

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 ЗНАХОДЖЕННЯ

(назва теми)

ОДНОДУМЦІВ ДЛЯ ПІШОГО ТУРИЗМУ. SOFTWARE TO
SEARCH FOR LIKE-MINDED PEOPLE FOR HIKING

Виконав(ла): студент(ка) 4 курсу, групи КНТ-220

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

ДУБИНА О.Ю.

(ПРИЗВИЩЕ та ініціали)

Керівник СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ та ініціали)

Рецензент ГОЛУБ Т.В.

(ПРИЗВИЩЕ та ініціали)

2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет _____ КНТ _____
Кафедра _____ програмних засобів _____
Ступінь вищої освіти _____ бакалавр _____
Спеціальність _____ 122 Комп'ютерні науки _____
(код і найменування)
Освітня програма (спеціалізація) _____ Комп'ютерні науки _____
(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)

ДУБИНИ Олега Юрійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення знаходження однодумців для пішого туризму. Software to Search for Like-Minded People for Hiking

керівник проєкту (роботи) к.т.н., доцент СКРУПСЬКИЙ Степан Юрійович,
(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. Розробка архітектури програми. 3. Реалізація 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	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	СКРУПСЬКИЙ С.Ю., 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)

Олег ДУБИНА
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

Степан СКРУПСЬКИЙ
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 89 с., 23 рис., 2 табл., 3 дод., 8 джерел.

ПІШИЙ ТУРИЗМ, ПОХІД, ГІРСЬКА ВЕРШИНА, МАРШРУТ, ВЕБЗАСТОСУНОК, ВЕБФРЕЙМВОРК.

Об'єкт роботи – процес планування пішого походу.

Предмет роботи – програмне забезпечення знаходження однодумців для пішого туризму.

Метою роботи є розробка програмного забезпечення знаходження однодумців для пішого туризму, щоби сприяти підвищенню взаємодії в цій галузі та розвитку галузі загалом.

Матеріали, методи та технічні засоби: мови Python, JavaScript, система MySQL, фреймворк Django.

Результати. Результатами виконання роботи є результати виконання кожного окремого завдання, які сукупно виражені в програмі знаходження однодумців для пішого туризму, що охоплює не тільки можливості встановлення контактів між користувачами, але і визначення власних планів походів, щоби створити підґрунтя щодо взаємодії стосовно конкретних спільних походів, створення екосистеми пішого туризму, в якій визначено регіони пішого туризму, гірські вершини, маршрути.

Висновки. Програма у підсумку призначена для знаходження однодумців для пішого туризму, визначення власних вражень від відвідування гірських вершин, регіонів пішого туризму, взаємодії з іншими користувачами, визначення спільних походів, визначення планів та маршрутів власних походів пішого туризму.

Галузь використання. Організація пішого туризму в групах, індивідуальна зацікавленість.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 89 pages, 23 figures, 2 tables, 3 appendixes, 8 sources.

HIKING TOURISM, HIKING TRIP, MOUNTAIN TOP, ROUTE, WEB APPLICATION, WEBFRAMEWORK.

The object of the work is the process of planning a hiking trip.

The subject of the work is software for finding like-minded people for hiking.

The goal of the work is the development of software for finding like-minded people for hiking tourism in order to promote increased interaction in this field and the evolution of the field in general.

Materials, methods and technical means: Python languages, JavaScript, MySQL system, Django framework.

The results. The results of the work are the results of the individual task execution, which are collectively expressed in the program of finding like-minded people for hiking tourism, which includes not only the possibility of establishing contacts between users, but also the definition of own hiking plans in order to create a basis for interaction regarding specific joint hiking, creating an ecosystem of hiking tourism, which defines hiking tourism regions, mountain peaks and routes.

Conclusions. In the end, the program is intended for finding like-minded people for hiking, determining own impressions of visiting mountain peaks, hiking regions, interacting with other users, identifying joint hikes, determining plans and routes of own hiking trips.

Field of use. Organization of hiking tourism in groups, individual interest.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	9
1.1 Аналіз застосунків для знаходження однодумців для пішого туризму ..	9
1.2 Порівняльний аналіз розробки та аналогів	13
1.3 Висновки за розділом 1	14
2 Розробка архітектури програми.....	15
2.1 Вибір засобів проєктування	15
2.2 Сценарії використання програми	16
2.3 Структура бази даних	20
2.4 Висновки за розділом 2	35
3 Реалізація програми	36
3.1 Визначення порядку роботи програми	36
3.2 Результати реалізації програмних одиниць застосунку.....	37
3.3 Висновки за розділом 3	51
4 Експлуатація та тестування програми	52
4.1 Призначення програми	52
4.2 Умови виконання програми	52
4.3 Виконання і тестування програми.....	52
4.4 Висновки за розділом 4	56
Висновки	57
Перелік джерел посилення	58
Додаток А Технічне завдання	59
Додаток Б Текст програми	65
Додаток В Слайди презентації.....	83

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

БД – база даних;

ВФ – вебфреймворк.

ВСТУП

Піший туризм є достатньо привабливою галуззю, яка можливо не досить є медіаактивною в нашій країні, але має значно більшу привабливість в інших країнах, визнану виробничою підтримкою. Це стосується виготовлення спеціальних речей, одягу, взуття для пішого туризму. На прикладі того, що виготовленню, продажу таких речей приділяється багато уваги, можна говорити про поширеність загалом цієї галузі.

Проте підтримка пішого туризму здійснюється на достатньо різних рівнях:

- через інтернет-магазини для продажу обладнання, одягу, взуття для пішого туризму;
- через відеоролики, які створюють блогери;
- через визначення маршрутів пішого туризму на окремих вебсайтах з можливостями трекінгу.

Проте не приділяється увага реалізації можливості пошуку однодумців для пішого туризму. Це стосується випадків, коли люди готові піти в похід, але шукають цікаву, підходящу компанію для цього. Звичайно, що можливо не кожен шукає саме таких друзів, а йде в похід з друзями з реального життя, яких досить давно знає, проте встановлення контактів між любителями пішого туризму може сприяти тривалості контактів, встановленню таких контактів. Деякі люди можуть бути готовими піти у похід разом, щоб обмінятися досвідом, щоб знайти людей, які відповідають їхньому рівню, зокрема фізичному рівню. Загалом же і популярність соціальних мереж сама по собі підкреслює все ж потенційну можливість для щонайменше деяких людей знайти собі таких однодумців. Саме на таких можливостях має бути зроблено акцент у цій розробці.

Метою роботи є розробка програмного забезпечення знаходження однодумців для пішого туризму, щоби сприяти підвищенню взаємодії в цій галузі та розвитку галузі загалом.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз застосунків для знаходження однодумців для пішого туризму

Під час аналізу було знайдено цілий ряд застосунків, які так чи інакше стосуються сфери знаходження однодумців для пішого туризму, при цьому не вирішуючи проблему в повній мірі.

Існує цілий ряд програмних застосунків, які представляють інформацію про піші походи в певному регіоні, зазвичай містять загальну інформацію про можливі маршрути, іноді включаючи інформацію про наявність туристичного клубу. В якості прикладу такого програмного застосунку винесено вебсайт [1]. Такі рішення не дозволяють через них знайти однодумців щодо пішого туризму, планувати походу, але містять інформацію про піший туризм, про туристичний клуб. Однак саме встановлення контактів вже приховано за отриманням права бути членом туристичного клубу.

Zracks (рис. 1.1) [2] виділений як вебзастосунок, що є прикладом певного занурення в область пішого туризму, містять при цьому багато різних засобів і функціональних можливостей. Зокрема у програмі можна переглядати перелік походів, до яких можна долучитись. Знову ж одразу треба визначити, що це авторські походи, відповідно це фактично можливість взяти участь у формуванні певної групи людей, яка може розширюватись до заданих обсягів. За походом можна переглянути його детальний опис, проте записатися на нього неможливо. Детальний опис є текстовим, на мапі маршрут конкретний не відображається. Але окрім того застосунок дозволяє шукати потрібне спорядження для пішого туризму, купляти таке спорядження, тобто функціонує як інтернет-магазин. Також наявна велика кількість статей щодо спорядження, пішого туризму.

Social Hikers [3] надає можливості брати участь у певних змаганнях, які представляють собою походи в певних регіонах. У програмі можна взяти

участь у такому змаганні, представити опис свого походу, включаючи зокрема і фотографії. Проте маршрут на мапі також не представлений (рис. 1.2).

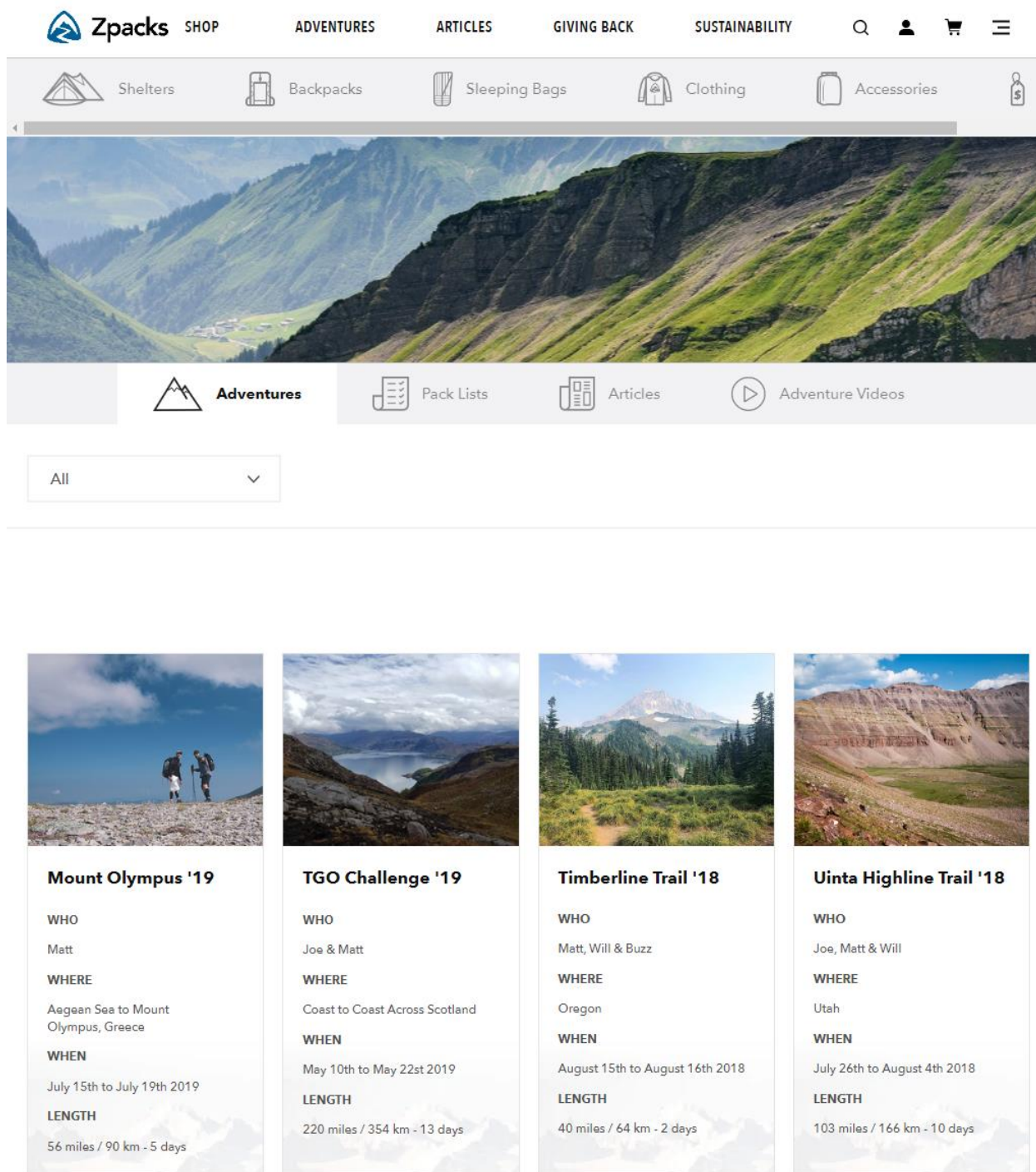


Рисунок 1.1 – Вебзастосунок Zpacks

Походи, виконані конкретними користувачами, можна коментувати. Користувачам можна писати, ставати друзями з ними. Тобто цей застосунок надає певні засоби комунікації, таким чином дозволяє знаходити однодумців

для пішого туризму, хоча і процес планування має вміщуватися у чітко визначені схеми походів, якими є ці змагання.

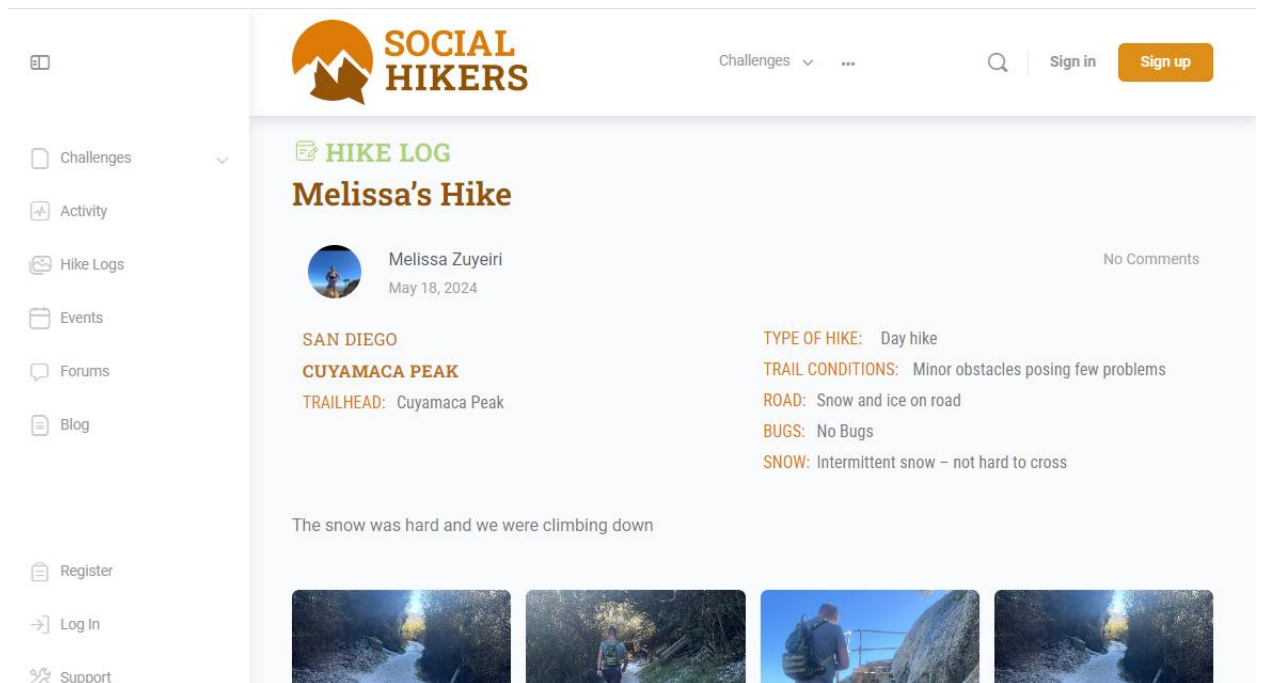


Рисунок 1.2 – Вебзастосунок Social Hikers

Загалом виходячи з рекомендацій, приведених у цих вебзастосунках зокрема, можна зробити висновок, що фактично любителі пішого туризму використовуються Meetup [4] (рис. 1.3) для пошуку однодумців для пішого туризму.

Однак Meetup направлений не саме на туристичну галузь, не на особливості пішого туризму. Це застосунок, який може використовуватися для пошуку будь-яких подій. Відповідно пошук саме походів, маршрутів походів ускладнений. Маршрути виражаються текстом (рис. 1.4), представлення на мапі саме маршруту (не початкової точки) немає. Відповідно деталі маршруту фактично в підсумку не можна зрозуміти, тільки загальну канву. Додаткової інформації про гірські вершини, регіони пішого туризму тощо немає. Відповідно доступний загальний пошук, який не враховує розташування вершин у певних областях або іншої специфічної інформації.

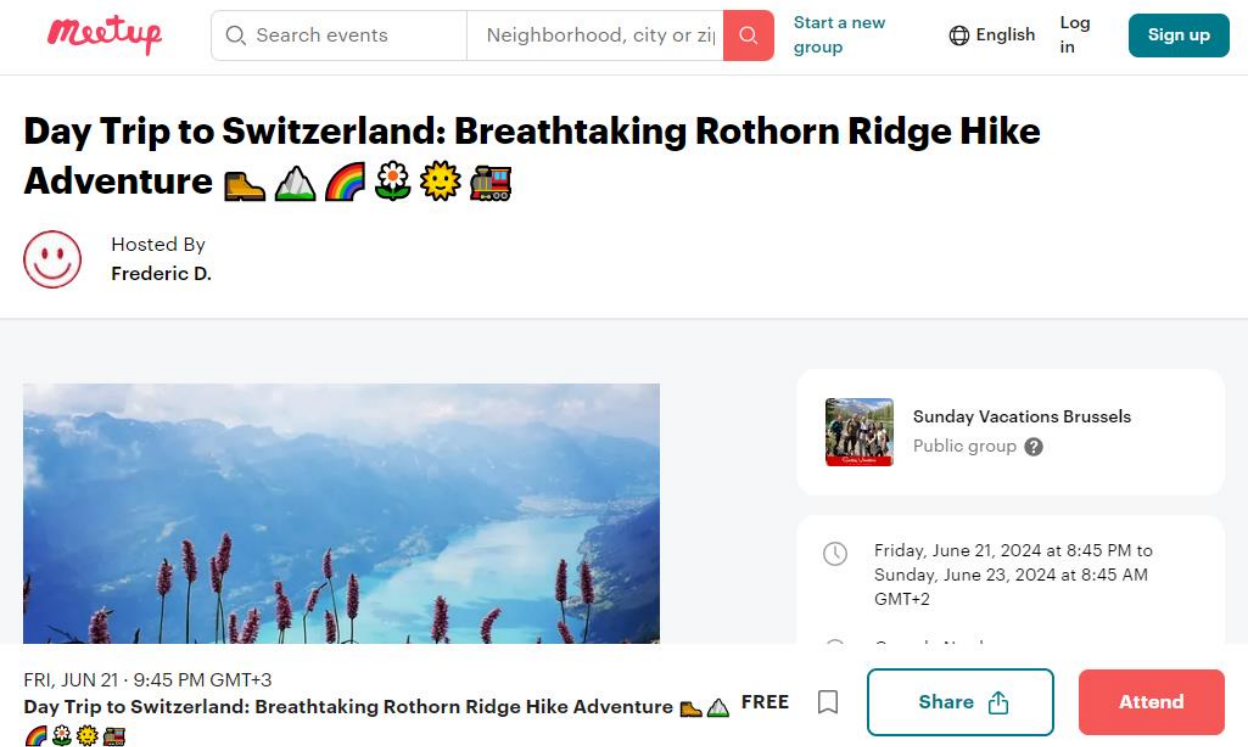


Рисунок 1.3 – Подія походу у вебзастосунку Meetup

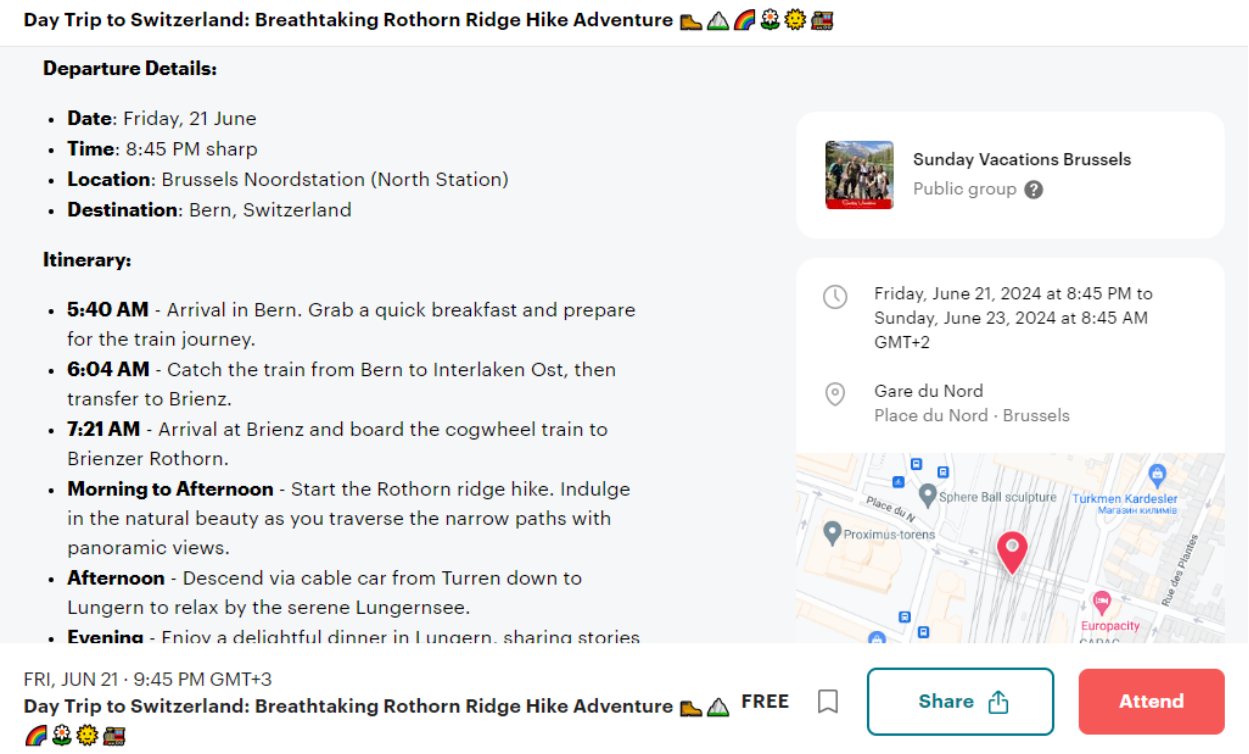


Рисунок 1.4 – Маршрут походу у вебзастосунку Meetup

1.2 Порівняльний аналіз розробки та аналогів

До фінального порівняння були внесені застосунки Meetup та Social Hikers, які були порівняні з розробкою (табл. 1.1).

Таблиця 1.1 – Порівняння програм для знаходження однодумців для пішого туризму

Критерій	Meetup	Social Hikers	Розробка
Визначення походів	Як окрема подія з можливим текстовим описом маршруту	За декількома визначеними у програмі маршрутами	За маршрутами, які створюють самі користувачі
Відображення маршруту на мапі	Немає	Немає	Є
Описи минулих походів	Ні	Так	Так
Можливість приєднатися до нового походу	Так	Ні	Так
Можливість комунікувати з любителями пішого туризму	Так	Так	Так
Висловлення вражень	Про подію (похід)	Про похід	Про похід, про вершини, про регіони

Порівняння продемонструвало відсутність визначення конкретного маршруту походу на мапі у аналогів, неможливість повноцінного власного

визначення походу, бо вони або створюються за обмеженим числом шаблонів, або не є походами початково, а є подіями. Окрім того описи минулих походів можливі тільки в Social Hikers, де тим не менш не можна приєднатися до нового походу іншим користувачам. Також висловлення вражень досить обмежено в аналогах.

1.3 Висновки за розділом 1

Завдання роботи:

- проаналізувати програмне забезпечення, придатне для знаходження однодумців для пішого туризму;
- визначити засоби проєктування програми;
- виділити вимоги до програми знаходження однодумців для пішого туризму;
- виконати реалізацію програми знаходження однодумців для пішого туризму.

Виконання завдання аналізу програмного забезпечення, придатного для знаходження однодумців для пішого туризму, дозволило виявити, що подібні рішення або підтримують організацію однієї конкретної туристичної групи, або мають функціональність, направлену на пошук учасників подій в широкому сенсі, не визначаючи ті складові пішого туризму, які є важливими в походах. На ліквідацію цих недоліків стосовно саме знаходження однодумців для пішого туризму було направлено розробку, що має бути виконана в наступних розділах.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Вибір засобів проєктування

Вебзастосунок доцільно створювати на основі вебфреймворку (ВФ) в сучасних умовах, таким чином маючи певний каркас для розробки. Враховуючи гнучкість і простоту мови Python, вирішено було розробляти програму саме цією мовою, а серед ВФ для неї розглянути Django [5] і Pyramid [6]-[7]. Критерії порівняння (табл. 2.1) стосуються загального підходу до створення застосунку, до роботи з базою даних (БД), до створення інтерфейсу користувача.

Таблиця 2.1 – Вибір засобів для розробки програми знаходження однодумців для пішого туризму

Критерій	Pyramid	Django
Підхід до створення застосунку	Мінімальні базові засоби з можливістю розширення за необхідності	Всі наявні засоби наявні за замовчуванням, частина далі може налаштовуватись
Підхід до роботи з БД	Об'єктно-реляційне відображення необхідно встановлювати окремо	Об'єктно-реляційне відображення доступне за замовчуванням, систему керування БД можна обрати
Підхід до створення інтерфейсу користувача	На основі рушія шаблонів, але рушій необхідно обирати і встановлювати	На основі рушія шаблонів, доступного за замовчуванням, але його можна змінити за потреби

Враховуючи специфіку програми, що має розроблятися, яка має надавати інформацію і про регіони піших походів, і про гірські вершини, і про

маршрути походів, то звичайно, що великою є роль БД. При використанні Django можна мати можливості для створення коду, який дозволить згенерувати структуру таблиць бази даних, об'єктно-реляційне відображення роботи з даними за замовчуванням. Так само і зі створенням інтерфейсу, наявністю адміністративної панелі, системи авторизації. Усе це сукупно призводить до того, що у ВФ Django можна мати потужні, випробувані на практиці інструменти одразу після створення застосунку. Тож час, витрачений на розробку, менший. Уникнути потреби в таких інструментах, враховуючи необхідні функціональні можливості, щоб отримати перевагу в меншій кількості початково доступних засобів, неможливо. Тому обрано ВФ Django.

2.2 Сценарії використання програми

Користувачі не розділялись за ролями, щоби кожен з них міг використовувати весь спектр функцій, сприяючи поширенню використання різних функцій, а також представленню кожного користувача з різних боків. Основний потік сценаріїв:

- авторизація любителя пішого туризму;
- збереження враження від взаємодії з любителем пішого туризму;
- перегляд власних вражень та вражень заданого користувача, включаючи враження від взаємодії з любителями пішого туризму, враження від відвідування регіону пішого туризму, враження від відвідування гірської вершини;
- перегляд вражень різних користувачів від взаємодії з заданим любителем пішого туризму;
- перегляд сторінки регіону пішого туризму;
- перегляд сторінки гірської вершини;
- перегляд вражень від відвідувань заданого регіону пішого туризму;
- перегляд вражень від відвідувань заданої гірської вершини;
- перегляд маршрутів походів за заданою гірською вершиною;

- перегляд профілю любителя пішого туризму;
- перегляд плану походу пішого туризму;
- редагування профілю любителя пішого туризму;
- створення плану походу, включаючи визначення розподілу за днями;
- перегляд маршрутів походів за заданим регіоном пішого туризму;
- перегляд планів майбутніх походів любителя пішого туризму;
- перегляд планів завершених походів любителя пішого туризму;
- збереження враження від відвідування регіону пішого туризму користувачем;
- збереження враження від відвідування гірської вершини користувачем;
- створення маршруту походу пішого туризму;
- перегляд маршруту походу пішого туризму, включаючи перегляд коментарів;
- перегляд маршрутів походів любителя пішого туризму;
- перегляд рейтингу маршрутів походів пішого туризму загалом;
- перегляд рейтингу маршрутів походів пішого туризму за регіоном;
- перегляд рейтингу любителів піших походів;
- перегляд рейтингу гірських вершин;
- перегляд рейтингу регіонів пішого туризму;
- збереження маршруту походу пішого туризму;
- перегляд збережених маршрутів піших походів;
- створення пропозиції про спільний похід;
- підтвердження пропозиції про спільний похід;
- перегляд пропозицій про спільний похід, які адресовані користувачу;
- перегляд пропозицій про спільний похід, які створені користувачем;
- перегляд планів спільних походів пішого туризму, у тому числі завершених та запланованих;
- коментування маршруту походу пішого туризму;
- перегляд переліку маршрутів походів пішого туризму, які

прокоментував користувач;

- визначення симпатії маршруту походів пішого туризму;
- перегляд власних симпатій до маршрутів походів пішого туризму;
- відправлення персонального повідомлення;
- перегляд створених спілкувань з користувачами;
- перегляд персональних повідомлень за спілкуванням;
- визначення власного одностороннього;
- перегляд планів подорожей власних односторонніх;
- перегляд переліку власних односторонніх;
- універсальний пошук, який повинен охоплювати пошук гірських вершин, регіонів пішого туризму, планів походів, маршрутів походів, дозволяючи виконувати вибір серед цих варіантів як вибір значення параметра пошуку.

Альтернативні сценарії:

- реєстрація любителів пішого туризму як користувачів у програмі до сценарію авторизації любителів пішого туризму, коли любитель пішого туризму не є користувачем програми і не має облікового запису;
- редагування плану походу до сценарію створення плану походу, коли план поход був створений цим же користувачем, проте він бажає змінити його певним чином;
- редагування маршруту походу пішого туризму до сценарію створення маршруту походу пішого туризму, коли маршрут походу пішого туризму був створений цим же користувачем, проте він бажає змінити його певним чином;
- вилучення маршруту походу пішого туризму зі збережених до сценарію збереження маршруту походу пішого туризму, коли маршрут походу пішого туризму тим же самим користувачем вже був збережений, але більше користувач не має наміру бачити його серед своїх збережених маршрутів походів пішого туризму (рис. 2.1).

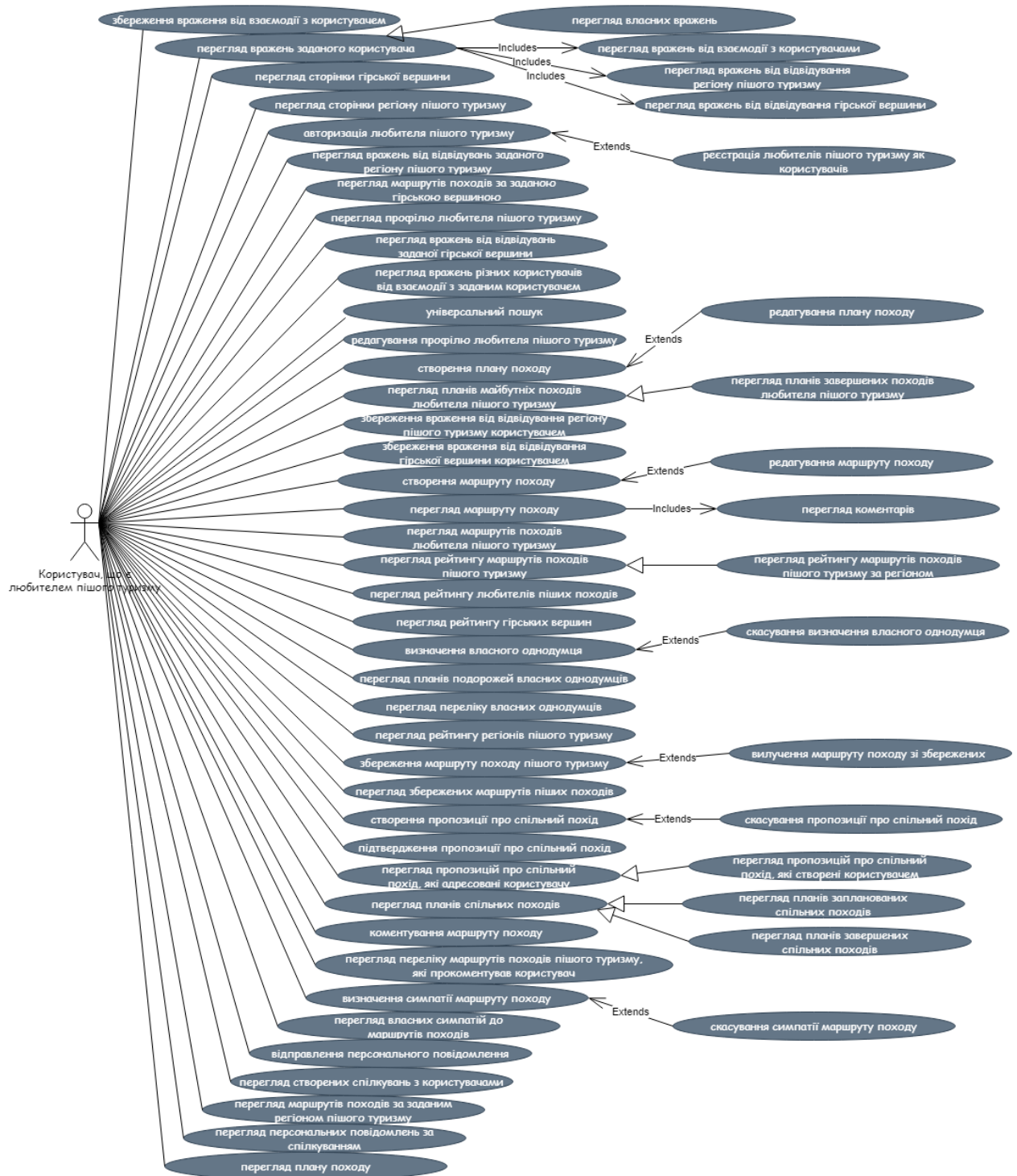


Рисунок 2.1 – Діаграма прецедентів програми знаходження однодумців для пішого туризму

Інші альтернативні сценарії:

– скасування пропозиції про спільний похід до сценарію створення пропозиції про спільний похід, коли користувач створив пропозицію про спільний похід, але сам же вирішив, що цю пропозицію більше не підтримує,

або адресат цієї пропозиції не має бажання йти у похід з цим любителем пішого туризму;

- скасування симпатії маршруту походів пішого туризму до сценарію визначення симпатії маршруту походів пішого туризму, коли користувач визначив симпатію маршруту походу пішого туризму, але більше ця симпатія не відповідає дійсності;

- скасування визначення власного одностороннього до сценарію визначення власного одностороннього, коли користувач визначив іншого своїм одностороннім, але більше це не відповідає дійсності і переглядати нові плани походів цього користувача він більше не має бажання

2.3 Структура бази даних

MySQL використано як основу створення БД, що підтримує цей робочий процес, забезпечуючи високий рівень надійності, характерний для цієї системи, та при цьому безкоштовність. Сама структура презентована нижче.

Таблиця БД `hiker` визначає користувача, який є любителем пішого туризму, за стовпцями:

- `id` – ідентифікатор користувача, який є любителем пішого туризму;
- `useraccount` – обліковий запис, за яким працює користувач, який є любителем пішого туризму, у програмі;
- `hikename` – прізвище, за яким відомий цей любитель пішого туризму серед колег;
- `fullname` – повне ім'я цього любителя пішого туризму;
- `hikeexperience` – рік, з якого займається пішим туризмом цей користувач;
- `birthdate` – дата народження любителя пішого туризму;
- `countryregion` – область країни, в якій живе користувач, який є любителем пішого туризму;

- hikerdescription – презентаційний опис любителя пішого туризму;
- impression – середня оцінка, яку за враженнями від взаємодії з іншими користувачами було виставлено цьому любителю пішого туризму.

Поле countryregion є зовнішнім ключем до таблиці БД Countryregion, таким чином закріплюючи те, що кожен любитель пішого туризму проживає в одній області країни, при цьому одна область країни може мати від нуля і більше любителів пішого туризму (тип один-до-багатьох).

Враження від взаємодії з користувачем, який є любителем пішого туризму, визначені таблицею Hikerimpression зі стовпцями:

- id – ідентифікатор враження від взаємодії з користувачем, який є любителем пішого туризму;
- hiker – користувач, який є любителем пішого туризму та враження про якого визначено;
- hikerimpressed – користувач, який є любителем пішого туризму та який отримав і зберіг це враження;
- point – числова оцінка за даним враженням, виставлена користувачу, який є любителем пішого туризму;
- impressiontext – текстом виставлене враження про користувача, який є любителем пішого туризму;
- impressiondate – дата, коли було збережено враження про любителя пішого туризму;
- commonplan – визначення того, чи було це враження про любителя пішого туризму, підтверджено спільним планом участі в поході, чи ні.

Поле hiker є зовнішнім ключем до таблиці БД Hiker, таким чином закріплюючи те, що кожне враження є враженням тільки про одного любителя пішого туризму, при цьому про одного любителя пішого туризму різні користувачі можуть створити від нуля і більше вражень (тип один-до-багатьох).

Поле hikerimpressed є зовнішнім ключем до таблиці БД Hiker, таким чином закріплюючи те, що кожне враження є враженням тільки одного

любителя пішого туризму, при цьому один любитель пішого туризму може створити від нуля і більше вражень (тип один-до-багатьох).

Зображення любителя пішого туризму визначені таблицею `Hikerimage` зі стовпцями:

- `id` – ідентифікатор зображення любителя пішого туризму;
- `hiker` – любитель пішого туризму, який представлений на цьому зображенні;
- `image` – вміст зображення, на якому представлений цей любитель пішого туризму;
- `mainphoto` – це зображення є основним, тобто застосовується для представлення самого любителя пішого туризму там, де потрібне тільки одне зображення, або ні.

Поле `hiker` є зовнішнім ключем до таблиці БД `Hiker`, таким чином закріплюючи те, що кожне зображення представляє тільки одного любителя пішого туризму, при цьому до одного любителя пішого туризму може зберігатися від нуля і більше зображень (тип один-до-багатьох).

У програмі любителі пішого туризму можуть і ділитися колишнім досвідом, і визначати майбутні плани, щоб за ними шукати однодумців, але структура обох цих сутностей фактично однакова, відрізняється тільки час (минулий або майбутній), визначений відповідними датами, а також те, чи це описується досвід, чи плани. Досвід може бути підкріплений фотографіями. Але фотографій може бути декілька до одного опису походу, тому вони мають бути винесені окремо в таблицю. Тому виходить, що структурно різниці між цими сутностями в межах такої таблиці БД немає. Тому створена одна спільна таблиця `Hikingplan` зі стовпцями:

- `id` – ідентифікатор плану походу пішого туризму, яка вже відбулась або планується;
- `hiker` – користувач, який є любителем пішого туризму та який створив цей похід пішого туризму, яка вже відбулась або планується;

- `hikingstart` – дата, коли має розпочатися похід пішого туризму або коли вона вже фактично розпочалась, якщо описується історія, за цим планом;
- `hikingend` – дата, коли має закінчитися похід пішого туризму або коли вона вже фактично закінчилась, якщо описується історія, за цим планом;
- `caption` – загальна назва як підпис плану походу пішого туризму;
- `overalldescription` – детальний загальний опис плану походу пішого туризму;
- `presentimage` – зображення для ілюстрації.

Поле `hiker` є зовнішнім ключем до таблиці БД `Hiker`, таким чином закріплюючи те, що кожен план походу пішого туризму може бути створений тільки одним любителем пішого туризму, при цьому один любитель пішого туризму може створити від нуля і більше планів (тип один-до-багатьох).

Якщо похід пішого туризму вже відбувся за цим планом, то історія визначається зображеннями з походу в таблиці `Hikinghistory` зі стовпцями:

- `id` – ідентифікатор зображення з походу пішого туризму, яка відбулась раніше;
- `hikingplan` – план походу пішого туризму, під час якої було отримано це зображення;
- `image` – вміст зображення, на якому представлений момент з походу пішого туризму, яка вже відбулась.

Поле `hikingplan` є зовнішнім ключем до таблиці БД `Hikingplan`, таким чином закріплюючи те, що кожне зображення представляє тільки один план походу пішого туризму на ньому, при цьому один плану походу пішого туризму може створити від нуля і більше зображень (тип один-до-багатьох).

Регіони пішого туризму визначені в таблиці `Hikingregion` зі стовпцями:

- `id` – ідентифікатор регіону пішого туризму;
- `regionname` – назва регіону пішого туризму;
- `impression` – середня оцінка, яку за враженнями було виставлено цьому регіону пішого туризму любителями пішого туризму, які його відвідували;

– presentimage – базове зображення, на якому показаний цей регіон пішого туризму.

Відношення регіонів пішого туризму до областей країни, де ці регіони знаходяться, визначені в таблиці Hikingcountryregion зі стовпцями:

- id – ідентифікатор відношення регіону пішого туризму до області країни;
- hikingregion – регіон пішого туризму, який розташований в цій області;
- countryregion – область країни, в якій розташований цей регіон пішого туризму.

Таблиця Hikingcountryregion вже є сама по собі реалізацією зв'язку багато-до-багатьох між таблицями Countryregion і Hikingregion. Виникає він тому, що регіон пішого туризму може охоплювати декілька областей країни, наприклад, знаходячись сам по собі на межі областей. Тому може бути така ситуація, що один регіон пішого туризму знаходиться в різних областях країни, а в свою чергу кожна область країни може містити від 0 і більше регіонів пішого туризму.

Враження від регіону пішого туризму визначені в таблиці Hikingregionimpression зі стовпцями:

- id – ідентифікатор враження від регіону пішого туризму любителя пішого туризму;
- hiker – користувач, який є любителем пішого туризму та який зберіг враження про регіон пішого туризму;
- hikingregion – регіон пішого туризму, про який зберіг враження користувач, який є любителем пішого туризму;
- hikingstart – дата початку того походу пішого туризму, на основі якого любитель пішого туризму створив це враження: якщо це було зроблено на основі декількох походів, то може бути обрано один, який презентує основне враження або останній;

– `hikingend` – дата закінчення того походу пішого туризму, на основі якого любитель пішого туризму створив це враження;

– `point` – числова оцінка за даним враженням, виставлена регіону пішого туризму користувачем, який є любителем пішого туризму та побував у цьому регіоні;

– `impressiontext` – текстом висловлене враження користувачем, який є любителем пішого туризму, про регіон пішого туризму;

– `impressiondate` – дата, коли було збережено враження про регіон пішого туризму любителем пішого туризму.

Поле `hiker` є зовнішнім ключем до таблиці БД `Hiker`, таким чином закріплюючи те, що кожне враження про регіон пішого туризму створене тільки одним любителем пішого туризму, при цьому один любитель пішого туризму може створити від нуля і більше вражень (тип один-до-багатьох).

Поле `hikingregion` є зовнішнім ключем до таблиці БД `Hikingregion`, таким чином закріплюючи те, що кожне враження є враженням про один регіон пішого туризму, при цьому про один регіон пішого туризму може бути створено від нуля і більше вражень (тип один-до-багатьох).

Області країн (або певні регіони, агломерації), що підтримуються в програмі, визначені в таблиці `Countryregion` зі стовпцями:

- `id` – ідентифікатор області країни, що підтримується в програмі;
- `country` – країна, в якій знаходиться область;
- `regionname` – назва області.

Таблиця `Countryregion` необхідна для врахування адміністративного поділу відповідної країни.

Гірські вершини визначені в таблиці `Peak` зі стовпцями:

- `id` – ідентифікатор гірської вершини;
- `peakname` – назва гірської вершини;
- `hikingregion` – регіон пішого туризму, де знаходиться ця гірська вершина;

– impression – середня оцінка, яку за враженнями було виставлено цій гірській вершині любителями пішого туризму, які її відвідували;

– presentimage – базове зображення, на якому показана ця гірська вершина.

Поле hikingregion є зовнішнім ключем до таблиці БД Hikingregion, таким чином закріплюючи те, що кожна гірська вершина розташована в одному регіоні пішого туризму, при цьому в одному регіоні пішого туризму може бути багато гірських вершин (тип один-до-багатьох).

Враження від відвідування гірських вершин користувачами, які є любителями пішого туризму, визначені в таблиці Peakimpression зі стовпцями:

– id – ідентифікатор враження від відвідування гірської вершини користувачем, який є любителем пішого туризму;

– hiker – любитель пішого туризму, який зберіг це враження від відвідування цієї гірської вершини;

– peak – гірська вершина, відвідування якої стосується враження любителя пішого туризму;

– hikingstart – дата початку того походу пішого туризму, на основі якого любитель пішого туризму створив це враження: якщо це було зроблено на основі декількох походів, то може бути обрано один, який презентує основне враження або останній;

– hikingend – дата закінчення того походу пішого туризму, на основі якого любитель пішого туризму створив це враження;

– point – числова оцінка за даним враженням, виставлена цій гірській вершині користувачем, який є любителем пішого туризму та побував на цій вершині;

– impressiontext – текстом висловлене враження користувачем, який є любителем пішого туризму, про цю гірську вершину;

– impressiondate – дата, коли було збережено враження про цю гірську вершину любителем пішого туризму.

Поле `peak` є зовнішнім ключем до таблиці БД `Peak`, таким чином закріплюючи те, що кожне враження є враженням про одну гірську вершину, при цьому про одну гірську вершину може бути створено від нуля і більше вражень (тип один-до-багатьох).

Поле `hiker` є зовнішнім ключем до таблиці БД `Hiker`, таким чином закріплюючи те, що кожне враження є враженням, створеним одним любителем пішого туризму, при цьому один любитель пішого туризму може створити від нуля і більше вражень (тип один-до-багатьох).

Дні плану походу пішого туризму визначені таблицею `Hikingday` зі стовпцями:

- `id` – ідентифікатор дня плану походу пішого туризму;
- `hikingplan` – план походу пішого туризму, до якого відноситься цей день;
- `daynumber` – порядковий номер цього дня у цьому плані походу пішого туризму;
- `dayname` – коротка назва, що представляє суть цього дня у цьому плані походу пішого туризму;
- `daydescription` – детальний опис цього дня походу пішого туризму за планом;
- `hikingroute` – маршрут походу, за яким відбувається за планом похід в цей день;
- `presentimage` – зображення для ілюстрації.

Поле `hikingplan` є зовнішнім ключем до таблиці БД `Hikingplan`, таким чином закріплюючи те, що кожен день походу стосується конкретного походу пішого туризму, визначеного його планом, при цьому один план походу пішого туризму може мати від нуля (в момент поки дані не внесені в БД), одного і більше днів (тип один-до-багатьох).

Поле `hikingroute` є зовнішнім ключем до таблиці БД `Hikingroute`, таким чином закріплюючи те, що кожен день походу може мати конкретний маршрут походу, який його деталізовано, при цьому один маршрут походу може

стосуватися від 0 і більше днів планів походів, дозволяючи один і той самий маршрут походу застосовувати повторно іншим любителям пішого туризму, які спиралися на цей маршрут (тип один-до-багатьох).

Маршрут походу визначається таблицею Hikingroute зі стовпцями:

- id – ідентифікатор маршруту походу;
- caption – загальна коротка назва походу як підпис;
- routedescription – деталізований опис маршруту походу;
- authorhiker – любитель пішого туризму, який створив цей маршрут походу;

- impression – середня оцінка, яку за враженнями було виставлено цьому маршруту походу любителями пішого туризму, які його використовували.

Поле authorhiker є зовнішнім ключем до таблиці БД Hiker, таким чином закріплюючи те, що кожен маршрут походу створюється одним любителем пішого туризму, при цьому один любитель пішого туризму може створити від нуля і більше маршрутів походів (тип один-до-багатьох).

Маршрут походу складається з конкретних зупинок або розгалуджень, що знаходяться в таблиці Hikingroutestop зі стовпцями:

- id – ідентифікатор зупинки або розгалудження в маршруті походу;
- hikingroute – маршрут походу, в який входить ця зупинка або розгалудження;
- stoplat – широта позиції зупинки або розгалудження з маршруту походу на мапі;
- stoplong – довгота позиції зупинки або розгалудження з маршруту походу на мапі;
- number – порядковий номер зупинки або розгалудження в маршруті походу.

Поле hikingroute є зовнішнім ключем до таблиці БД Hikingroute, таким чином закріплюючи те, що кожна зупинка або розгалудження на мапі відноситься до одного конкретного маршруту походу, при цьому один

маршрут походу може мати 0 і більше зупинок або розгалуджень, що і визначають сам маршрут походу (тип один-до-багатьох).

Для зручності роботи зупинки або розгалудження маршруту походу зв'язуються з гірськими маршрутами в таблиці `Hikingroutestoppeak` зі стовпцями:

- `id` – ідентифікатор відношення зупинки або розгалудження маршруту походу до гірської вершини;
- `hikingroutestop` – маршрут походу, до якого належить зупинка або розгалудження на мапі;
- `peak` – гірська вершина, на якій знаходиться ця зупинка або розгалудження маршруту походу на мапі.

Збереження маршруту походу користувачем, який є любителем пішого туризму, знаходяться в таблиці `Hikingroutesave` зі стовпцями:

- `id` – ідентифікатор збереження маршруту походу користувачем, який є любителем пішого туризму;
- `hikingroute` – маршрут походу, який збережено користувачем, який є любителем пішого туризму;
- `hiker` – любитель пішого туризму, який зберіг цей маршрут походу.

Поле `hikingroute` є зовнішнім ключем до таблиці БД `Hikingroute`, таким чином закріплюючи те, що кожне збереження любителем пішого туризму відноситься до одного маршруту походу, при цьому один маршрут походу може бути збережений 0 і більше користувачами, які є любителями пішого туризму (тип один-до-багатьох).

Поле `hiker` є зовнішнім ключем до таблиці БД `Hiker`, таким чином закріплюючи те, що кожне збереження виконується одним конкретним любителем пішого туризму, при цьому один любитель пішого туризму може зберегти 0 і більше маршрутів походів (тип один-до-багатьох).

Джерела, які використовував та хоче позначити стосовно власного плану походу пішого туризму користувач, який є любителем пішого туризму, визначаються в таблиці `Hikingplansource` зі стовпцями:

– `id` – ідентифікатор джерела, яке використовував та хоче позначити стосовно власного плану походу пішого туризму користувач, який є любителем пішого туризму;

– `hikingplan` – план походу пішого туризму, при створенні якого використано або який ілюструє це джерело;

– `sourceurl` – посилання на джерело, яке використовував та хоче позначити стосовно власного плану походу пішого туризму користувач, який є любителем пішого туризму.

Поле `hikingplan` є зовнішнім ключем до таблиці БД `Hikingplan`, таким чином закріплюючи те, що кожне джерело визначається в програмі стосовно тільки одного плану походу пішого туризму, при цьому один план походу пішого туризму може включати 0 і більше джерел (тип один-до-багатьох).

Пропозиції до можливих однодумців стосовно спільних походів пішого туризму визначені в таблиці `Hikingplanproposition` зі стовпцями:

– `id` – ідентифікатор пропозиції до можливого однодумця стосовно спільного походу пішого туризму;

– `hikingplan` – план пішого туризму, за яким буде здійснюватися цей пропонуванний спільний похід;

– `hiker` – користувач, який є любителем пішого туризму та якому адресовано цю пропозицію стосовно спільного походу пішого туризму як до однодумця;

– `hikersource` – позначення того, яка зі сторін є ініціатором пропозиції до можливого однодумця стосовно спільного походу пішого туризму: це враховує те, що пропозиція може бути сформована автором плану пішого походу та направлятися іншому любителю пішого туризму, але може бути і так, що любитель пішого туризму сам знайшов цей план, ознайомився з ним та ініціював пропозицію про спільний похід до автора плану походу;

– `proposition` – текст, який доданий до пропозиції до можливого однодумця стосовно спільного походу пішого туризму;

– approve – підтвердження, відхилення або відсутність відповіді стосовно пропозиції до можливого однодумця стосовно спільного походу пішого туризму;

– propositiontime – час створення пропозиції до можливого однодумця стосовно спільного походу пішого туризму.

Поле hikingplan є зовнішнім ключем до таблиці БД Hikingplan, таким чином закріплюючи те, що кожна пропозиція до можливого однодумця стосовно спільного походу пішого туризму може бути визначена тільки за одним планом походу, при цьому один план походу може мати 0 і більше пропозицій до можливого однодумця стосовно спільного походу пішого туризму (тип один-до-багатьох).

Поле hiker є зовнішнім ключем до таблиці БД Hiker, таким чином закріплюючи те, що тільки один користувач, який є любителем пішого туризму, може створити пропозицію до можливого однодумця стосовно спільного походу пішого туризму, при цьому один користувач, який є любителем пішого туризму, може створити 0 і більше пропозицій до можливих однодумців стосовно спільного походу пішого туризму (тип один-до-багатьох).

Симпатії до маршрутів походів пішого туризму визначаються в таблиці Hikingroutelike зі стовпцями:

– id – ідентифікатор симпатії до маршруту походу пішого туризму;

– hikingroute – маршрут походу пішого туризму, за яким визначена симпатія користувачем, який є любителем пішого туризму;

– hiker – любитель пішого туризму, який визначив симпатію до маршруту походу пішого туризму.

Поле hiker є зовнішнім ключем до таблиці БД Hiker, таким чином закріплюючи те, що тільки один користувач, який є любителем пішого туризму, може встановити одночасно симпатію маршруту походу пішого туризму, при цьому один користувач, який є любителем пішого туризму, може

встановити 0 і більше симпатій різним маршрутам походів пішого туризму (тип один-до-багатьох).

Поле `hikingroute` є зовнішнім ключем до таблиці БД `Hikingroute`, таким чином закріплюючи те, що кожна симпатія стосується тільки одного маршруту походу пішого туризму, при цьому один маршрут походу пішого туризму може мати 0 і більше симпатій (тип один-до-багатьох).

Коментарі до маршрутів походів пішого туризму визначені в таблиці `Hikingroutecomment` зі стовпцями:

- `id` – ідентифікатор коментаря до маршруту походу пішого туризму користувача, який є любителем пішого туризму;
- `hikingroute` – маршрут походу пішого туризму, до якого користувач, який є любителем пішого туризму, написав коментар;
- `hiker` – користувач, який є любителем пішого туризму та написав цей коментар до маршруту походу пішого туризму;
- `comment` – текст коментаря до маршруту походу пішого туризму користувача, який написав коментар;
- `commenttime` – час, коли було написано коментар до походу пішого туризму.

Поле `hikingroute` є зовнішнім ключем до таблиці БД `Hikingroute`, таким чином закріплюючи те, що кожен коментар адресується одному маршруту походу пішого туризму, при цьому один маршрут походу пішого туризму може мати 0 і більше коментарів (тип один-до-багатьох).

Поле `hiker` є зовнішнім ключем до таблиці БД `Hiker`, таким чином закріплюючи те, що кожен коментар може бути написаний тільки одним любителем пішого туризму, при цьому один любитель пішого туризму може створити 0 і більше коментарів (тип один-до-багатьох).

Персональне повідомлення між однодумцями з пішого туризму визначено таблицею `Hikermesssage` зі стовпцями:

- `id` – ідентифікатор персонального повідомлення між однодумцями з пішого туризму;

– `hiker` – користувач, який є любителем пішого туризму та який надіслав повідомлення до свого однодумця з пішого туризму;

– `hikermessaged` – користувач, який є любителем пішого туризму та якому було надіслано повідомлення однодумцем з пішого туризму;

– `messagetext` – текст персонального повідомлення між однодумцями з пішого туризму;

– `messagetime` – час відправлення персонального повідомлення між однодумцями з пішого туризму.

Поле `hiker` є зовнішнім ключем до таблиці БД `Hiker`, таким чином закріплюючи те, що кожне персонального повідомлення між однодумцями з пішого туризму може бути надіслане тільки одним користувачем, який є любителем пішого туризму, при цьому один любитель пішого туризму може створити 0 і більше персональних повідомлень (тип один-до-багатьох).

Поле `hiker` є зовнішнім ключем до таблиці БД `Hiker`, таким чином закріплюючи те, що кожне персонального повідомлення між однодумцями з пішого туризму може бути надіслане тільки одному користувачу, який є любителем пішого туризму, при цьому одному любителю пішого туризму може бути надіслано 0 і більше персональних повідомлень (тип один-до-багатьох).

Визначення користувача, який є любителем пішого туризму, власним однодумцем виконано таблицею `Hikerattention` зі стовпцями:

– `id` – ідентифікатор визначення користувача, який є любителем пішого туризму, власним однодумцем;

– `hiker` – користувач, який є любителем пішого туризму і який визначив іншого користувача своїм однодумцем стосовно пішого туризму;

– `hikerattended` – користувач, який є любителем пішого туризму і якого визначив своїм однодумцем стосовно пішого туризму інший користувач (рис. 2.2).

Поле `hiker` є зовнішнім ключем до таблиці БД `Hiker`, таким чином закріплюючи те, що кожне визначення користувача, який є любителем пішого

Рисунок 2.2 – Схема БД програми знаходження однодумців для пішого туризму

2.4 Висновки за розділом 2

Обґрунтовано прийняття рішення, пов'язаного з вибором ВФ Django в якості основи розробки, описано сценарії використання програми знаходження однодумців для пішого туризму з визначенням діаграми прецедентів, представлено структуру БД.

3 РЕАЛІЗАЦІЯ ПРОГРАМИ

3.1 Визначення порядку роботи програми

Функціональна схема програми знаходження однодумців для пішого туризму базується на меню, яке пропонується користувачу для роботи, починаючи з роботи з рейтингом маршрутів походів та подальшим вибором відповідного розділу та пункту меню з поверненням до вибору на кожному наступному кроці (рис. 3.1).

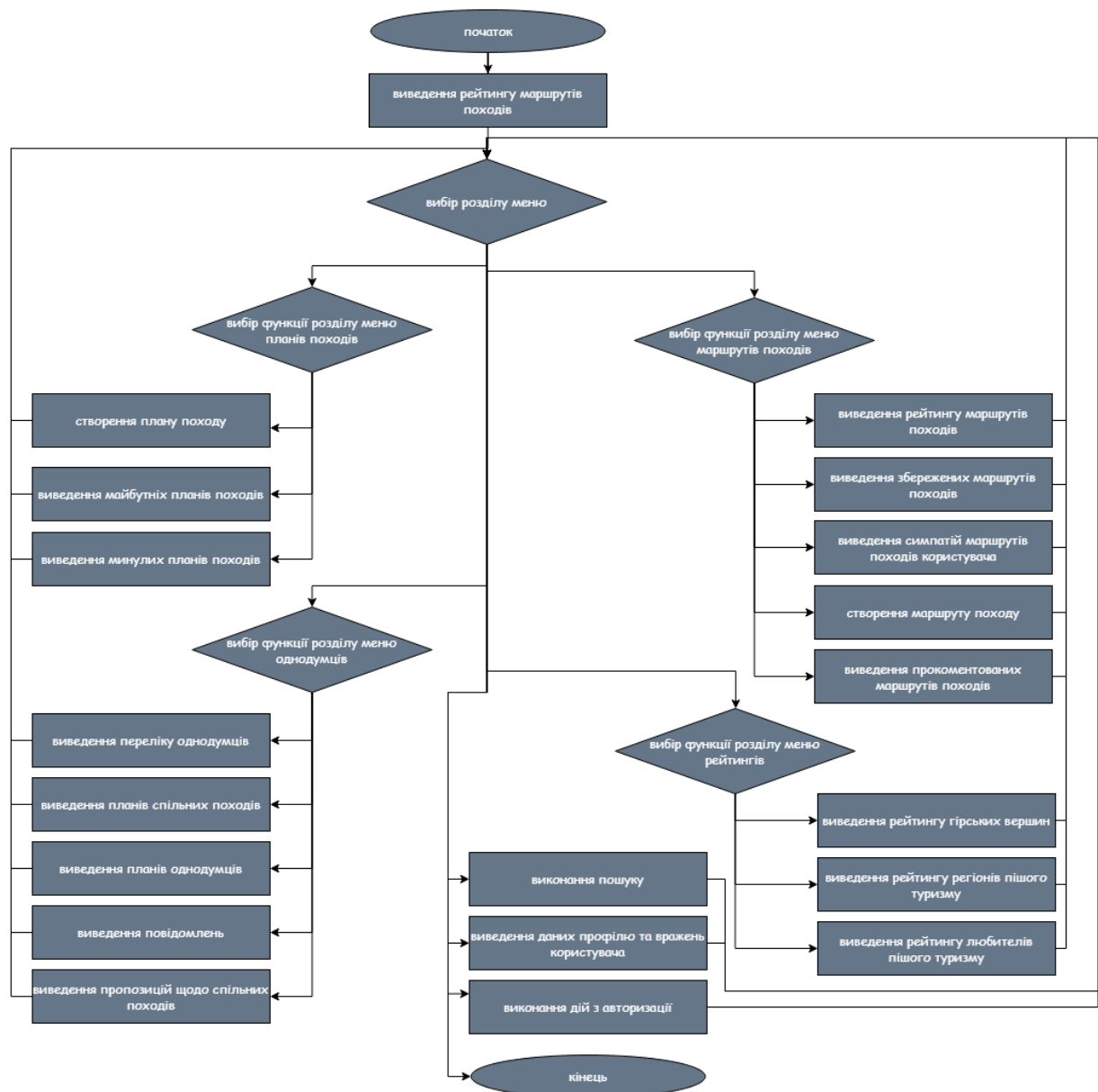


Рисунок 3.1 – Функціональна схема програми знаходження однодумців для пішого туризму

3.2 Результати реалізації програмних одиниць застосунку

Перегляд сторінки регіону пішого туризму реалізований функцією `hikingregionview`, яка реалізує представлення, що працює з заданим ідентифікатором регіону пішого туризму `hikingregionId`, який є параметром функції, а результати заповнюють шаблон вебсторінки `hikingregion.html`, розроблений за допомогою шаблону [8]. До результатів належать дані регіону пішого туризму, 8 останніх вражень любителів пішого туризму про цей регіон, гірські вершини, які знаходяться в цьому регіоні, 8 маршрутів походів пішого туризму з найбільшою кількістю симпатій, кількість симпатій регіону пішого туризму.

Перегляд вражень від відвідувань заданого регіону пішого туризму реалізований функцією `hikingregionimpressionsview`, яка реалізує представлення, що працює з заданим ідентифікатором регіону пішого туризму `hikingregionId`, який є параметром функції, а результати заповнюють шаблон вебсторінки `hikingregionimpressions.html`.

Перегляд сторінки гірської вершини реалізований функцією `peakview`, яка реалізує представлення, що працює з заданим ідентифікатором гірської вершини `peakId`, який є параметром функції, а результати заповнюють шаблон вебсторінки `peak.html`. До результатів належать дані гірської вершини, 8 останніх вражень любителів пішого туризму про гірську вершину, 8 маршрутів гірських походів, що проходять через цю гірську вершину, з найбільшою кількістю уподобань.

Перегляд вражень від відвідувань заданої гірської вершини реалізований функцією `peakimpressionsview`, яка реалізує представлення, що працює з заданим ідентифікатором гірської вершини `peakId`, який є параметром функції, а результати заповнюють шаблон вебсторінки `peakimpressions.html`.

Перегляд рейтингу гірських вершин реалізований функцією `peakratingview`, де визначаються 100 гірських вершин з найкращою середньою оцінкою, заповнюючи шаблон вебсторінки `peakrating.html`.

Перегляд рейтингу регіонів пішого туризму реалізований функцією `hikingregionratingview`, де визначаються 100 регіонів пішого туризму з найкращою середньою оцінкою, заповнюючи шаблон вебсторінки `hikingregionrating.html`.

Перегляд маршрутів походів за заданим регіоном пішого туризму реалізований функцією `hikingregionroutesview`, яка реалізує представлення, що працює з заданим ідентифікатором регіону пішого туризму `hikingregionId`, який є параметром функції, а результати заповнюють шаблон вебсторінки `hikingregionroutes.html`. Маршрути впорядковуються за кількістю висловлених симпатій з найбільшої. За кожним маршрутом походу пішого туризму встановлюється, якщо любитель пішого туризму працює як авторизований користувач, чи він зберіг цей маршрут, чи він поставив йому симпатію.

Перегляд маршрутів походів за заданою гірською вершиною реалізований функцією `peakroutesview`, яка реалізує представлення, що працює з заданим ідентифікатором гірської вершини `peakId`, який є параметром функції, вибираючи ті маршрути, які проходять через цю гірську вершину, впорядковуючи їх за кількістю симпатій, а результати заповнюють шаблон вебсторінки `peakroutes.html`.

Перегляд маршрутів походів любителя пішого туризму реалізований функцією `hikerroutesview`, яка реалізує представлення, що працює з заданим користувачем з логіном `hikerName` або любителем пішого туризму, який авторизувався, якщо логін не задано. Результати направляються в шаблон сторінки `hikingroutes.html`.

Перегляд переліку маршрутів походів пішого туризму, які прокоментував користувач, реалізований функцією `commentedroutesview`. Дані стосуються користувача, який авторизувався. Вибираються коментарі цього автора, а з них унікальні маршрути походів. За кожним маршрутом

встановлюється, чи встановив симпатію, зберіг цей маршрут користувач. Результати направляються в шаблон сторінки `commentedhikingroutes.html`.

Перегляд планів подорожей власних однодумців реалізований функцією `attentionhikingplansview`. Відбираються однодумці авторизованого користувача, а за ними всі плани походів пішого туризму, які ці користувачі створили, починаючи з тих, які створені останніми. Результати направляються в шаблон сторінки `attentionhikingplans.html`.

Перегляд планів майбутніх походів любителя пішого туризму реалізований функцією `futurehikingplansview`, що працює з заданим логіном любителя пішого туризму `hikerName`, який є параметром функції, а результати заповнюють шаблон вебсторінки `futurehikingplans.html`. Визначаються маршрути походів, початок яких пізніше за поточний момент.

Перегляд планів завершених походів любителя пішого туризму реалізований функцією `historyhikingplansview`, що працює з заданим логіном любителя пішого туризму `hikerName`, який є параметром функції, а результати заповнюють шаблон вебсторінки `historyhikingplans.html`. Визначаються маршрути походів, початок яких пізніше за поточний момент.

Перегляд власних вражень та вражень заданого користувача реалізований функцією `hikerimpressionsview`, що працює з заданим логіном любителя пішого туризму `hikerName`, який є параметром функції, а результати заповнюють шаблон вебсторінки `hikerimpressions.html`. До результатів входять враження від взаємодії з любителями пішого туризму, враження самого користувача від взаємодії з іншими любителями пішого туризму, враження від відвідування регіону пішого туризму, враження від відвідування гірської вершини, за кожною групою по 8 останніх.

Перегляд вражень різних користувачів від взаємодії з заданим любителем пішого туризму реалізований функцією `abouthikerimpressionsview`, що працює з заданим логіном любителя пішого туризму `hikerName`, який є параметром функції, а результати заповнюють шаблон вебсторінки `abouthikerimpressions.html`.

Перегляд профілю любителя пішого туризму реалізований функцією `hikerprofileview`, що працює з заданим логіном любителя пішого туризму `hikerName`, який є параметром функції, а результати заповнюють шаблон вебсторінки `hiker.html`. Сюди також входять враження, дані самого любителя пішого туризму, його майбутні та минулі плани походів пішого туризму.

Перегляд маршруту походу пішого туризму реалізований функцією `hikingrouteview`, що працює з заданим маршрутом походу пішого туризму `hikingrouteId`, який є параметром функції, а результати заповнюють шаблон вебсторінки `hikingroute.html`. До результатів входять загальні дані самого маршруту походу, перелік запинок маршруту, включаючи інформацію про пов'язані гірські вершини з цими зупинками, коментарі до маршруту походу, а також дані про симпатію та збереження маршруту походу авторизованим користувачем.

Перегляд рейтингу любителів піших походів реалізований функцією `hikerratingview`, що відбирає 100 найкращих за оцінкою любителів піших походів та заповнює ними сторінку `hikerrating.html`.

Перегляд плану походу пішого туризму реалізований функцією `hikingplanview`, що працює з заданим планом походу пішого туризму `hikingplanId`, який є параметром функції, а результати заповнюють шаблон вебсторінки `hikingplan.html`. До результатів належать дані самого плану походу, фотографії, якщо це план походу, який відбувся, опис кожного з днів походу, дані маршрутів, які відповідають тим дням походу.

Збереження враження від взаємодії з любителем пішого туризму реалізовано функцією `hikerimpressionaddview`, яка реалізує представлення, що працює з заданим користувачем з логіном `hikerName`. Формування самого враження реалізується в БД на основі об'єкту класу форми `Hikerimpressionform`. Аналогічно збереження враження від відвідування регіону пішого туризму користувачем реалізовано функцією `regionimpressionaddview`, збереження враження від відвідування гірської вершини користувачем – функцією `peakimpressionaddview`.

Створення плану походу реалізовано функцією `hikingplanaddview`, як підтримується формами класів `Hikingplanform` та `Hikingdayform` для визначення розподілу за днями.

Редагування плану походу реалізовано функцією `hikingplaneditview`, що працює з заданим планом походу пішого туризму `hikingplanId`, який є параметром функції з перевіркою того, що авторизований користувач створив цей план походу. Підтримка також за відповідними формами відбувається, а для цієї функції, як і попередньої, а також для інших функцій створення, редагування не забезпечується шаблон вебсторінки, а відбувається перенаправлення за результатами збереження на іншу сторінку (в даному випадку перегляду власних планів походів).

Перегляд рейтингу маршрутів походів пішого туризму реалізований функцією `hikingroutesratingview`, де визначаються 100 маршрутів походів пішого туризму з найкращою середньою оцінкою, заповнюючи шаблон вебсторінки `hikingroutesrating.html`. Функція має параметр `hikingregionId`, який задає ідентифікатор регіону пішого туризму, а якщо він пропущений, то відбір відбувається зі всіх регіонів загалом.

Збереження маршруту походу пішого туризму реалізоване функцією `hikingroutesaveview`, що працює з заданим маршрутом походу пішого туризму `hikingrouteId`, який є параметром функції, перевіряючи наявність у БД запису про збереження цим любителем пішого туризму цього маршруту походу: зберігаючи у випадку відсутності, вилучаючи у випадку наявності.

Визначення симпатії маршруту походів пішого туризму реалізоване функцією `hikingroutelikeview`, що працює з заданим маршрутом походу пішого туризму `hikingrouteId`, який є параметром функції, перевіряючи наявність у БД запису про визначення симпатії цим любителем пішого туризму цьому маршруту походу: зберігаючи у випадку відсутності, вилучаючи у випадку наявності.

Перегляд збережених маршрутів піших походів реалізований функцією `hikingroutesavesview` за любителем пішого туризму, який авторизувався.

Створення пропозиції про спільний похід реалізовано функцією `hikingplanpropositionaddview`, що працює з заданим планом походу пішого туризму `hikingplanId`, який є параметром функції, залежно від того, чи авторизований користувач сам раніше створив цей план, чи ні, обираючи сторону, а текст пропозиції передаючи через параметр функції `proposition`.

Скасування, підтвердження пропозиції про спільний похід реалізовано функцією `hikingplanpropositionmanageview`, що працює з заданою пропозицією про спільний похід `hikingplanpropositionId`, яка є параметром функції. Параметр `change` визначає потрібну зміну (підтвердження, скасування). Виконується перевірка сторін: підтвердити може не та сторона, яка подавала пропозицію. Не можна змінити пропозицію, якщо час походу за планом за нею вже розпочався.

Перегляд пропозицій про спільний похід, які адресовані користувачу, що авторизувався, реалізовано функцією `hikingplanpropositionsview`. Результати направляються в шаблон сторінки `hikingplanpropositions.html`.

Перегляд пропозицій про спільний похід, які створені користувачем, що авторизувався, реалізовано функцією `tohikingplanpropositionsview`. Результати направляються в шаблон сторінки `tohikingplanpropositions.html`.

Перегляд планів спільних походів пішого туризму реалізовано функцією `plushikingplanpropositionsview` для авторизованого любителя пішого туризму. Параметр `timer` визначає, чи це мають бути завершені, чи тільки заплановані плани спільних подорожей. Результати направляються в шаблон сторінки `plushikingplanpropositions.html`.

Не представлені функції влаштовані подібно, тому не приводилось детально у цьому розділі.

Клас моделі `Hiker` визначає користувача, який є любителем пішого туризму, з такими полями.

Поле `useraccount` визначає обліковий запис, за яким працює користувач, який є любителем пішого туризму, у програмі, а оскільки це зв'язок зі стандартною таблицею `User Django`, то використано для реалізації клас

OneToOneField за цією моделлю, тобто зв'язок один-до-одного (один обліковий запис не можуть використовувати різні любителі пішого туризму).

Поле `hikename` визначає прізвище, за яким відомий цей любитель пішого туризму серед колег, на основі класу `CharField`.

Поле `fullname` визначає повне ім'я цього любителя пішого туризму на основі класу `CharField`.

Поле `hikeexperience` визначає рік, з якого займається пішим туризмом цей користувач, на основі класу `PositiveSmallIntegerField`.

Поле `birthdate` визначає дату народження любителя пішого туризму на основі класу `DateField`.

Поле `countryregion` визначає область країни, в якій живе користувач, який є любителем пішого туризму, на основі класу `ForeignKey` зі зв'язуванням з моделлю `Countryregion`.

Поле `hikerdescription` визначає презентаційний опис любителя пішого туризму на основі класу `CharField`.

Поле `impression` визначає середню оцінку, яку за враженнями від взаємодії з іншими користувачами було виставлено цьому любителю пішого туризму, на основі класу `FloatField`.

Поле `hikingroutesaves` визначає маршрути походів, які збережено цим користувачем, який є любителем пішого туризму, на основі класу `ManyToManyField` зі зв'язуванням з моделлю `Hikingroute`.

Поле `hikingroutelikes` визначає маршрути походів пішого туризму, за якими визначена симпатія цього користувачем, який є любителем пішого туризму, на основі класу `ManyToManyField` зі зв'язуванням з моделлю `Hikingroute`.

Поле `hikerattenteds` визначає користувачів, які є любителями пішого туризму і які визначили своїм однодумцем стосовно пішого туризму цього користувача, на основі класу `ManyToManyField` зі зв'язуванням з моделлю `Hiker`.

Клас моделі `Hikerimpression` визначає враження від взаємодії з користувачем, який є любителем пішого туризму, з такими полями.

Поле `hiker` визначає користувача, який є любителем пішого туризму та враження про якого визначено, на основі класу `ForeignKey` зі зв'язуванням з моделлю `Hiker`.

Поле `hikerimpressed` визначає користувача, який є любителем пішого туризму та який отримав і зберіг це враження, на основі класу `ForeignKey` зі зв'язуванням з моделлю `Hiker`.

Поле `point` визначає числову оцінку за даним враженням, виставлена користувачу, який є любителем пішого туризму, на основі класу `PositiveSmallIntegerField`.

Поле `impressiontext` визначає текстом висловлене враження про користувача, який є любителем пішого туризму, на основі класу `CharField`.

Поле `impressiondate` визначає дату, коли було збережено враження про любителя пішого туризму, на основі класу `DateField`.

Поле `commonplan` визначає визначення того, чи було це враження про любителя пішого туризму, підтверджено спільним планом участі в поході, чи ні, на основі класу `BooleanField`.

Клас моделі `Hikerimage` визначає зображення любителя пішого туризму з такими полями.

Поле `hiker` визначає любителя пішого туризму, який представлений на цьому зображенні, на основі класу `ForeignKey` зі зв'язуванням з моделлю `Hiker`.

Поле `image` визначає вміст зображення, на якому представлений цей любитель пішого туризму, на основі класу `ImageField`.

Поле `mainorno` визначає те, чи це зображення є основним, тобто застосовується для представлення самого любителя пішого туризму там, де потрібне тільки одне зображення, чи ні, реалізовано на основі класу `BooleanField`.

Клас моделі `Hikingplan` визначає план походу пішого туризму з такими полями.

Поле `hiker` визначає користувача, який є любителем пішого туризму та який створив цей похід пішого туризму, яка вже відбулась або планується, на основі класу `ForeignKey` зі зв'язуванням з моделлю `Hiker`.

Поле `hikingstart` визначає дату, коли має розпочатися похід пішого туризму або коли вона вже фактично розпочалась, якщо описується історія, за цим планом, на основі класу `DateField`.

Поле `hikingend` визначає дату, коли має закінчитися похід пішого туризму або коли вона вже фактично закінчилась, якщо описується історія, за цим планом, на основі класу `DateField`.

Поле `caption` визначає загальну назву як підпис плану походу пішого туризму на основі класу `CharField`.

Поле `overalldescription` визначає детальний загальний опис плану походу пішого туризму на основі класу `CharField`.

Поле `presentimage` визначає зображення для ілюстрації на основі класу `ImageField`.

Клас моделі `Hikinghistory` визначає історію походу пішого туризму за планом на основі фотографій з такими полями.

Поле `hikingplan` визначає план походу пішого туризму, під час якої було отримано це зображення, на основі класу `ForeignKey` зі зв'язуванням з моделлю `Hikingplan`.

Поле `image` визначає вміст зображення, на якому представлений момент з походу пішого туризму, яка вже відбулась, на основі класу `ImageField`.

Клас моделі `Hikingregion` визначає регіони пішого туризму з такими полями.

Поле `regionname` визначає назву регіону пішого туризму на основі класу `CharField`.

Поле `impression` визначає середню оцінку, яку за враженнями було виставлено цьому регіону пішого туризму любителями пішого туризму, які його відвідували, на основі класу `FloatField`.

Поле `presentimage` визначає базове зображення, на якому показаний цей регіон пішого туризму, на основі класу `ImageField`.

Поле `countryregions` визначає області країни, в яких розташований цей регіон пішого туризму, на основі класу `ManyToManyField` зі зв'язуванням з моделлю `Countryregion`.

Клас моделі `Hikingregionimpression` визначає враження від регіону пішого туризму з такими полями.

Поле `hiker` визначає користувача, який є любителем пішого туризму та який зберіг враження про регіон пішого туризму, на основі класу `ForeignKey` зі зв'язуванням з моделлю `Hiker`.

Поле `hikingregion` визначає регіон пішого туризму, про який зберіг враження користувач, який є любителем пішого туризму, на основі класу `ForeignKey` зі зв'язуванням з моделлю `Hikingregion`.

Поле `hikingstart` визначає дату початку того походу пішого туризму, на основі якого любитель пішого туризму створив це враження, на основі класу `DateField`.

Поле `hikingend` визначає дату закінчення того походу пішого туризму, на основі якого любитель пішого туризму створив це враження, на основі класу `DateField`.

Поле `point` визначає числову оцінку за даним враженням, виставлену регіону пішого туризму користувачем, який є любителем пішого туризму та побував у цьому регіоні, на основі класу `PositiveSmallIntegerField`.

Поле `impressiontext` визначає текстом висловлене враження користувачем, який є любителем пішого туризму, про регіон пішого туризму, на основі класу `CharField`.

Поле `impressiondate` визначає дату, коли було збережено враження про регіон пішого туризму любителем пішого туризму, на основі класу `DateField`.

Клас моделі Countryregion визначає області країн, що підтримуються в програмі, з такими полями.

Поле country визначає країну, в якій знаходиться область, на основі класу CharField.

Поле regionname визначає назву області на основі класу CharField.

Клас моделі Peak визначає гірські вершини з такими полями.

Поле peakname визначає назву гірської вершини на основі класу CharField.

Поле hikingregion визначає регіон пішого туризму, де знаходиться ця гірська вершина, на основі класу ForeignKey зі зв'язуванням з моделлю Hikingregion.

Поле impression визначає середню оцінку, яку за враженнями було виставлено цій гірській вершині любителями пішого туризму, які її відвідували, на основі класу FloatField.

Поле presentimage визначає базове зображення, на якому показана ця гірська вершина, на основі класу ImageField.

Клас моделі Peakimpression визначає враження від відвідування гірських вершин користувачами, які є любителями пішого туризму, з такими полями.

Поле hiker визначає любителя пішого туризму, який зберіг це враження від відвідування цієї гірської вершини, на основі класу ForeignKey зі зв'язуванням з моделлю Hiker.

Поле peak визначає гірську вершину, відвідування якої стосується враження любителя пішого туризму, на основі класу ForeignKey зі зв'язуванням з моделлю Peak.

Поле hikingstart визначає дату початку того походу пішого туризму, на основі якого любитель пішого туризму створив це враження, на основі класу DateField.

Поле `hikingend` визначає дату закінчення того походу пішого туризму, на основі якого любитель пішого туризму створив це враження, на основі класу `DateField`.

Поле `point` визначає числову оцінку за даним враженням, виставлену цій гірській вершині користувачем, який є любителем пішого туризму та побував на цій вершині, на основі класу `PositiveSmallIntegerField`.

Поле `impressiontext` визначає текстом висловлене враження користувачем, який є любителем пішого туризму, про цю гірську вершину, на основі класу `CharField`.

Поле `impressiondate` визначає дату, коли було збережено враження про цю гірську вершину любителем пішого туризму, на основі класу `DateField`.

Клас моделі `Hikingday` визначає дні плану походу пішого туризму з такими полями.

Поле `hikingplan` визначає план походу пішого туризму, до якого відноситься цей день, на основі класу `ForeignKey` зі зв'язуванням з моделлю `Hikingplan`.

Поле `daynumber` визначає порядковий номер цього дня у цьому плані походу пішого туризму на основі класу `PositiveSmallIntegerField`.

Поле `dayname` визначає коротку назву, що представляє суть цього дня у цьому плані походу пішого туризму, на основі класу `CharField`.

Поле `daydescription` визначає детальний опис цього дні походу пішого туризму за планом на основі класу `CharField`.

Поле `hikingroute` визначає маршрут походу, за яким відбувається за планом похід в цей день, на основі класу `ForeignKey` зі зв'язуванням з моделлю `Hikingroute`.

Поле `presentimage` визначає зображення для ілюстрації на основі класу `ImageField`.

Клас моделі `Hikingroute` визначає маршрут походу з такими полями.

Поле `caption` визначає загальну коротку назва походу як підпис на основі класу `CharField`.

Поле `routedescription` визначає деталізований опис маршруту походу на основі класу `CharField`.

Поле `authorhiker` визначає любителя пішого туризму, який створив цей маршрут походу, на основі класу `ForeignKey` зі зв'язуванням з моделлю `Hiker`.

Поле `impression` визначає середню оцінку, яку за враженнями було виставлено цьому маршруту походу любителями пішого туризму, які його використовували, на основі класу `FloatField`.

Клас моделі `Hikingroutestop` визначає конкретні зупинки або розгалудження маршруту походу з такими полями.

Поле `hikingroute` визначає маршрут походу, в який входить ця зупинка або розгалудження, на основі класу `ForeignKey` зі зв'язуванням з моделлю `Hikingroute`.

Поле `stoplat` визначає широту позиції зупинки або розгалудження з маршруту походу на мапі на основі класу `FloatField`.

Поле `stoplong` визначає довготу позиції зупинки або розгалудження з маршруту походу на мапі на основі класу `FloatField`.

Поле `peak` визначає гірські вершини, на яких знаходиться ця зупинка або розгалудження маршруту походу на мапі, на основі класу `ManyToManyField` зі зв'язуванням з моделлю `Peak`.

Поле `number` визначає порядковий номер зупинки або розгалудження в маршруті походу на основі класу `PositiveSmallIntegerField`.

Клас моделі `Hikingplansource` визначає джерела, які використовували та хоча позначити стосовно власного плану походу пішого туризму користувач, який є любителем пішого туризму, з такими полями.

Поле `hikingplan` визначає план походу пішого туризму, при створенні якого використано або який ілюструє це джерело, на основі класу `ForeignKey` зі зв'язуванням з моделлю `Hikingplan`.

Поле `sourceurl` визначає посилання на джерело, яке використовували та хоча позначити стосовно власного плану походу пішого туризму користувач, який є любителем пішого туризму, на основі класу `URLField`.

Клас моделі `Hikingplanproposition` визначає пропозиції до можливих однодумців стосовно спільних походів пішого туризму з такими полями.

Поле `hikingplan` визначає план пішого туризму, за яким буде здійснюватися цей пропонований спільний похід, на основі класу `ForeignKey` зі зв'язуванням з моделлю `Hikingplan`.

Поле `hiker` визначає користувач, який є любителем пішого туризму та якому адресовано цю пропозицію стосовно спільного походу пішого туризму як до однодумця, на основі класу `ForeignKey` зі зв'язуванням з моделлю `Hiker`.

Поле `hikersource` визначає позначення того, яка зі сторін є ініціатором пропозиції до можливого однодумця стосовно спільного походу пішого туризму, на основі класу `BooleanField`.

Поле `proposition` визначає текст, який доданий до пропозиції до можливого однодумця стосовно спільного походу пішого туризму, на основі класу `CharField`.

Поле `approve` визначає підтвердження, відхилення або відсутність відповіді стосовно пропозиції до можливого однодумця стосовно спільного походу пішого туризму на основі класу `PositiveSmallIntegerField`.

Поле `propositiontime` визначає час створення пропозиції до можливого однодумця стосовно спільного походу пішого туризму на основі класу `DateTimeField`.

Клас моделі `Hikingroutecomment` визначає коментарі до маршрутів походів пішого туризму з такими полями.

Поле `hikingroute` визначає маршрут походу пішого туризму, до якого користувач, який є любителем пішого туризму, написав коментар, на основі класу `ForeignKey` зі зв'язуванням з моделлю `Hikingroute`.

Поле `hiker` визначає користувача, який є любителем пішого туризму та написав цей коментар до маршруту походу пішого туризму, на основі класу `ForeignKey` зі зв'язуванням з моделлю `Hiker`.

Поле `comment` визначає текст коментаря до маршруту походу пішого туризму користувача, який написав коментар, на основі класу `CharField`.

Поле `commenttime` визначає час, коли було написано коментар до походу пішого туризму, на основі класу `DateTimeField`.

Клас моделі `Hikermesssage` визначає персональне повідомлення між однодумцями з пішого туризму з такими полями.

Поле `hiker` визначає користувача, який є любителем пішого туризму та який надіслав повідомлення до свого однодумця з пішого туризму, на основі класу `ForeignKey` зі зв'язуванням з моделлю `Hiker`.

Поле `hikermesssaged` визначає користувача, який є любителем пішого туризму та якому було надіслано повідомлення однодумцем з пішого туризму, на основі класу `ForeignKey` зі зв'язуванням з моделлю `Hiker`.

Поле `messagetext` визначає текст персонального повідомлення між однодумцями з пішого туризму на основі класу `CharField`.

Поле `messagetime` визначає час відправлення персонального повідомлення між однодумцями з пішого туризму на основі класу `DateTimeField`.

3.3 Висновки за розділом 3

Представлена функціональна схема програми знаходження однодумців для пішого туризму, що базується на меню вебзастосунку, реалізовано всі функції програми та описано створені програмні одиниці.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Призначення програми

Програма призначена для знаходження однодумців для пішого туризму, визначення власних вражень від відвідування гірських вершин, регіонів пішого туризму, взаємодії з іншими користувачами, визначення спільних походів, визначення планів та маршрутів власних походів пішого туризму.

4.2 Умови виконання програми

Обов'язковою умовою експлуатації програми знаходження однодумців для пішого туризму є наявність адміністратора програми або користувача, здатного виконувати такі функції. В обов'язки адміністратора входить наповнення даними гірський вершин, регіонів пішого туризму, областей країн. Також адміністратор повинен виконати розгортання додатку на сервері.

Технічне забезпечення сервера повинно включати мінімально:

- процесор Intel Xeon E5440 (частота 2,83 ГГц);
- жорсткий диск – 100 Гб;
- оперативна пам'ять – 4 Гб.

4.3 Виконання і тестування програми

Реалізацію тестування забезпечено в межах перевірки набору функціональних можливостей, які повинні були бути забезпеченими для роботи користувача. Тестування продемонструвало успішні результати забезпечення всіх запланованих функцій. Далі на основі результатів тестування описано приклади і логіку роботи з програмою.

Користувач може використовувати цілий ряд варіантів рейтингів з однойменного розділу меню, включаючи рейтинги любителів пішого туризму,

гірських вершин та цілих регіонів пішого туризму (рис. 4.1). Для кожної позиції виводиться базова інформація з можливістю переходу на відповідну окрему сторінку, при наведенні на фотографію виводиться оцінка за враженнями любителів пішого туризму.

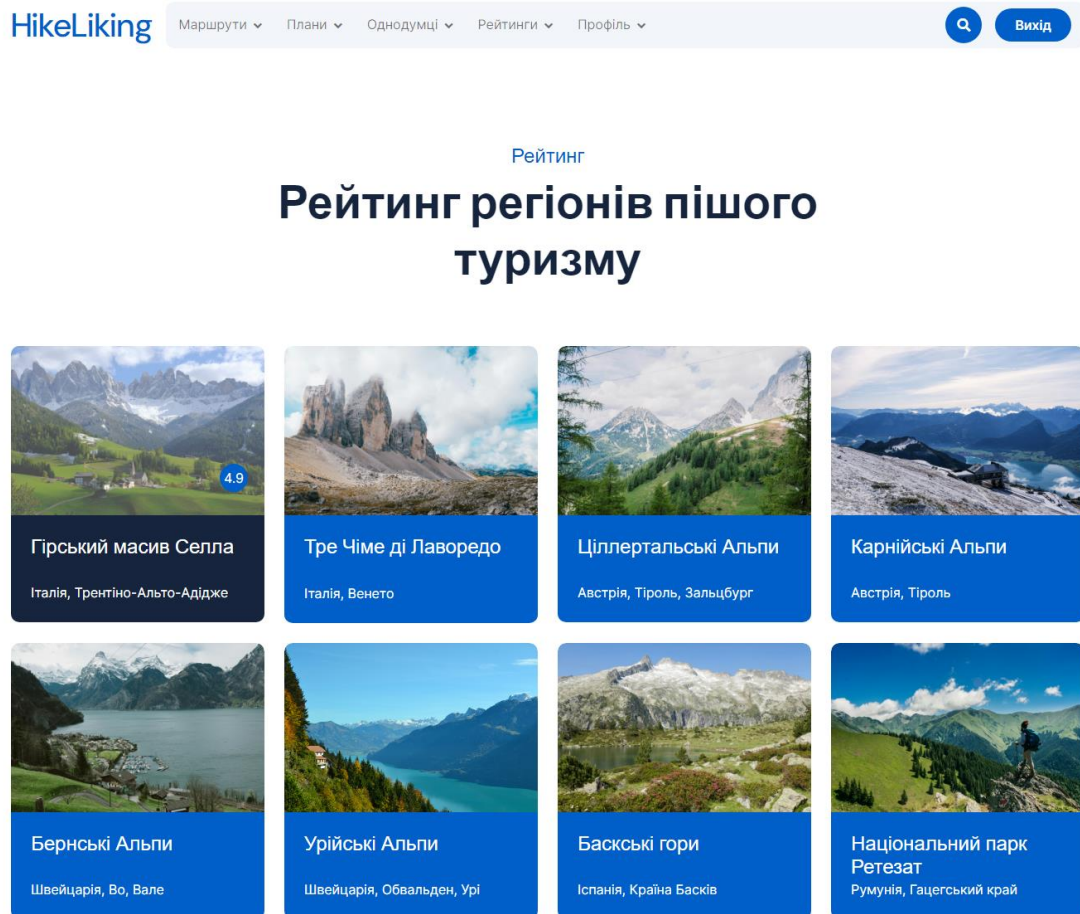


Рисунок 4.1 – Сторінка рейтингу регіонів пішого туризму

З розділу меню маршрутів можна перейти до рейтингу маршрутів походів, створити новий свій власний маршрут, переглянути збережені маршрути походів, ті маршрути походів, які прокоментував сам користувач, симпатії, які стосуються маршрутів походів. Кожен маршрут можна відкрити і детальну з ним ознайомитись, включаючи кожен зупинку або розгалудження, їх відповідність вершинам гір.

З розділу меню планів можна перейти до створення нового плану походу, можна переглянути власні минулі та майбутні плани походів. На

сторінці плану походу відображається інформація про його назву, опис, період походу, основна фотографія, інформація про автора з можливістю підписатися на цього любителя пішого туризму прямо з цієї ж сторінки або відкрити його окрему сторінку. З цієї ж сторінки можна створити пропозицію для цього любителя пішого туризму, щоб піти з ним в запланований похід разом (рис. 4.2).

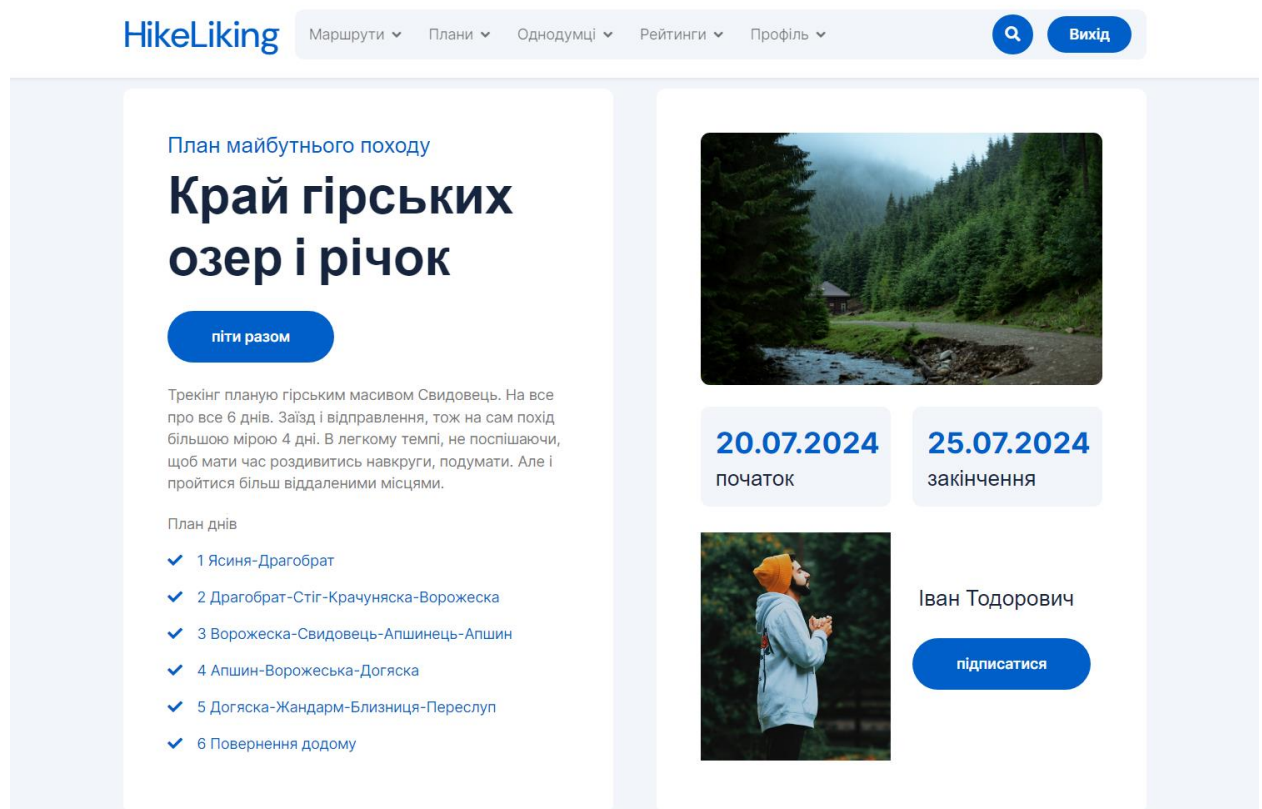


Рисунок 4.2 – Сторінка плану походу

На цій же сторінці відображається план на кожен день походу. Кожен день включає назву, опис, можливість перейти до перегляду детально маршруту походу в цей день, фотографію, можливість визначити симпатію цьому маршруту походу, кількість симпатій на поточний момент, можливість зберегти маршрут, а знизу відображається мапа з маршрутом походу в цей день на ній (рис. 4.3). Саме ця функція відображення маршруту на мапі відсутня у аналогів.

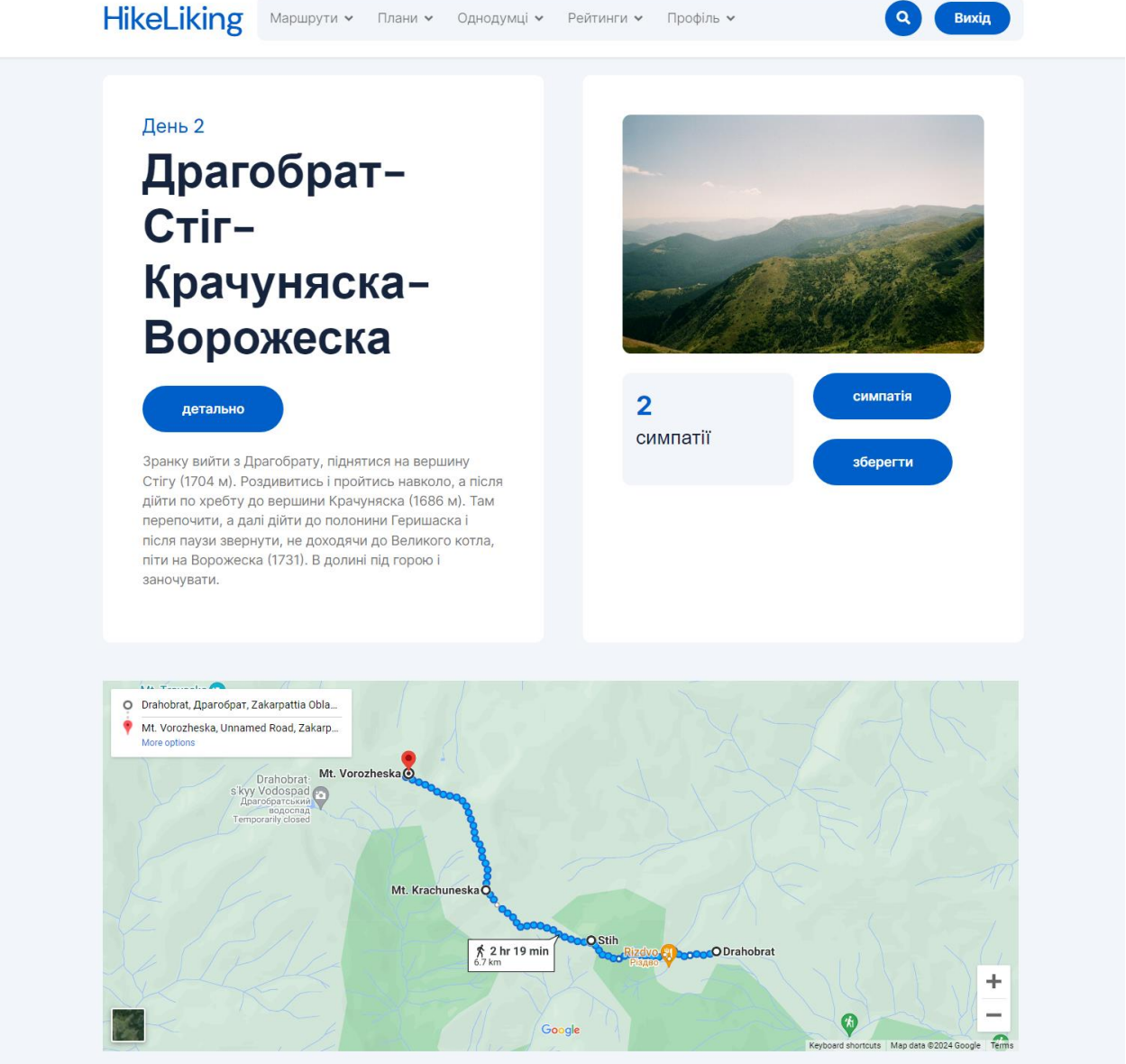
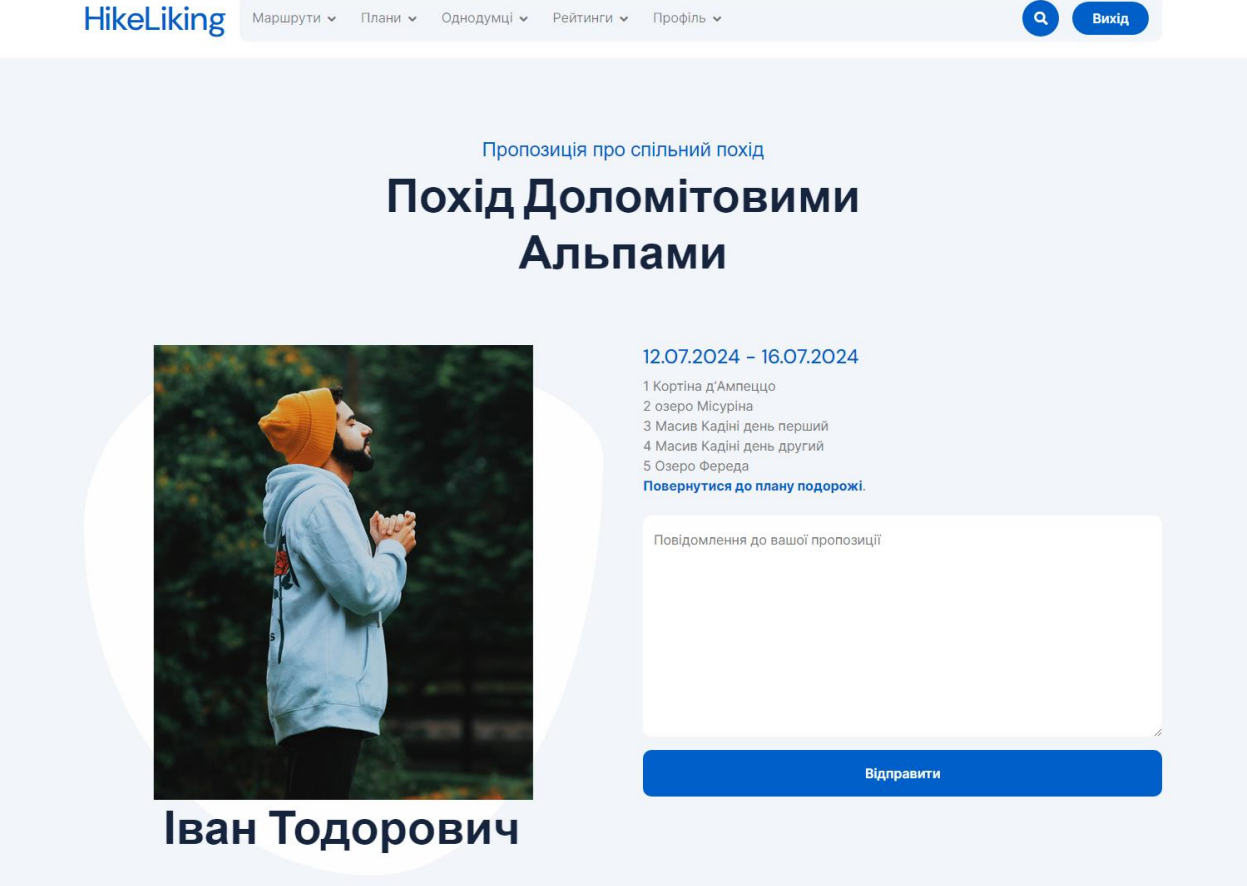


Рисунок 4.3 – План дня на сторінці плану походу

Сторінка створення пропозиції про спільний похід (рис. 4.4), на яку можна перейти через відповідну кнопку зі сторінки плану походу, містить інформацію про автора плану походу, коротко назви днів походу, назву всього плану походу, а також поле для визначення повідомлення до пропозиції і кнопку для надсилання такої пропозиції. Далі інша сторона може скасовувати або підтримувати цю пропозицію.

Самі пропозиції доступні з розділу меню однодумців, звідки можна окрім цього переглянути також спільні плани походів, тобто підтвержені пропозиції, перелік однодумців (на які підписався користувач), плани походів

таких однодумців, надіслані повідомлення. Повідомлення надсилаються або з самих вже наявних діалогів, або потрібно відкрити сторінку потрібного користувача, звідки доступна кнопка створення повідомлення.



HikeLiking Маршрути Плани Однодумці Рейтинги Профіль Вихід

Пропозиція про спільний похід

Похід Долomiteвими Альпами

12.07.2024 – 16.07.2024

- 1 Кортіна д'Ампеццо
- 2 озеро Місуріна
- 3 Масив Кадіні день перший
- 4 Масив Кадіні день другий
- 5 Озеро Ферета

[Повернутися до плану подорожі.](#)

Повідомлення до вашої пропозиції

Відправити

Іван Тодорович

Рисунок 4.4 – Сторінка створення пропозиції про спільний похід

Пошук реалізований універсально з вибором з випадającego переліку, чи його виконувати за гірськими вершинами, чи за регіонами пішого туризму, чи за планами походів, чи за маршрутами.

4.4 Висновки за розділом 4

Програму знаходження однодумців для пішого туризму успішно реалізовано з забезпеченням всіх функціональних можливостей, що були визначені, з наданням відповідного меню для керування послідовністю цих дій.

ВИСНОВКИ

Виходячи з отриманих результатів, у роботі було досягнуто мету, якою є розробка програмного забезпечення знаходження однодумців для пішого туризму, щоби сприяти підвищенню взаємодії в цій галузі та розвитку галузі загалом.

Виконання завдання аналізу програмного забезпечення, придатного для знаходження однодумців для пішого туризму, дозволило виявити, що подібні рішення або підтримують організацію однієї конкретної туристичної групи, або мають функціональність, направлену на пошук учасників подій в широкому сенсі, не визначаючи ті складові пішого туризму, які є важливими в походах. На ліквідацію цих недоліків стосовно саме знаходження однодумців для пішого туризму було направлено дану розробку. Щоби визначити, які саме функціональні вимоги програма має забезпечувати, створено діаграму прецедентів на основі визначення відповідних сценаріїв користувача.

Результатом виконання завдання визначення засобів проєктування програми є набір технологій, до яких входять мови Python, JavaScript, система MySQL, ВФ Django. Ще одним результатом виконання роботи і одночасно загальним результатом є програма знаходження однодумців для пішого туризму, що охоплює не тільки можливості встановлення контактів між користувачами, але і визначення власних планів походів, щоби створити підґрунтя щодо взаємодії стосовно конкретних спільних походів, створення екосистеми пішого туризму, в якій визначено регіони пішого туризму, гірські вершини, маршрути.

Програма у підсумку призначена для знаходження однодумців для пішого туризму, визначення власних вражень від відвідування гірських вершин, регіонів пішого туризму, взаємодії з іншими користувачами, визначення спільних походів, визначення планів та маршрутів власних походів пішого туризму.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Pacific Crest Trail Association – Preserving, Protecting and Promoting [Electronic resource]. – Access mode : <https://www.pcta.org/>.
2. Adventures – Zpacks [Electronic resource]. – Access mode : <https://zpacks.com/blogs/adventures>.
3. Meetup. Find Local Groups, Events, and Activities Near You [Electronic resource]. – Access mode : <https://www.meetup.com/>.
4. Hiking Together [Electronic resource] : Finding Others to Hike With – Social Hikers. – Access mode : <https://socialhiker.net/hiking-together-finding-others-to-hike-with/>.
5. Getting started with Django. Django [Electronic resource]. – Access mode : <https://www.djangoproject.com/start/>.
6. The Pyramid Web Framework [Electronic resource]. – Access mode : <https://docs.pylonsproject.org/projects/pyramid/en/latest/>.
7. Tomaszuk, D. Python's Frameworks Comparison [Electronic resource] : Django, Pyramid, Flask, Sanic, Tornado, BottlePy and More / D. Tomaszuk. – Access mode : <https://www.netguru.com/blog/python-frameworks-comparison>.
8. LifeSure – Life Insurance Website Template [Electronic resource]. – Access mode : <https://htmlcodex.com/life-insurance-website-template/>.

ДОДАТОК А
Технічне завдання

Вступ

Піший туризм є достатньо привабливою галуззю, яка можливо не досить є медіаактивною в нашій країні, але має значно більшу привабливість в інших країнах, визнану виробничою підтримкою. Це стосується виготовлення спеціальних речей, одягу, взуття для пішого туризму. На прикладі того, що виготовленню, продажу таких речей приділяється багато уваги, можна говорити про поширеність загалом цієї галузі. Проте підтримка її здійснюється на достатньо різних рівнях: через інтернет-магазини для продажу обладнання, одягу, взуття для пішого туризму, через відеоролики, які створюють блогери, через визначення маршрутів пішого туризму на окремих вебсайтах з можливостями трекінгу. Проте не приділяється увага реалізації можливості пошуку однодумців для пішого туризму. Це стосується випадків, коли люди готові піти в похід, але шукають цікаву, підходящу компанію для цього. Звичайно, що можливо не кожен шукає саме таких друзів, а йде в похід з друзями з реального життя, яких досить давно знає, проте встановлення контактів між любителями пішого туризму може сприяти тривалості контактів, встановленню таких контактів. Деякі люди можуть бути готовими піти у похід разом, щоб обмінятися досвідом, щоб знайти людей, які відповідають їхньому рівню, зокрема фізичному рівню. Загалом же і популярність соціальних мереж сама по собі підкреслює все ж потенційну можливість для щонайменше деяких людей знайти собі так однодумців. Саме на таких можливостях має бути зроблено акцент у цій розробці.

A.1 Підстави для розробки

Дипломна кваліфікаційна робота виконувалась у відповідності та на підставі Наказу № 168 від 24 квітня 2024 р. за Національним університетом «Запорізька політехніка», яким визначено тему роботи – «Програмне забезпечення знаходження однодумців для пішого туризму».

A.2 Призначення розробки

Програма призначена для знаходження однодумців для пішого туризму, визначення власних вражень від відвідування гірських вершин, регіонів пішого туризму, взаємодії з іншими користувачами, визначення спільних походів, визначення планів та маршрутів власних походів пішого туризму.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Функціональні характеристики програми знаходження однодумців для пішого туризму повинні включати:

- реєстрація любителів пішого туризму як користувачів у програмі, авторизація;
- збереження враження від взаємодії з любителем пішого туризму;
- перегляд власних вражень та вражень заданого користувача, включаючи враження від взаємодії з любителями пішого туризму, враження від відвідування регіону пішого туризму, враження від відвідування гірської вершини;
- перегляд вражень різних користувачів від взаємодії з заданим любителем пішого туризму;
- перегляд сторінки регіону пішого туризму;
- перегляд сторінки гірської вершини;
- перегляд вражень від відвідувань заданого регіону пішого туризму;
- перегляд вражень від відвідувань заданої гірської вершини;
- перегляд маршрутів походів за заданою гірською вершиною;
- перегляд профілю любителя пішого туризму;
- перегляд плану походу пішого туризму;
- редагування профілю любителя пішого туризму;

- створення плану походу, включаючи визначення розподілу за днями;
- редагування плану походу;
- перегляд планів майбутніх походів любителя пішого туризму;
- перегляд планів завершених походів любителя пішого туризму;
- збереження враження від відвідування регіону пішого туризму користувачем;
- збереження враження від відвідування гірської вершини користувачем;
- створення маршруту походу пішого туризму;
- редагування маршруту походу пішого туризму;
- перегляд маршруту походу пішого туризму, включаючи перегляд коментарів;
- перегляд маршрутів походів любителя пішого туризму;
- перегляд рейтингу маршрутів походів пішого туризму загалом, за регіоном;
- перегляд рейтингу любителів піших походів;
- перегляд рейтингу гірських вершин;
- перегляд рейтингу регіонів пішого туризму;
- збереження маршруту походу пішого туризму;
- вилучення маршруту походу пішого туризму зі збережених;
- перегляд збережених маршрутів піших походів;
- створення пропозиції про спільний похід;
- скасування пропозиції про спільний похід;
- підтвердження пропозиції про спільний похід;
- перегляд пропозицій про спільний похід, які адресовані користувачу;
- перегляд пропозицій про спільний похід, які створені користувачем;
- перегляд планів спільних походів пішого туризму, у тому числі завершених та запланованих;
- коментування маршруту походу пішого туризму;
- перегляд переліку маршрутів походів пішого туризму, які

прокоментував користувач;

- визначення симпатії маршруту походів пішого туризму;
- скасування симпатії маршруту походів пішого туризму;
- перегляд власних симпатій до маршрутів походів пішого туризму;
- відправлення персонального повідомлення;
- перегляд створених спілкувань з користувачами;
- перегляд персональних повідомлень за спілкуванням;
- визначення власного одностороннього;
- скасування визначення власного одностороннього;
- перегляд планів подорожей власних односторонніх;
- перегляд переліку власних односторонніх.

A.3.2 Умови експлуатації

Обов'язковою умовою експлуатації програми знаходження односторонніх для пішого туризму є наявність адміністратора програми або користувача, здатного виконувати такі функції. В обов'язки адміністратора входить наповнення даними гірський вершин, регіонів пішого туризму, областей країн. Також адміністратор повинен виконати розгортання додатку на сервері.

A.3.3 Вимоги до складу та параметрів технічних засобів

Технічне забезпечення сервера повинно включати мінімально:

- процесор Intel Xeon E5440 (частота 2,83 ГГц);
- жорсткий диск – 100 Гб (наявність фотографій може значно збільшувати вимоги до сервера, тому це потребує контролю та можливо введення обмежень на обсяг фотографій, які користувач може завантажувати, інакше необхідно розширювати потрібний простір);
- оперативна пам'ять – 4 Гб.

A.4 Порядок контролю та приймання

Починаючи з дати видачі завдання – 15 квітня, протягом наступних 8 тижнів має бути забезпечено виконання роботи за календарним планом, що містить в завданні на роботу. Календарний план визначає послідовність етапів виконання роботи та за кожним таким етапом – строк виконання і основний очікуваний результат. Після забезпечення цих очікуваних результатів формується пояснювальна записка до дипломної кваліфікаційної роботи. Остаточним кроком приймання роботи є її публічний захист.

ДОДАТОК Б
Текст програми

```

from .models import Hikingregion
from django.shortcuts import render
from .models import Hikingregionimpression
from .models import Peak
from .models import Hiker
from .models import Hikingroutestoppeak
from django.db.models import Count
from .models import Peakimpression
from .models import Hikingroutecomment
from .models import Hikingplan
from .models import Hikingroutestop
from .models import Hikerimage

def hikingregionview(request, hikingregionId):
    hikingregion_context = {}
    try:
        hikingregion_data =
Hikingregion.objects.prefetch_related("countryregions")
        hikingregion_data = hikingregion_data.get(id =
hikingregionId)
    except Hikingregion.DoesNotExist:
        return render(request, "hikeliking/error.html",
{"message": "Про такий регіон пішого туризму мені нічого невідомо.
Зв'яжіться з адміністратором, щоб він дав тому раду, якщо він все ж є,
а я про те нічого не знаю. Або вкажіть правильний номер"})
        hikingregion_context["region"] = hikingregion_data
        hikregionimpressions =
Hikingregionimpression.objects.select_related("hiker")
        hikregionimpressions =
hikregionimpressions.filter(hikingregion = hikingregion_data)
        hikregionimpressions = hikregionimpressions.order_by("-
impressiondate")
        for hikregionimpression in hikregionimpressions:
            hikerimages = Hikerimage.objects.filter(hiker =
hikregionimpression.hiker, mainorno = True)
            hikregionimpression.image = hikerimages
            hikingregion_context["impressions"] =
hikregionimpressions[:8]
            authhiker = None
            if request.user.is_active:
                authhiker =
Hiker.objects.prefetch_related("hikingroutesaves")

```

```

        authhiker =
authhiker.prefetch_related("hikingroutelikes")
        authhiker = authhiker.filter(useraccount = request.user)
        ownhikregionimpressions =
Hikingregionimpression.objects.filter(hikingregion =
hikingregion_data, hiker = authhiker)
        if ownhikregionimpressions:
            hikingregion_context["ownimpression"] = True
        else:
            hikingregion_context["ownimpression"] = False
        hikingregionpeaks = Peak.objects.filter(hikingregion =
hikingregion_data)
        hikingregionpeaks = hikingregionpeaks.order_by("-
impression")
        hikingregion_context["peaks"] = hikingregionpeaks
        hikingroutes =
Hikingroutestoppeak.objects.select_related("hikingroutestop")
        hikingroutes =
hikingroutes.select_related("hikingroutestop__hikingroute")
        hikingroutes = hikingroutes.filter(peak__in =
hikingregionpeaks)
        hikingroutes =
hikingroutes.annotate(hiker_likes_count=Count('hikerlikes'))
        hikingroutes = hikingroutes.order_by('hiker_likes_count')
        hikingroutes =
hikingroutes.values("hikingroutestop__hikingroute")
        hikingroutes = hikingroutes.distinct()
        hikingroutes = hikingroutes[:8]
        if authhiker:
            for hikingroute in hikingroutes:
                if hikingroute in authhiker.hikingroutesaves:
                    hikingroute.saved = True
                else:
                    hikingroute.saved = False
                if hikingroute in authhiker.hikingroutelikes:
                    hikingroute.liked = True
                else:
                    hikingroute.liked = False
            hikingregion_context["hikingroutes"] = hikingroutes
        return render(request, "hikeliking/hikingregion.html",
hikingregion_context)

```

```

def hikingregionimpressionsview(request, hikingregionId):

```

```

        hikingregion_context = {}
        try:
            hikingregion_data =
Hikingregion.objects.prefetch_related("countryregions")
            hikingregion_data = hikingregion_data.get(id =
hikingregionId)
        except Hikingregion.DoesNotExist:
            return render(request, "hikeliking/error.html",
{"message": "Про такий регіон пішого туризму мені нічого невідомо.
Зв'яжіться з адміністратором, щоб він дав тому раду, якщо він все ж є,
а я про те нічого не знаю. Або вкажіть правильний номер"})
            hikingregion_context["region"] = hikingregion_data
            hikregionimpressions =
Hikingregionimpression.objects.select_related("hiker")
            hikregionimpressions =
hikregionimpressions.filter(hikingregion = hikingregion_data)
            hikregionimpressions = hikregionimpressions.order_by("-
impressiondate")
            for hikregionimpression in hikregionimpressions:
                hikerimages = Hikerimage.objects.filter(hiker =
hikregionimpression.hiker, mainorno = True)
                hikregionimpression.image = hikerimages
                hikingregion_context["impressions"] = hikregionimpressions
                authhiker = None
                if request.user.is_active:
                    authhiker =
Hiker.objects.prefetch_related("hikingroutesaves")
                    authhiker = authhiker.filter(useraccount = request.user)
                    ownhikregionimpressions =
Hikingregionimpression.objects.filter(hikingregion =
hikingregion_data, hiker = authhiker)
                    if ownhikregionimpressions:
                        hikingregion_context["ownimpression"] = True
                    else:
                        hikingregion_context["ownimpression"] = False
            return render(request,
"hikeliking/hikingregionimpressions.html", hikingregion_context)

def peakview(request, peakId):
    peak_context = {}
    try:
        peak_data = Peak.objects.select_related("hikingregion")

```

```

        peak_data =
peak_data.select_related("hikingregion__countryregion")
        peak_data = peak_data.get(id = peakId)
    except Peak.DoesNotExist:
        return render(request, "hikeliking/error.html",
{"message": "Про таку гірську вершину мені нічого невідомо. Зв'яжіться
з адміністратором, щоб він дав тому раду, якщо він все ж є, а я про те
нічого не знаю. Або вкажіть правильний номер"})
        peak_context["peak"] = peak_data
        peakimpressions =
Peakimpression.objects.select_related("hikingregion")
        peakimpressions =
peakimpressions.select_related("hikingregion__countryregion")
        peakimpressions = peakimpressions.filter(peak =
hikingregion_data)
        peakimpressions = peakimpressions.order_by("-
impressiondate")
        for peakimpression in peakimpression:
            hikerimages = Hikerimage.objects.filter(hiker =
peakimpression.hiker, mainorno = True)
            peakimpression.image = hikerimages
            peak_context["impressions"] = peakimpressions[:8]
            authhiker = None
            if request.user.is_active:
                authhiker =
Hiker.objects.prefetch_related("hikingroutesaves")
                authhiker =
authhiker.prefetch_related("hikingroutelikes")
                authhiker = authhiker.filter(useraccount = request.user)
                ownpeakimpressions = Peakimpression.objects.filter(peak
= peak_data, hiker = authhiker)
                if ownpeakimpressions:
                    peak_context["ownimpression"] = True
                else:
                    peak_context["ownimpression"] = False
            hikingroutes =
Hikingroutestoppeak.objects.select_related("hikingroutestop")
            hikingroutes =
hikingroutes.select_related("hikingroutestop__hikingroute")
            hikingroutes = hikingroutes.filter(peak = peak_data)
            hikingroutes =
hikingroutes.annotate(hiker_likes_count=Count('hikerlikes'))
            hikingroutes = hikingroutes.order_by('hiker_likes_count')

```

```

        hikingroutes =
hikingroutes.values("hikingroutestop__hikingroute")
        hikingroutes = hikingroutes.distinct()
        hikingroutes = hikingroutes[:8]
        if authhiker:
            for hikingroute in hikingroutes:
                if hikingroute in authhiker.hikingroutesaves:
                    hikingroute.saved = True
                else:
                    hikingroute.saved = False
                if hikingroute in authhiker.hikingroutelikes:
                    hikingroute.liked = True
                else:
                    hikingroute.liked = False
        peak_context["hikingroutes"] = hikingroutes
        return render(request, "hikeliking/peak.html", peak_context)

def peakimpressionsview(request, peakId):
    peak_context = {}
    try:
        peak_data = Peak.objects.select_related("hikingregion")
        peak_data =
peak_data.select_related("hikingregion__countryregion")
        peak_data = peak_data.get(id = peakId)
    except Peak.DoesNotExist:
        return render(request, "hikeliking/error.html",
{"message": "Про таку гірську вершину мені нічого невідомо. Зв'яжіться
з адміністратором, щоб він дав тому раду, якщо він все ж є, а я про те
нічого не знаю. Або вкажіть правильний номер"})
        peak_context["peak"] = peak_data
        peakimpressions =
Peakimpression.objects.select_related("hikingregion")
        peakimpressions =
peakimpressions.select_related("hikingregion__countryregion")
        peakimpressions = peakimpressions.filter(peak =
hikingregion_data)
        peakimpressions = peakimpressions.order_by("-
impressiondate")
        peakimpressions = peakimpressions.order_by("-
impressiondate")
        for peakimpression in peakimpression:
            hikerimages = Hikerimage.objects.filter(hiker =
peakimpression.hiker, mainorno = True)

```

```

        peakimpression.image = hikerimages
        peak_context["impressions"] = peakimpressions
        authhiker = None
        if request.user.is_active:
            authhiker =
Hiker.objects.prefetch_related("hikingroutesaves")
            authhiker = authhiker.filter(useraccount = request.user)
            ownpeakimpressions = Peakimpression.objects.filter(peak
= peak_data, hiker = authhiker)
            if ownpeakimpressions:
                peak_context["ownimpression"] = True
            else:
                peak_context["ownimpression"] = False
        return render(request, "hikeliking/peakimpressions.html",
peak_context)

def peakratingview(request):
    peaks_context = {}
    peaks_data = Peak.objects.select_related("hikingregion")
    peaks_data =
peaks_data.select_related("hikingregion__countryregion")
    peaks_data = peaks_data.all()
    peaks_data = peaks_data.order_by("-impression")
    peaks_context["peaks"] = peaks_data[:100]
    return render(request, "hikeliking/peakrating.html",
peaks_context)

def hikingregionratingview(request):
    hikingregions_context = {}
    hikingregions_data =
Hikingregion.objects.prefetch_related("countryregions")
    hikingregions_data = hikingregions_data.all()
    hikingregions_data = hikingregions_data.order_by("-
impression")
    hikingregions_context["regions"] = hikingregions_data[:100]
    return render(request, "hikeliking/hikingregionrating.html",
hikingregions_context)

def hikingregionroutesview(request, hikingregionId):
    hikingregion_context = {}
    try:
        hikingregion_data =
Hikingregion.objects.prefetch_related("countryregions")

```

```

        hikingregion_data = hikingregion_data.get(id =
hikingregionId)
        except Hikingregion.DoesNotExist:
            return render(request, "hikeliking/error.html",
{"message": "Про такий регіон пішого туризму мені нічого невідомо.
Зв'яжіться з адміністратором, щоб він дав тому раду, якщо він все ж є,
а я про те нічого не знаю. Або вкажіть правильний номер"})
            hikingregion_context["region"] = hikingregion_data
            hikingroutes =
Hikingroutestoppeak.objects.select_related("hikingroutestop")
            hikingroutes =
hikingroutes.select_related("hikingroutestop__hikingroute")
            hikingroutes = hikingroutes.filter(peak__in =
hikingregionpeaks)
            hikingroutes =
hikingroutes.annotate(hiker_likes_count=Count('hikerlikes'))
            hikingroutes = hikingroutes.order_by('hiker_likes_count')
            hikingroutes =
hikingroutes.values("hikingroutestop__hikingroute")
            hikingroutes = hikingroutes.distinct()
            hikingroutes = hikingroutes
            if request.user.is_active:
                authhiker =
Hiker.objects.prefetch_related("hikingroutesaves")
                authhiker =
authhiker.prefetch_related("hikingroutelikes")
                authhiker = authhiker.filter(useraccount = request.user)
                for hikingroute in hikingroutes:
                    if hikingroute in authhiker.hikingroutesaves:
                        hikingroute.saved = True
                    else:
                        hikingroute.saved = False
                    if hikingroute in authhiker.hikingroutelikes:
                        hikingroute.liked = True
                    else:
                        hikingroute.liked = False
                hikingregion_context["hikingroutes"] = hikingroutes
                return render(request, "hikeliking/hikingregionroutes.html",
hikingregion_context)

def hikeroutesview(request, hikerName = None):
    hikingroutes_context = {}
    if not request.user.is_active:

```

```

        authhiker =
Hiker.objects.prefetch_related("hikingroutesaves")
        authhiker =
authhiker.prefetch_related("hikingroutelikes")
        authhiker = authhiker.filter(useraccount__username =
hikerName)
    else:
        authhiker =
Hiker.objects.prefetch_related("hikingroutesaves")
        authhiker =
authhiker.prefetch_related("hikingroutelikes")
        authhiker = authhiker.filter(useraccount = request.user)
        peakimpressions = peakimpressions.order_by("-
impressiondate")
        hikerimages = Hikerimage.objects.filter(hiker = authhiker,
mainorno = True)
        authhiker.image = hikerimages
        hikingroutes =
Hikingroutestoppeak.objects.select_related("hikingroutestop")
        hikingroutes =
hikingroutes.select_related("hikingroutestop__hikingroute")
        hikingroutes = hikingroutes.filter(authorhiker = authhiker)
        hikingroutes =
hikingroutes.annotate(hiker_likes_count=Count('hikerlikes'))
        hikingroutes = hikingroutes.order_by('hiker_likes_count')
        hikingroutes =
hikingroutes.values("hikingroutestop__hikingroute")
        hikingroutes = hikingroutes.distinct()
        if hikerName:
            authhiker =
Hiker.objects.prefetch_related("hikingroutesaves")
            authhiker = authhiker.filter(useraccount = request.user)
            for hikingroute in hikingroutes:
                if hikingroute in authhiker.hikingroutesaves:
                    hikingroute.saved = True
                else:
                    hikingroute.saved = False
                if hikingroute in authhiker.hikingroutelikes:
                    hikingroute.liked = True
                else:
                    hikingroute.liked = False
            hikingroutes_context["hikingroutes"] = hikingroutes

```

```

        return render(request, "hikeliking/hikingroutes.html",
hikingroutes_context)

    def commentedroutesview(request):
        hikingroutes_context = {}
        if not request.user.is_active:
            return render(request, "hikeliking/error.html",
{"message": "Ця функція вимагає авторизації. Якщо ви її виконували і
не виходили, то зв'яжіться з адміністратором, щоб він дав тому раду"})
        else:
            authhiker =
Hiker.objects.prefetch_related("hikingroutesaves")
            authhiker =
authhiker.prefetch_related("hikingroutelikes")
            authhiker = authhiker.filter(useraccount = request.user)
            routecomments =
Hikingroutecomment.objects.select_related("hikingroute")
            routecomments = routecomments.filter(hiker = authhiker)
            routecomments = routecomments.order_by("-commenttime")
            for routecomment in routecomments:
                routecomment.image = Hikerimage.objects.filter(hiker =
routecomment.hiker, mainorno = True)
                hikingroutes_commented =
routecomments.values("hikingroute__id")
                hikingroutes =
Hikingroute.objects.select_related("authorhiker")
                hikingroutes = hikingroutes.filter(id__in =
hikingroutes_commented)
                if authhiker:
                    for hikingroute in hikingroutes:
                        if hikingroute in authhiker.hikingroutesaves:
                            hikingroute.saved = True
                        else:
                            hikingroute.saved = False
                        if hikingroute in authhiker.hikingroutelikes:
                            hikingroute.liked = True
                        else:
                            hikingroute.liked = False
            hikingroutes_context["hikingroutes"] = hikingroutes
            return render(request,
"hikeliking/commentedhikingroutes.html", hikingroutes_context)

    def attentionhikingplansview(request):

```

```

        hikingplans_context = {}
        if not request.user.is_active:
            return render(request, "hikeliking/error.html",
{"message": "Ця функція вимагає авторизації. Якщо ви її виконували і не виходили, то зв'яжіться з адміністратором, щоб він дав тому раду"})
        else:
            authhiker =
Hiker.objects.prefetch_related("hikerattenteds")
            authhiker = authhiker.filter(useraccount = request.user)
            hikingplans = Hikingplan.select_related("hiker")
            hikingplans = hikingplans.filter(hiker__in =
authhiker.hikerattenteds)
            hikingplans = hikingplans.order_by("-hikingstart")
            hikingplans_context["plans"] = hikingplans
            return render(request,
"hikeliking/attentionhikingplans.html", hikingroutes_context)

import datetime

def futurehikingplansview(request, hikerName = None):
    hikingplans_context = {}
    if not request.user.is_active:
        authhiker = Hiker.objects.filter(useraccount__username =
hikerName)
    else:
        authhiker = Hiker.objects.filter(useraccount =
request.user)
        hikingplans = Hikingplan.select_related("hiker")
        hikingplans = hikingplans.filter(hiker = authhiker)
        future_point = datetime.datetime.now()
        hikingplans = hikingplans.filter(hikingstart__gt =
future_point)
        hikingplans = hikingplans.order_by("-hikingstart")
        hikingplans_context["plans"] = hikingplans
        return render(request, "hikeliking/futurehikingplans.html",
hikingplans_context)

def historyhikingplansview(request, hikerName = None):
    hikingplans_context = {}
    if not request.user.is_active:
        authhiker = Hiker.objects.filter(useraccount__username =
hikerName)
    else:

```

```

        authhiker = Hiker.objects.filter(useraccount =
request.user)
        hikingplans = Hikingplan.select_related("hiker")
        hikingplans = hikingplans.filter(hiker = authhiker)
        future_point = datetime.datetime.now()
        hikingplans = hikingplans.filter(hikingend__lt =
future_point)
        hikingplans = hikingplans.order_by("hikingstart")
        hikingplans_context["plans"] = hikingplans
        return render(request, "hikeliking/historyhikingplans.html",
hikingplans_context)

def hikerimpressionsview(request, hikerName = None):
    hikingimpressions_context = {}
    if not request.user.is_active:
        authhiker =
Hiker.objects.prefetch_related("hikingroutesaves")
        authhiker = authhiker.filter(useraccount__username =
hikerName)
    else:
        authhiker =
Hiker.objects.prefetch_related("hikingroutesaves")
        authhiker = authhiker.filter(useraccount = request.user)
        hikerimages = Hikerimage.objects.filter(hiker = authhiker,
mainorno = True)
        authhiker.image = hikerimages
        hikerimpressions =
Hikerimpression.objects.select_related("hiker")
        hikerimpressions =
hikerimpressions.select_related("hikerimpressed")
        hikerimpressions = hikerimpressions.filter(hikerimpressed =
authhiker)
        hikerimpressions = hikerimpressions.order_by("-
impressiondate")
        for hikerimpression in hikerimpressions:
            hikerimages = Hikerimage.objects.filter(hiker =
hikerimpression.hiker, mainorno = True)
            hikerimages.image = hikerimages
            hikerimpressions = hikerimpressions[:8]
            hikingimpressions_context["impressed"] = hikerimpressions
            hikerimpressions =
Hikerimpression.objects.select_related("hiker")

```

```

        hikerimpressions =
hikerimpressions.select_related("hikerimpressed")
        hikerimpressions = hikerimpressions.filter(hiker =
authhiker)
        hikerimpressions = hikerimpressions.order_by("-
impressiondate")
        for hikerimpression in hikerimpressions:
            hikerimages = Hikerimage.objects.filter(hiker =
hikerimpression.hikerimpressed, mainorno = True)
            hikerimages.image = hikerimages
            hikerimpressions = hikerimpressions[:8]
            hikingimpressions_context["hiker"] = hikerimpressions
            hikerimpressions =
Hikingregionimpression.objects.select_related("hiker")
            hikerimpressions =
hikerimpressions.select_related("hikingregion")
            hikerimpressions = hikerimpressions.filter(hiker =
authhiker)
            hikerimpressions = hikerimpressions.order_by("-
impressiondate")
            for hikerimpression in hikerimpressions:
                hikerimages = Hikerimage.objects.filter(hiker =
hikerimpression.hiker, mainorno = True)
                hikerimages.image = hikerimages
                hikerimpressions = hikerimpressions[:8]
                hikingimpressions_context["region"] = hikerimpressions
                hikerimpressions =
Peakimpression.objects.select_related("hiker")
                hikerimpressions = hikerimpressions.select_related("peak")
                hikerimpressions = hikerimpressions.filter(hiker =
authhiker)
                hikerimpressions = hikerimpressions.order_by("-
impressiondate")
                for hikerimpression in hikerimpressions:
                    hikerimages = Hikerimage.objects.filter(hiker =
hikerimpression.hiker, mainorno = True)
                    hikerimages.image = hikerimages
                    hikerimpressions = hikerimpressions[:8]
                    hikingimpressions_context["peak"] = hikerimpressions
                    return render(request, "hikeliking/hikerimpressions.html",
hikingimpressions_context)

def abouthikerimpressionsview(request, hikerName = None):

```

```

        hikingimpressions_context = {}
        if not request.user.is_active:
            authhiker =
Hiker.objects.prefetch_related("hikingroutesaves")
            authhiker = authhiker.filter(useraccount__username =
hikerName)
        else:
            authhiker =
Hiker.objects.prefetch_related("hikingroutesaves")
            authhiker = authhiker.filter(useraccount = request.user)
            hikerimpressions =
Hikerimpression.objects.select_related("hiker")
            hikerimpressions =
hikerimpressions.select_related("hikerimpressed")
            hikerimpressions = hikerimpressions.filter(hiker =
authhiker)
            hikerimpressions = hikerimpressions.order_by("-
impressiondate")
            hikerimpressions = hikerimpressions
            hikingimpressions_context["hiker"] = hikerimpressions
            return render(request,
"hikeliking/abouthikerimpressions.html", hikingimpressions_context)

    def hikerprofileview(request, hikerName = None):
        hiker_context = {}
        if not request.user.is_active:
            authhiker =
Hiker.objects.prefetch_related("hikingroutesaves")
            authhiker =
authhiker.prefetch_related("hikingroutelikes")
            authhiker = authhiker.select_related("countryregion")
            authhiker = authhiker.filter(useraccount__username =
hikerName)
        else:
            authhiker =
Hiker.objects.prefetch_related("hikingroutesaves")
            authhiker =
authhiker.prefetch_related("hikingroutelikes")
            authhiker = authhiker.select_related("countryregion")
            authhiker = authhiker.filter(useraccount = request.user)
            authhiker.image = Hikerimage.objects.filter(hiker =
authhiker)
            hiker_context["hiker"] = authhiker

```

```

        hikerimpressions =
Hikerimpression.objects.select_related("hiker")
        hikerimpressions =
hikerimpressions.select_related("hikerimpressed")
        hikerimpressions = hikerimpressions.filter(hikerimpressed =
authhiker)
        hikerimpressions = hikerimpressions.order_by("-
impressiondate")
        hikerimpressions = hikerimpressions[:8]
        hiker_context["impressed"] = hikerimpressions
        hikerimpressions =
Hikerimpression.objects.select_related("hiker")
        hikerimpressions =
hikerimpressions.select_related("hikerimpressed")
        hikerimpressions = hikerimpressions.filter(hiker =
authhiker)
        hikerimpressions = hikerimpressions.order_by("-
impressiondate")
        hikerimpressions = hikerimpressions[:8]
        hiker_context["hiker"] = hikerimpressions
        hikerimpressions =
Hikingregionimpression.objects.select_related("hiker")
        hikerimpressions =
hikerimpressions.select_related("hikingregion")
        hikerimpressions = hikerimpressions.filter(hiker =
authhiker)
        hikerimpressions = hikerimpressions.order_by("-
impressiondate")
        hikerimpressions = hikerimpressions[:8]
        hiker_context["region"] = hikerimpressions
        hikerimpressions =
Peakimpression.objects.select_related("hiker")
        hikerimpressions = hikerimpressions.select_related("peak")
        hikerimpressions = hikerimpressions.filter(hiker =
authhiker)
        hikerimpressions = hikerimpressions.order_by("-
impressiondate")
        hikerimpressions = hikerimpressions[:8]
        hikingplans = Hikingplan.select_related("hiker")
        hikingplans = hikingplans.filter(hiker = authhiker)
        future_point = datetime.datetime.now()
        hikingplans = hikingplans.filter(hikingstart__gt =
future_point)

```

```

        hikingplans = hikingplans.order_by("-hikingstart")
        hiker_context["futureplans"] = hikingplans
        hikingplans = Hikingplan.select_related("hiker")
        hikingplans = hikingplans.filter(hiker = authhiker)
        future_point = datetime.datetime.now()
        hikingplans = hikingplans.filter(hikingstart__lt =
future_point)
        hikingplans = hikingplans.order_by("hikingstart")
        hiker_context["oldplans"] = hikingplans
        hikingroutes =
Hikingroutestoppeak.objects.select_related("hikingroutestop")
        hikingroutes =
hikingroutes.select_related("hikingroutestop__hikingroute")
        hikingroutes = hikingroutes.filter(authorhiker = authhiker)
        hikingroutes =
hikingroutes.annotate(hiker_likes_count=Count('hikerlikes'))
        hikingroutes = hikingroutes.order_by('hiker_likes_count')
        hikingroutes =
hikingroutes.values("hikingroutestop__hikingroute")
        hikingroutes = hikingroutes.distinct()
        if hikerName:
            for hikingroute in hikingroutes:
                if hikingroute in authhiker.hikingroutesaves:
                    hikingroute.saved = True
                else:
                    hikingroute.saved = False
                if hikingroute in authhiker.hikingroutelikes:
                    hikingroute.liked = True
                else:
                    hikingroute.liked = False
            hikingroutes_context["hikingroutes"] = hikingroutes
        return render(request, "hikeliking/hiker.html",
hiker_context)

def hikingrouteview(request, hikingrouteId):
    hikingroute_context = {}
    try:
        hikingroute_data =
Hikingroute.objects.prefetch_related("authorhiker")
        hikingroute_data = hikingroute_data.get(id =
hikingrouteId)
        hikingroute_data =
hikingroute_data.annotate(hiker_likes_count=Count('hikerlikes'))

```

```

except Hikingroute.DoesNotExist:
    return render(request, "hikeliking/error.html",
{"message": "Про такий маршрут походу пішого туризму мені нічого
невідомо. Зв'яжіться з адміністратором, щоб він дав тому раду, якщо
він все ж є, а я про те нічого не знаю. Або вкажіть правильний
номер"})

    hikerimages = Hikerimage.objects.filter(hiker =
hikingroute_data.authorhiker, mainorno = True)
    hikingroute_data.images = hikerimages
    hikingroutestops =
Hikingroutestop.objects.filter(hikingroute = hikingroute_data)
    for hikingroutestop in hikingroutestops:
        hikingroutepeaks =
Hikingroutestoppeak.objects.select_related("peak")
        hikingroutepeaks =
hikingroutepeaks.filter(hikingroutestop = hikingroutestop)
        hikingroutestop.peaks = hikingroutepeaks
        hikingroute_context["stops"] = hikingroutestops
        routecomments =
Hikingroutecomment.objects.select_related("hiker")
        routecomments = routecomments.filter(hikingroute =
hikingroute_data)
        routecomments = routecomments.order_by("-commenttime")
        hikingroute_context["comments"] = routecomments
        if request.user.is_active:
            authhiker =
Hiker.objects.prefetch_related("hikingroutesaves")
            authhiker =
authhiker.prefetch_related("hikingroutelikes")
            authhiker = authhiker.filter(useraccount = request.user)
            if hikingroute_data in authhiker.hikingroutesaves:
                hikingroute_data.save = True
            else:
                hikingroute_data.save = False
            if hikingroute_data in authhiker.hikingroutelikes:
                hikingroute_data.like = True
            else:
                hikingroute_data.like = False
            if hikingroute_data in authhiker.hikingroutelikes:
                hikingroute_data.liked = True
            else:
                hikingroute_data.liked = False
        hikingroute_context["route"] = hikingroute_data

```

```

        return render(request, "hikeliking/hikingroute.html",
hikingroute_context)

```

```

def hikerratingview(request):
    hiker_context = {}
    hikers_data = Hiker.objects.select_related("countryregion")
    hikers_data = hikers_data.all()
    hikers_data = hikers_data.order_by("-impression")
    hikers_data = hikers_data[:100]
    for hikerdata in hikers_data:
        hikerimage = Hikerimage.objects.filter(hiker =
hikerdata, mainorno = True)
        hikerdata.image = hikerimage
    hiker_context["hikers"] = hikers_data
    return render(request, "hikeliking/hikerrating.html",
hiker_context)

```

ДОДАТОК В
Слайди презентації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»
КАФЕДРА ПРОГРАМНИХ ЗАСОБІВ

ТЕМА ДИПЛОМНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА:
**ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ
ЗНАХОДЖЕННЯ ОДНОДУМЦІВ ДЛЯ
ПІШОГО ТУРИЗМУ. SOFTWARE TO SEARCH
FOR LIKE-MINDED PEOPLE FOR HIKING**

Виконавець:

Олег ДУБИНА

Керівник: к.т.н., доцент

Степан СКРУПСЬКИЙ

2024

Рисунок В.1 – Перший слайд

ОБ'ЄКТ, ПРЕДМЕТ, МЕТА РОБОТИ

- об'єкт роботи – процес планування пішого походу;
- предмет роботи – програмне забезпечення знаходження однодумців для пішого туризму;
- метою роботи є розробка програмного забезпечення знаходження однодумців для пішого туризму, щоби сприяти підвищенню взаємодії в цій галузі та розвитку галузі загалом.

2

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

ЗАВДАННЯ РОБОТИ

- проаналізувати програмне забезпечення, придатне для знаходження однодумців для пішого туризму;
- визначити засоби проектування програми;
- виділити вимоги до програми знаходження однодумців для пішого туризму;
- виконати реалізацію програми знаходження однодумців для пішого туризму.

3

Рисунок В.3 – Завдання роботи

ПОРІВНЯННЯ ПРОГРАМ ДЛЯ ЗНАХОДЖЕННЯ ОДНОДУМЦІВ ДЛЯ ПІШОГО ТУРИЗМУ

Критерій	Meetup	Social Hikers	Розробка
Визначення походів	Як окрема подія з можливим текстовим описом маршруту	За декількома визначеними у програмі маршрутами	За маршрутами, які створюють самі користувачі
Відображення маршруту на мапі	Немає	Немає	Є
Описи минулих походів	Ні	Так	Так
Можливість приєднатися до нового походу	Так	Ні	Так
Можливість комунікувати з любителями пішого туризму	Так	Так	Так
Висловлення вражень	Про подію (похід)	Про похід	Про похід, про вершини, про регіони

4

Рисунок В.4 – Порівняння програм для знаходження однодумців для пішого туризму

ДІАГРАМА ПРЕЦЕДЕНТІВ ПРОГРАМИ ЗНАХОДЖЕН- НЯ ОДНОДУМЦІВ ДЛЯ ПІШОГО ТУРИЗМУ

Рисунок В.6 – Діаграма прецедентів програми знаходження однодумців для пішого туризму

Рисунок В.7 – Схема бази даних програми знаходження однодумців для пішого туризму



Рисунок В.8 – Функціональна схема програми знаходження одностумців для пішого туризму

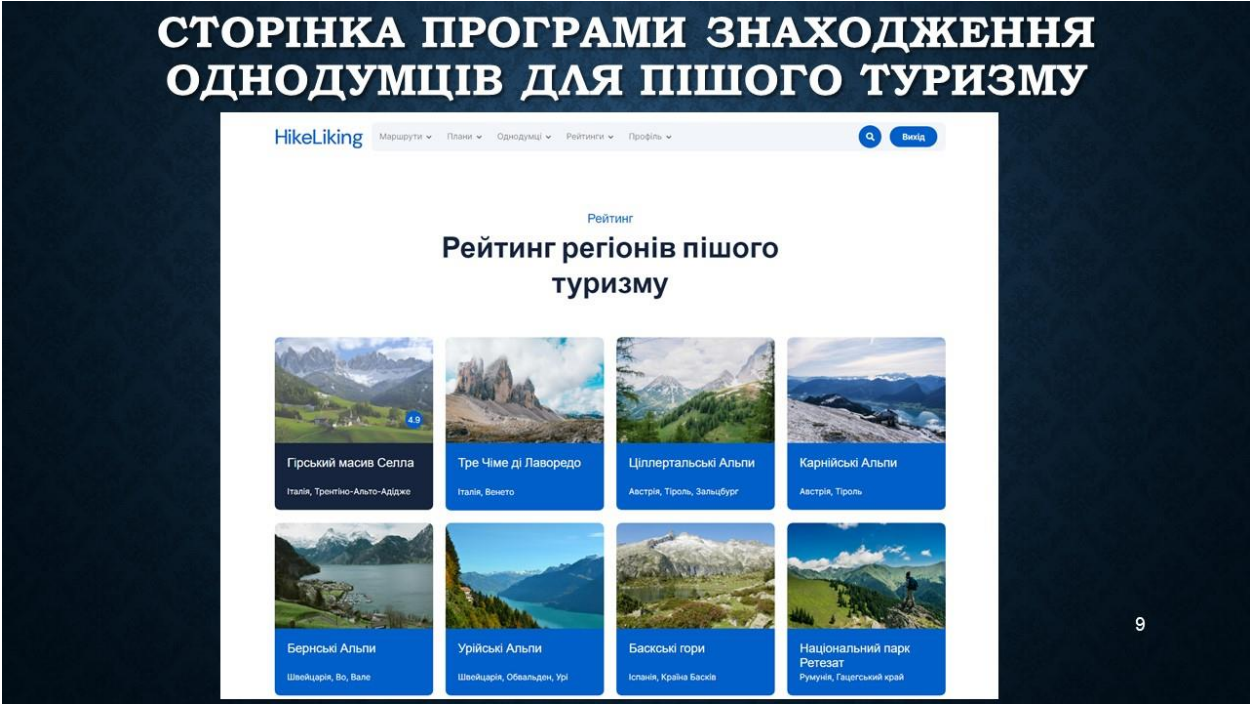


Рисунок В.9 – Сторінка програми знаходження однодумців для пішого туризму

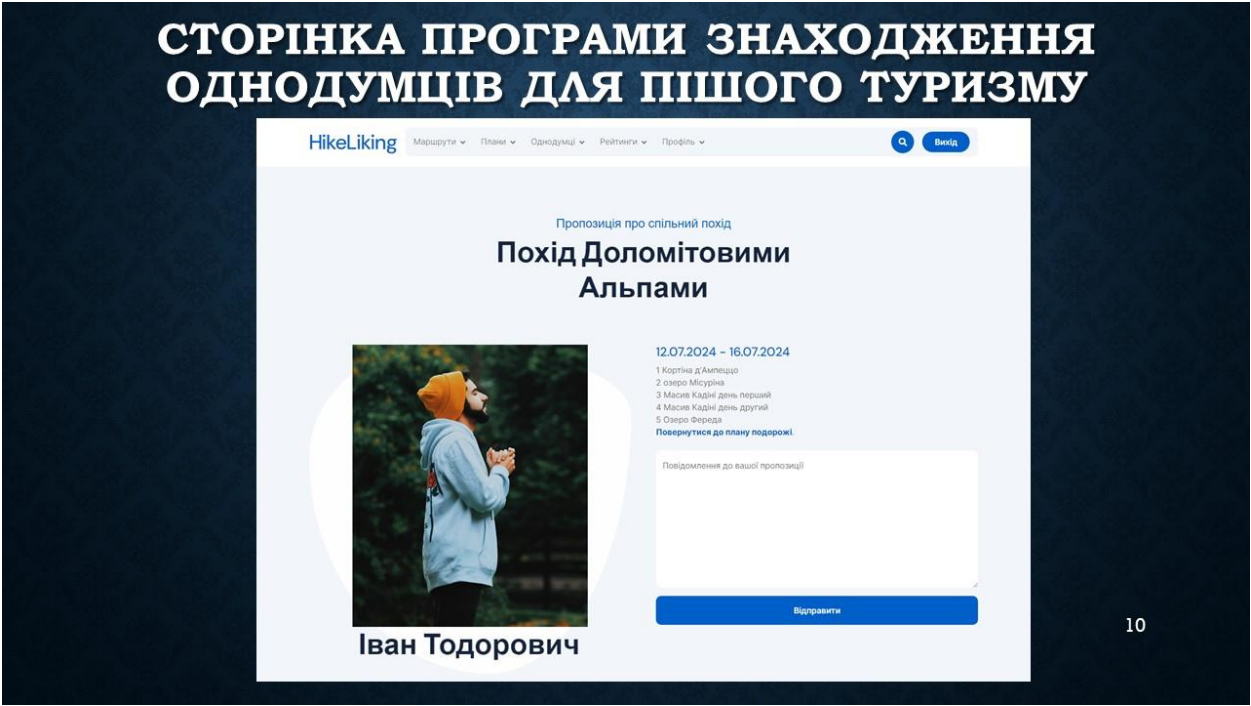


Рисунок В.10 – Сторінка програми знаходження однодумців для пішого туризму

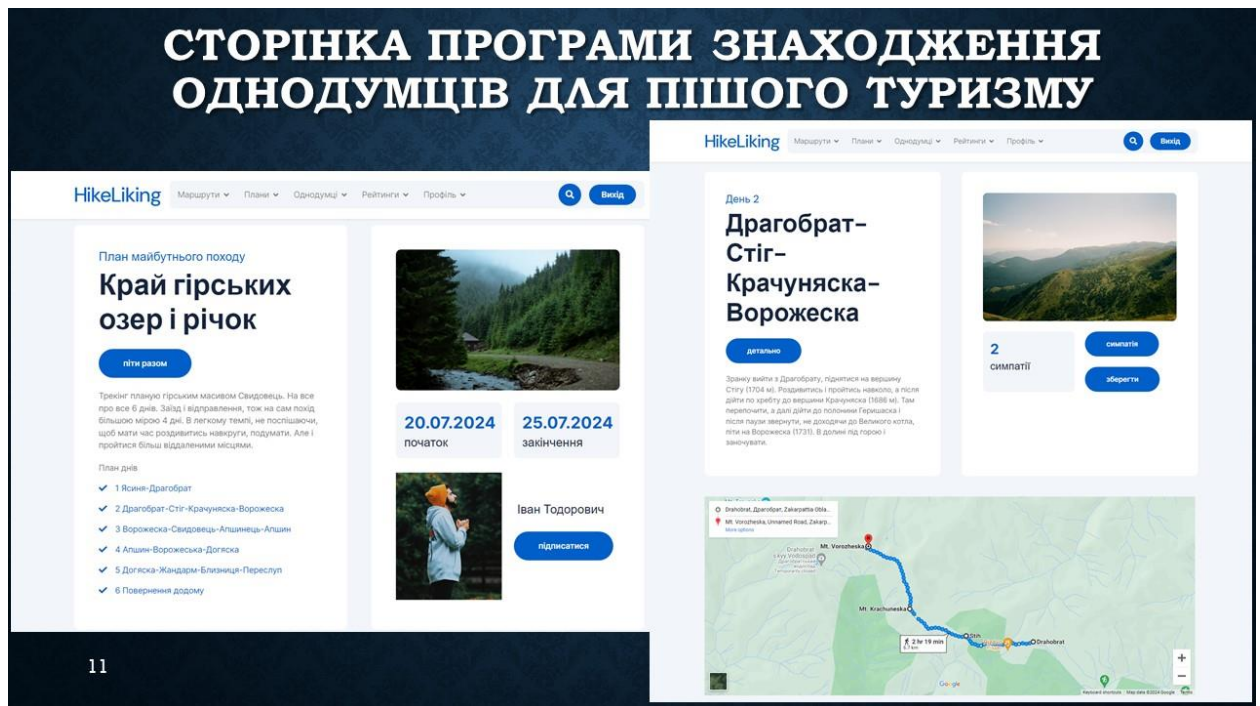


Рисунок В.11 – Сторінка програми знаходження однодумців для пішого туризму

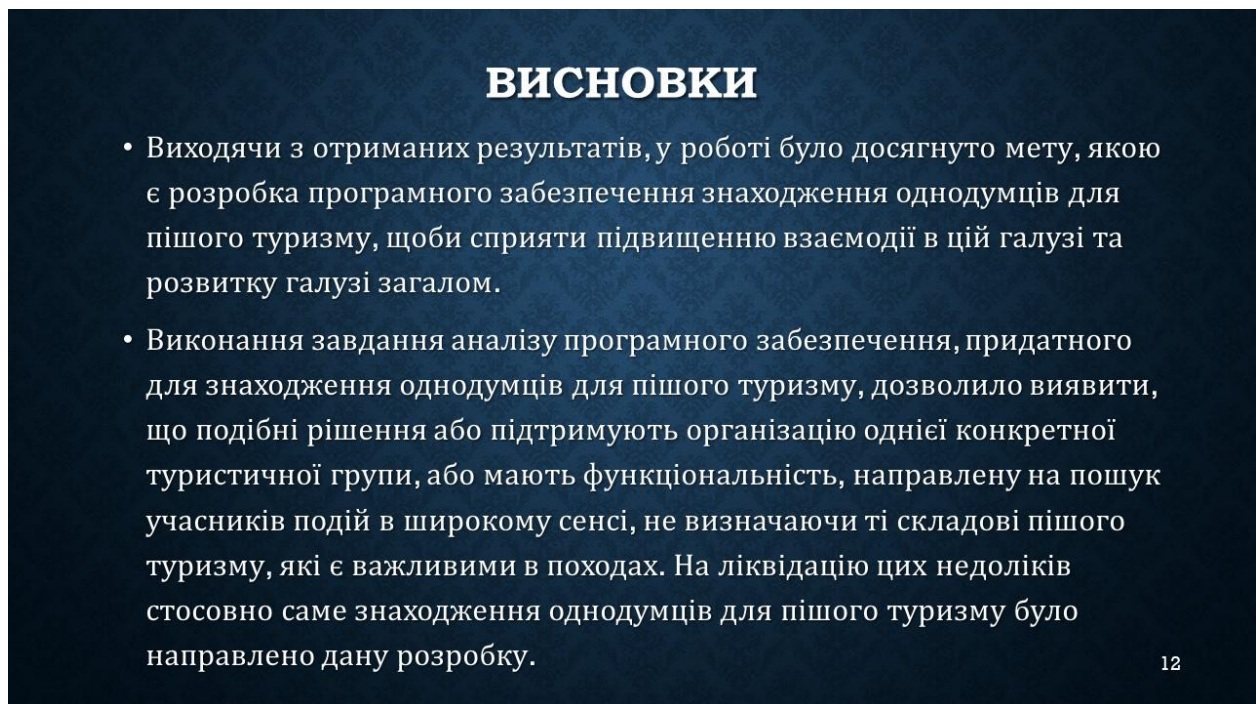


Рисунок В.12 – Висновки