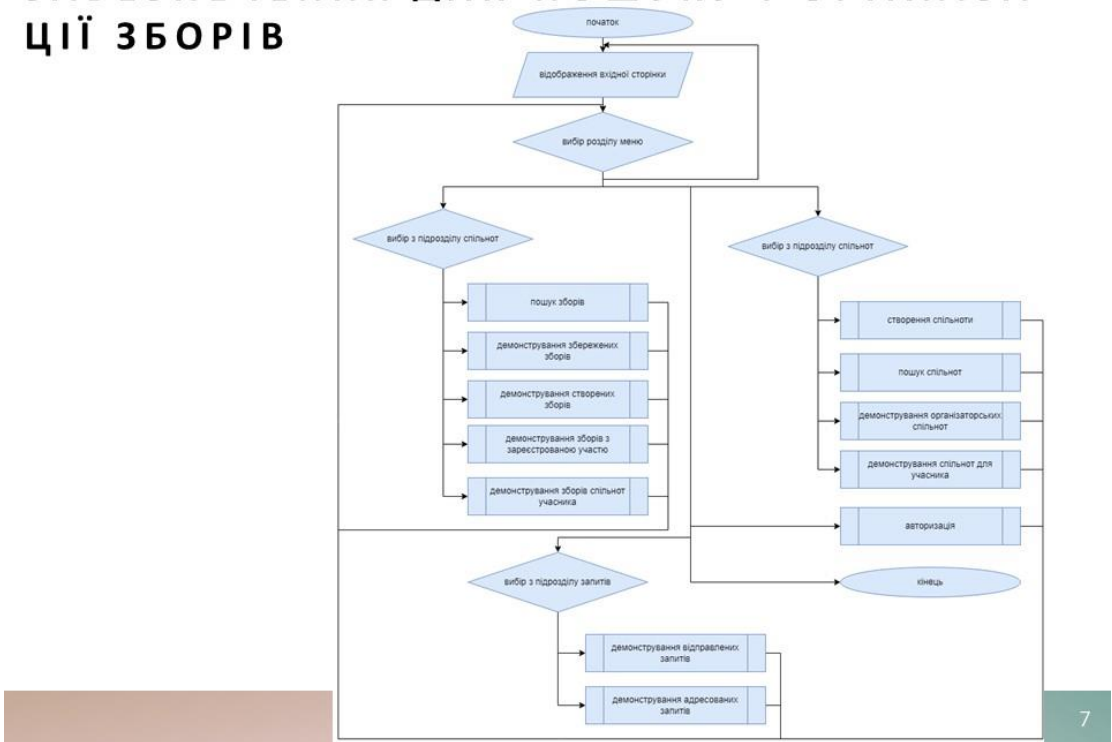


Рисунок В.7 – Схема функціонування програмного забезпечення для пошуку і організації зборів

ДЕМОНСТРУВАННЯ СПІЛЬНОТ ТА ЗБОРІВ

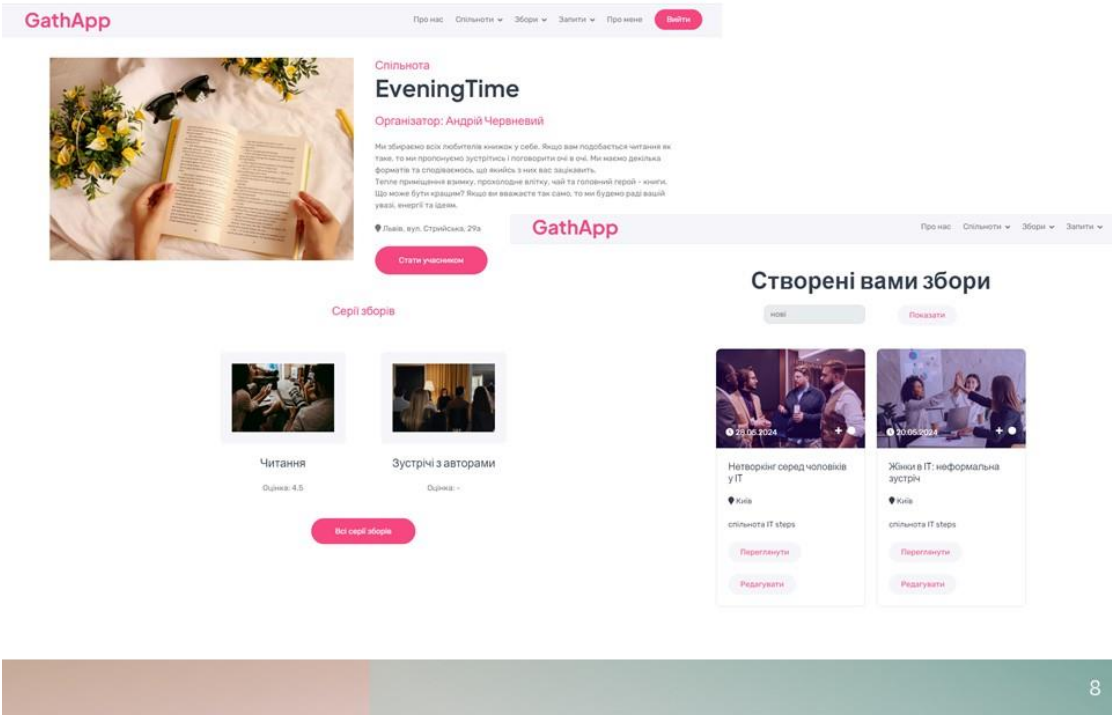
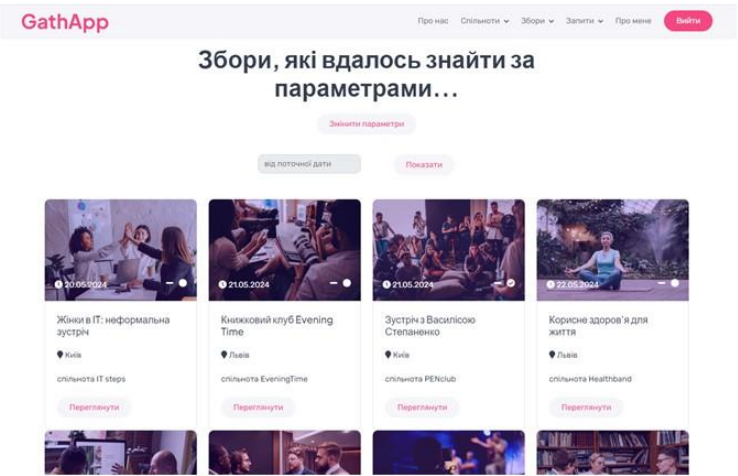


Рисунок В.8 – Демонстрування спільнот та зборів

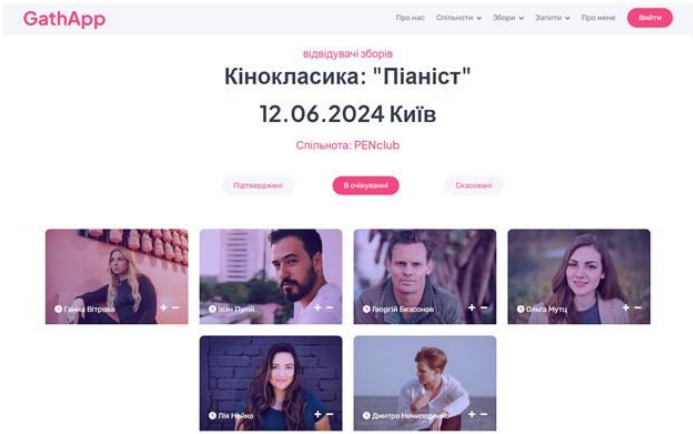
ПОШУК ЗБОРІВ



9

Рисунок В.9 – Пошук зборів

КЕРУВАННЯ ЧЕРГОЮ ОЧІКУВАННЯ  
ВІДВІДУВАЧІВ ЗБОРІВ



10

Рисунок В.10 – Керування чергою очікування відвідувачів зборів

## ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПОШУКУ І ОРГАНІЗАЦІЇ ЗБОРІВ

| Функціональна можливість                                            | Результат | Функціональна можливість                                                                      | Результат |
|---------------------------------------------------------------------|-----------|-----------------------------------------------------------------------------------------------|-----------|
| Створення спільноти відвідувачів зборів                             | Успіх     | Реєстрація участі в зборах                                                                    | Успіх     |
| Редагування інформації про власну спільноту відвідувачів зборів     | Успіх     | Скасування участі в зборах                                                                    | Успіх     |
| Додавання учасника до спільноти відвідувачів зборів                 | Успіх     | Додавання до переліку очікування зборів                                                       | Успіх     |
| Вилучення учасника зі спільноти відвідувачів зборів                 | Успіх     | Вилучення з переліку очікування зборів                                                        | Успіх     |
| Блокування учасника в спільноті відвідувачів зборів                 | Успіх     | Підтвердження участі в зборах власної спільноти з переліку очікування                         | Успіх     |
| Демонстрування переліку спільнот, учасниками яких є користувач      | Успіх     | Відхилення участі в зборах власної спільноти з переліку очікування                            | Успіх     |
| Демонстрування інформації про спільноту відвідувачів зборів         | Успіх     | Пошук зборів                                                                                  | Успіх     |
| Демонстрування переліку зборів, які організовані спільнотою         | Успіх     | Демонстрування переліку зборів з серії                                                        | Успіх     |
| Демонстрування переліку серій зборів, які організовані спільнотою   | Успіх     | Демонстрування переліку зборів, на які зареєструвався користувач                              | Успіх     |
| Додавання організатора власної спільноти відвідувачів зборів        | Успіх     | Демонстрування переліку зборів, що організовуються спільнотами, в яких є учасником користувач | Успіх     |
| Скасування організатора спільноти відвідувачів зборів               | Успіх     | Демонстрування переліку створених користувачем зборів                                         | Успіх     |
| Створення серії зборів власної спільноти                            | Успіх     | Демонстрування інформації про користувача                                                     | Успіх     |
| Редагування серії зборів                                            | Успіх     | Редагування даних користувацького запису                                                      | Успіх     |
| Створення запису про заплановані збори у власній спільноті          | Успіх     | Збереження зборів                                                                             | Успіх     |
| Редагування інформації про заплановані збори                        | Успіх     | Демонстрування переліку збережених зборів                                                     | Успіх     |
| Скасування проведення зборів                                        | Успіх     | Створення запиту після спільного відвідування зборів                                          | Успіх     |
| Демонстрування інформації про збори                                 | Успіх     | Скасування запиту після спільного відвідування зборів                                         | Успіх     |
| Авторизація в програмі                                              | Успіх     | Підтвердження адресованого користувачу запиту після спільного відвідування зборів             | Успіх     |
| Реєстрація користувацького запису                                   | Успіх     | Відхилення адресованого користувачу запиту після спільного відвідування зборів                | Успіх     |
| Оцінювання зборів за результатами проведення                        | Успіх     | Демонстрування переліку адресованих запитів                                                   | Успіх     |
| Демонстрування переліку коментарів до зборів                        | Успіх     | Демонстрування переліку відправлених запитів                                                  | Успіх     |
| Демонстрування переліку коментарів до спільноти відвідувачів зборів | Успіх     |                                                                                               |           |
| Демонстрування переліку спільнот, в яких користувач є організатором | Успіх     |                                                                                               |           |
| Пошук спільнот відвідувачів зборів                                  | Успіх     |                                                                                               |           |
| Демонстрування учасників зборів                                     | Успіх     |                                                                                               |           |

11

Рисунок В.11 – Тестування програмного забезпечення для пошуку і організації зборів

## ВИСНОВКИ

- Програму реалізовано повністю стосовно тих сценаріїв використання, які були визначені разом з альтернативними сценаріями та представлені на діаграмі прецедентів. Виконано відповідне тестування, що дозволило перевірити реалізацію цих здатностей.
- Розроблена програма пошуку і організації зборів надає можливості створення спільнот відвідувачів зборів, створення серій зборів та планування окремих зборів, додавання учасників до спільнот та реєстрації для проведення зборів, у тому числі створюючи черги очікування за обмеження кількості учасників та необхідності підтвердження участі, оцінювання зборів, що дозволяє визначати оцінки для серій зборів, демонстрування зборів, на які зареєструвався користувач, в яких він є організатором, створення персональних запитів після спільного відвідування зборів, коментування зборів і спільнот, організовуючи таким чином спілкування.
- У роботі досягнуто мету, яка полягала в тому, щоб розробити програмне забезпечення пошуку і організації зборів для сприяння живій або дистанційній комунікації у спільнотах.

12

Рисунок В.12 – Висновки