ЛЬВІВСЬКИЙ ТОРГОВЕЛЬНО-ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

Факультет міжнародних економічних відносин та інформаційних технологій

Кафедра комп'ютерних наук, прикладної та вищої математики

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Комп’ютерна підтримка розробки web-сайту знайомств

Виконав: здобувач вищої освіти першого (бакалаврського) рівня
ОПП "Комп'ютерні науки"
спеціальності 122 "Комп'ютерні науки"
денної форми навчання

_____ Козоріс Олексій Павлович

(підпис)

Науковий керівник
_____ к. е. н. Папка О.С.

(підпис)

Львів – 2025

ЛЬВІВСЬКИЙ ТОРГОВЕЛЬНО-ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

Факультет міжнародних економічних відносин та інформаційних технологій
Кафедра комп'ютерних наук, прикладної та вищої математики
Перший (бакалаврський) рівень вищої освіти
Спеціальність 122 "Комп'ютерні науки"
Освітньо-професійна програма "Комп'ютерні науки"

“ЗАТВЕРДЖУЮ”

Завідувач кафедри комп'ютерних наук,
прикладної та вищої математики

_____ З. Г. Новосад

28.10.2024 р.

Завдання
до кваліфікаційної роботи
Козорісу Олексію Павловичу
(прізвище, ім’я, по батькові)

1.Тема роботи: Комп’ютерна підтримка розробки web-сайту знайомств

керівник роботи Папка Олег Степанович, к. е. н.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджена наказом по Університету №1041/03 від "28" жовтня 2024 р.

2.Термін здачі здобувачем вищої освіти роботи 5 червня 2025 р.
3. Вхідні дані до роботи література, документація, матеріали та сайти Internet, що пов’язані з проектуванням та розробкою web-сайту знайомств
4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)
1. Дослідження предметної області онлайн-знайомств.
2. Аналіз засобів реалізації web-сайту.
3. Проектування та розробка web-сайтів знайомств.

5. Перелік графічного матеріалу до 12 слайдів у форматі PowerPoint
6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання 28.10. 2024 р.

Календарний план

№	Найменування етапів виконання кваліфікаційної роботи	Терміни виконання етапів	Примітка
1	Отримання завдання і аналіз інформаційних джерел за темою роботи	до 11.12.2024	виконано
2	Дослідження предметної області онлайн-знайомств	до 13.01.2025	виконано
3	Аналіз засобів реалізації web-сайту	до 07.02.2025	виконано
4	Проектування та реалізація бази даних для збереження інформації	до 01.04.2025	виконано
5	Проектування та реалізація web-сайту	до 15.04.2025	виконано
6	Тестування розробленого web-сайту	до 22.04.2025	виконано
7	Оформлення кваліфікаційної роботи. Подання роботи на перевірку керівнику	до 16.05.2025	виконано
8	Доопрацювання кваліфікаційної роботи з врахуванням зауважень керівника. Подання роботи для перевірки на рівень запозичень	до 20.05.2025	виконано
9	Подання кваліфікаційної роботи керівнику для отримання відгуку	до 27.05.2025	виконано
10	Представлення кваліфікаційної роботи на зовнішнє рецензування. Подання кваліфікаційної роботи на кафедру комп'ютерних наук	до 5.06.2025	виконано
11	Попередній захист кваліфікаційної роботи	05.06.2025-06.06.2025	виконано
12	Захист кваліфікаційної роботи на засіданні екзаменаційної комісії	згідно графіку роботи ЕК	

Здобувач вищої освіти _____ Козоріс О.П.

Керівник роботи _____ Папка О. С.

Реферат

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків та списку з використаних джерел. Основний зміст викладено на 57 сторінках і містить 8 рисунків та 2 додатки.

Метою цієї роботи є аналіз і реалізація простого у використанні веб-додатку для знайомств, що дозволить користувачам взаємодіяти між собою, створювати профілі, знаходити потенційних партнерів та обмінюватися повідомленнями. Для досягнення цієї мети буде проведений детальний аналіз засобів розробки веб-додатків, вибрано оптимальні технології та реалізовано базові функції сервісу знайомств.

Ключові слова: сайт знайомств, веб-розробка, Django, база даних, онлайн-комунікація, інтерфейс, безпека

Abstract

The qualification work consists of an introduction, three chapters, conclusions and a list of references. The main content is set out on 57 pages and contains 8 figures and 2 appendices.

The purpose of this work is to analyze and implement an easy-to-use web-based dating application that will allow users to interact with each other, create profiles, find potential partners, and exchange messages. To achieve this goal, a detailed analysis of web application development tools will be conducted, optimal technologies will be selected, and the basic functions of the dating service will be implemented.

Keywords: dating site, web development, Django, database, online communication, interface, security

ЗМІСТ

ВСТУП 7
1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ОНЛАЙН-ЗНАЙОМСТВ 9
1.1 Огляд ринку онлайн-знайомств 9
1.2 Особливості та специфіка роботи сайтів знайомств 10
1.3 Вимоги до сучасних web-сайтів для знайомств 13
2 АНАЛІЗ ЗАСОБІВ РЕАЛІЗАЦІЇ WEB-САЙТУ 15
2.1 Засоби розробки серверної частини (Backend) 15
2.2 Засоби розробки клієнтської частини (Frontend): HTML, CSS, Bootstrap . 16
2.3 Організація зберігання та обробки даних засобами СУБД SQLite 17
2.4 Вибір хмарної платформи для хостингу web-сайту – порівняння платформ Railway, Render, Vercel 19
3 ПРОЕКТУВАННЯ ТА РОЗРОБКА WEB-САЙТІВ ЗНАЙОМСТВ 22
3.1 Проектування структури сайту 22
3.2 Реалізація функціоналу створення профілю та пошуку партнерів 32
3.3 Архітектура серверної частини 37
3.4 Тестування функціоналу та безпеки системи 48
ВИСНОВКИ 55
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 56
ДОДАТКИ 58

ВСТУП

У сучасному світі інформаційні технології відіграють ключову роль у повсякденному житті людини, спрощуючи комунікацію, надаючи доступ до інформації та відкриваючи нові можливості для взаємодії між людьми. Одним із найпопулярніших напрямів використання цифрових технологій є онлайн-знайомства, що стали важливим соціальним явищем. Завдяки розвитку веб-технологій та зростанню популярності мобільних пристроїв, процес знайомства значно змінився, перетворившись у зручний та доступний інструмент для пошуку партнерів.

Розвиток інтернет-знайомств сприяв появі численних платформ, які надають користувачам можливість знайти людину зі схожими інтересами, цінностями та життєвими цілями. Однак успішність таких сервісів залежить від ефективності програмного забезпечення, що забезпечує швидку роботу, захист даних та комфортне використання. Враховуючи це, розробка інформаційної системи для створення сайту знайомств є актуальним завданням, яке потребує дослідження сучасних технологій та методів веб-розробки.

Актуальність теми дослідження зумовлена зростаючим попитом на онлайн-комунікацію та потребою користувачів у зручних, безпечних і доступних інструментах для пошуку нових знайомств. У сучасних умовах, коли соціальні взаємодії все частіше переносяться у цифрове середовище, розробка ефективних веб-сервісів для знайомств набуває особливої значущості.

Об’єктом дослідження є процес взаємодії між користувачами в цифровому середовищі з використанням сучасних веб-технологій.

Предметом дослідження є веб-ресурс — сайт знайомств, призначений для комунікації користувачів, його структура, функціональні можливості та вплив на ефективність цифрової взаємодії.

Метою кваліфікаційної роботи є розробка зручного та безпечного веб-ресурсу для онлайн-знайомств, який сприятиме покращенню взаємодії між

користувачами, забезпеченню конфіденційності, простоти використання та привабливого інтерфейсу.

У роботі розглянуто основні аспекти веб-розробки, зокрема використання серверної частини на основі фреймворку Django, організацію бази даних за допомогою SQLite, а також створення клієнтської частини з використанням HTML, CSS та Bootstrap. Окрім того, досліджується можливість розгортання проєкту на популярних хмарних платформах, таких як Railway, Render та Vercel, з метою вибору оптимального рішення для забезпечення стабільності та масштабованості сервісу.

Практична цінність роботи полягає у розробці веб-додатку, який може бути використаний як основа для подальшого вдосконалення та розширення функціоналу. Крім того, отримані результати можуть бути корисними для студентів та початківців у сфері веб-розробки, оскільки вони демонструють покроковий підхід до створення інформаційної системи з використанням сучасних технологій.

Структура дипломної роботи складається з трьох розділів. Перший розділ містить огляд літератури та аналіз існуючих рішень у сфері онлайн-знайомств. Другий розділ присвячений аналізу засобів реалізації веб-сайту та вибору відповідних технологій. Третій розділ охоплює процес розробки та тестування сайту, а завершальний розділ містить висновки та перспективи подальшого розвитку проєкту.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ОНЛАЙН-ЗНАЙОМСТВ

1.1 Огляд ринку онлайн-знайомств

Ринок онлайн-знайомств за останні десятиліття зазнав значного зростання, перетворившись на багатомільярдну індустрію. За даними дослідження, обсяг світового ринку онлайн-знайомств у 2024 році оцінюється в 5,64 мільярда доларів США, і прогнозується, що до 2035 року він досягне 12,0 мільярда доларів США, демонструючи середньорічний темп зростання (CAGR) у 7,11% протягом цього періоду [16], [17].

Популярність онлайн-знайомств зумовлена низкою факторів, серед яких зростання використання смартфонів, підвищення доступності інтернету та зміна соціальних норм щодо знайомств в інтернеті. Згідно з даними, у 2021 році кількість активних користувачів онлайн-знайомств у світі становила 323 мільйони і очікується, що до 2030 року ця цифра зросте до 500 мільйонів. [17].

Серед провідних гравців ринку можна виділити такі платформи, як Tinder, Bumble, Badoo та інші. Наприклад, Tinder у 2023 році мав середню базу передплатників у 10,4 мільйона, що свідчить про його значну популярність серед користувачів. [17].

Варто зазначити, що ринок онлайн-знайомств продовжує розвиватися, адаптуючись до нових технологій та змін у поведінці користувачів. Зокрема, зростає популярність нішевих платформ, орієнтованих на специфічні інтереси або демографічні групи, а також впровадження нових функцій, таких як відеочати та алгоритми на основі штучного інтелекту для покращення підбору партнерів.

Очікується, що світовий ринок послуг онлайн-знайомств зростатиме із середньорічним темпом зростання 7,89% протягом прогнозованого періоду (2021-2027), досягнувши 6,447.72 мільйона доларів США до 2027 року з 4,086.17 мільйона доларів США у 2021 році [16]. Ринковий попит еволюціонував від його історичної моделі випадкового свайпа на основі фотографій і біографії до

потреби в більш привабливих функціях, які дають змогу встановлювати людські зв'язки через додатки. Сервіси онлайн-знайомств рухаються в бік включення аудіо та відео в додаток, щоб забезпечити більшу надійність і зв'язок. Люди в усьому світі шукають у своїх партнерах певні характеристики. До таких характеристик належать схожі інтереси, однодумці та інші. Онлайн-знайомства допомагають їм знайти людей, які поділяють ці характеристики, тому онлайн-знайомства стають дедалі популярнішими. Очікується, що ринок стане свідком появи нових застосунків із диверсифікацією алгоритму угруповання для націлювання людей на основі їхнього віку, сексуальної орієнтації, етнічної приналежності та інших показників, а також стане інклюзивнішим, даючи змогу представляти всіх своїх користувачів.

Онлайн-знайомства прості у використанні, зручні, швидкі і вимагають менше зусиль. Крім того, це дає змогу обмежити кількість людей, які можуть зв'язатися з ними, використовуючи різні доступні функції. У результаті він широко використовується і сприяє зростанню ринку. Ринок послуг онлайн-знайомств був переважно популярний серед розвинених країн, таких як Сполучені Штати, але стає все більш актуальним і в інших регіонах. Ринок став свідком швидкого впровадження в країнах, що розвиваються, з легким доступом до смартфонів та Інтернету.

Пандемія COVID-19 збільшила кількість людей, які використовують платформи онлайн-знайомств по всьому світу. Уряди запровадили обмеження на поїздки в декількох країнах, а також закриття магазинів, торгових центрів і кафе сприяли глобальному впровадженню платформ онлайн-знайомств.

1.2 Особливості та специфіка роботи сайтів знайомств

Коли мова заходить про сайти знайомств, варто розуміти, що вони значно відрізняються від звичайних соціальних мереж. Якщо Facebook або Instagram створені в першу чергу для обміну інформацією та підтримки зв’язків, то сайти

знайомств мають чітко визначену мету – допомогти користувачам знайти партнера. Саме тому їхня робота ґрунтується на зовсім інших принципах.

Сайти знайомств у сучасному світі стали невід'ємною частиною нашого життя. Вони допомагають людям знаходити партнерів для серйозних стосунків, дружби, спілкування або навіть просто для цікавого проведення часу. Проте, за цією простотою ховається складна система, що поєднує в собі різноманітні технології, психологічні аспекти та маркетингові стратегії.

Перш за все, варто звернути увагу на профіль користувача. Це візитна картка людини на сайті, її обличчя та голос. Тут кожен може розповісти про себе, свої інтереси, захоплення, мрії та очікування від партнера. Чим більш детальним та цікавим є профіль, тим більше шансів привернути увагу потенційних партнерів. Проте, важливо пам'ятати про безпеку та не ділитися занадто особистою інформацією, яка може бути використана зловмисниками.

Пошук партнерів – це ключова функція сайтів знайомств. Завдяки різноманітним фільтрам, користувачі можуть звузити коло пошуку до тих, хто відповідає їхнім критеріям за віком, статтю, місцем проживання, інтересами тощо. Деякі сайти пропонують навіть розширений пошук за психологічними параметрами, що базується на теорії сумісності особистостей. Це дозволяє знаходити людей, з якими є не тільки спільні інтереси, але й психологічна сумісність, що є важливим для тривалих та гармонійних стосунків.

Комунікація – це серцевина сайту знайомств. Саме тут відбувається знайомство, спілкування та зближення людей. Сайти пропонують різні засоби зв'язку – від обміну текстовими повідомленнями до відеодзвінків. Важливо бути ввічливим, тактовним та відкритим до спілкування, але також не забувати про безпеку та обережність. Не варто відразу ділитися особистою інформацією, такою як номер телефону або домашня адреса. Краще спочатку поспілкуватися в межах сайту та переконатися в тому, що людина є дійсно тією, за кого себе видає.

Алгоритми підбору партнерів – це складна система, що лежить в основі роботи багатьох сайтів знайомств. Вони аналізують профілі користувачів, їхню

поведінку на сайті та інші дані, щоб запропонувати найбільш підходящих партнерів. Ці алгоритми постійно вдосконалюються, щоб забезпечити максимально точний та ефективний підбір. Проте, варто пам'ятати, що алгоритми – це лише інструмент, який допомагає знайти потенційних партнерів, але не гарантує успіху у стосунках.

Психологічний аспект – це важливий фактор успіху сайту знайомств. Люди приходять сюди з різними мотиваціями – хтось шукає серйозні стосунки, хтось – дружбу, а хтось – просто спілкування. Важливо враховувати ці потреби та створювати атмосферу довіри та підтримки на сайті. Адміністрація сайту повинна стежити за тим, щоб користувачі почувалися комфортно та безпечно, а також щоб на сайті не було місця для образ, дискримінації та інших негативних явищ.

Маркетингові стратегії – це необхідний елемент для залучення нових користувачів. Сайти знайомств використовують різні методи реклами та просування, щоб залучити аудиторію та збільшити свою популярність. Це можуть бути рекламні банери в інтернеті, оголошення в соціальних мережах, партнерські програми тощо. Проте, важливо, щоб маркетингові стратегії були чесними та прозорими, і не вводили користувачів в оману.

Безпека та модерація – це важливі аспекти роботи сайту знайомств. Адміністрація сайту повинна забезпечувати безпеку користувачів, контролювати контент та поведінку, а також боротися з шахрайством та зловживанням. Для цього використовуються різні методи, такі як модерація повідомлень, блокування підозрілих профілів, фільтрація контенту тощо. Проте, безпека користувачів залежить не тільки від адміністрації сайту, але й від самих користувачів. Важливо бути обережними та не довіряти незнайомим людям, не ділитися особистою інформацією та повідомляти про будь-які підозрілі дії.

Сайти знайомств – це складний та багатогранний світ, що постійно розвивається та змінюється. Вони допомагають людям знаходити одне одного, встановлювати зв'язки та будувати стосунки. Проте, важливо пам'ятати про безпеку та обережність, а також з повагою ставитися до інших користувачів.

1.3 Вимоги до сучасних web-сайтів для знайомств

Сучасні сайти знайомств - це вже не просто місце для швидкого пошуку партнера, це складний механізм, що поєднує в собі технології, дизайн, психологію та, звісно ж, безпеку. Користувачі очікують від таких платформ не тільки зручний інтерфейс, але й широкий спектр функцій, що допоможуть їм знайти саме ту людину, яка буде їм цікава.

Перш за все, сайт знайомств має бути інтуїтивно зрозумілим та легким у використанні. Навігація має бути логічною, а всі необхідні функції - під рукою. Користувач не повинен витрачати час на те, щоб розібратися, як працює той чи інший інструмент. Все має бути просто та зрозуміло з першого погляду.

Звичайно ж, дизайн відіграє важливу роль. Сучасний сайт знайомств повинен виглядати стильно та привабливо. Він має викликати позитивні емоції та бажання залишатися на сайті якомога довше. Важливо, щоб дизайн був не тільки красивим, але й функціональним. Він має сприяти зручному перегляду профілів, швидкому пошуку та комфортному спілкуванню.

Функціональність - це ще один важливий аспект. Сайт знайомств повинен пропонувати користувачам широкий спектр можливостей. Розширений пошук за різними критеріями, можливість додавати фото та відео, обмінюватися повідомленнями, ставити оцінки та залишати відгуки - це лише деякі з функцій, які можуть бути корисними. Чим більше можливостей, тим більше шансів у користувача знайти саме ту людину, яка йому потрібна.

Безпека - це, мабуть, найважливіший аспект. Сайт знайомств повинен гарантувати безпеку особистих даних користувачів та захищати їх від шахрайства та зловживань. Модерація профілів, можливість скаржитися на порушення, блокування небажаних користувачів - це необхідні інструменти для забезпечення безпеки. Користувачі повинні бути впевнені в тому, що їхні дані знаходяться під захистом.

Звичайно ж, сучасний сайт знайомств повинен бути адаптованим під різні пристрої. Все більше людей користуються смартфонами та планшетами для

доступу в інтернет, тому важливо, щоб сайт коректно відображався на екранах будь-якого розміру. Користувач повинен мати можливість користуватися сайтом з будь-якого місця та в будь-який час.

І, звичайно ж, не варто забувати про маркетингову стратегію. Сайт знайомств повинен бути відомим та популярним серед цільової аудиторії. Для цього необхідно використовувати різні методи просування - реклама в інтернеті, соціальних мережах, партнерські програми тощо. Чим більше людей знає про сайт, тим більше шансів у користувачів знайти свою долю.

Сучасні сайти знайомств - це складні та багатофункціональні платформи, які повинні відповідати високим вимогам користувачів. Вони повинні бути зручними, красивими, функціональними, безпечними та доступними на різних пристроях. Тільки тоді вони зможуть стати успішними та допомогти людям знайти своє щастя.

2 АНАЛІЗ ЗАСОБІВ РЕАЛІЗАЦІЇ WEB-САЙТУ

2.1 Засоби розробки серверної частини (Backend)

У сучасному світі розробка веб-сайтів є одним із ключових напрямів інформаційних технологій. Веб-сайт знайомств – це не просто набір веб-сторінок, а комплексний інформаційний продукт, що поєднує в собі різні технології та інструменти. Вибір засобів розробки значною мірою визначає як функціональні можливості сайту, так і його продуктивність, масштабованість, безпеку та зручність використання. У цьому розділі буде детально розглянуто технології, які використовуються для реалізації проєкту, зокрема засоби розробки серверної та клієнтської частин, а також засоби організації бази даних і вибору платформи для хостингу.

Django – це один із найпопулярніших фреймворків для розробки веб-додатків на Python. Він дозволяє швидко створювати складні веб-додатки завдяки вбудованим інструментам та гнучкій архітектурі. Розробникам надається можливість ефективно працювати з базами даних, налаштовувати аутентифікацію користувачів, керувати безпекою та адмініструванням сайту. Django базується на принципах "Don't Repeat Yourself" (DRY) та "Convention over Configuration", що дозволяє уникати зайвого дублювання коду та дотримуватись стандартних конфігурацій.

Однією з особливостей цього фреймворку є його архітектура Model-View-Template (MVT), яка дозволяє зручно організувати код. За допомогою моделей можна визначати структуру даних та взаємодіяти з базою без написання SQL-запитів, представлення відповідає за логіку обробки даних, а шаблони – за їх відображення у веб-інтерфейсі. Завдяки цій організації можна легко підтримувати та розширювати застосунок без зайвих ускладнень.

Django має багато переваг, серед яких автоматичне адміністрування, об'єктно-реляційне відображення даних (ORM), вбудовані механізми безпеки та масштабованість. Фреймворк підтримує різні бази даних, такі як PostgreSQL,

MySQL, SQLite та Oracle, що дає змогу адаптувати його під конкретні потреби проєкту.

Завдяки зручності та потужності Django активно використовується для створення різноманітних веб-додатків, включаючи системи керування контентом, соціальні мережі, інтернет-магазини та API для мобільних застосунків. Компанії на кшталт Instagram, Pinterest і Spotify інтегрували Django у свої технологічні рішення, що підтверджує його ефективність у розробці сучасних веб-застосунків.

Django поєднує в собі простоту та функціональність, що робить його привабливим як для новачків, так і для досвідчених розробників. Завдяки активній спільноті та широкій екосистемі, цей фреймворк залишається одним із найкращих інструментів для створення веб-додатків.

2.2 Засоби розробки клієнтської частини (Frontend): HTML, CSS, Bootstrap

Frontend розробка є важливою частиною створення сучасних веб-сайтів, особливо для таких складних проектів, як сайти знайомств. Вона відповідає за візуальну частину сайту, з якою безпосередньо взаємодіють користувачі. Для розробки клієнтської частини використовуються різні технології, серед яких HTML, CSS та Bootstrap є основними.

HTML (HyperText Markup Language)

HTML - це мова розмітки гіпертексту, яка є основою для створення веб-сторінок. Вона використовується для структурування контенту на сторінці, визначаючи заголовки, абзаци, списки, таблиці та інші елементи. HTML також дозволяє додавати на сторінку зображення, відео та інші мультимедійні об'єкти.

HTML складається з тегів, які описують різні елементи сторінки. Теги можуть мати атрибути, які задають додаткові властивості елементів. Наприклад, атрибут src тега <img> вказує на адресу зображення, яке потрібно відобразити на сторінці.

HTML5 - це остання версія мови HTML, яка включає в себе багато нових можливостей, таких як підтримка аудіо та відео, нові семантичні елементи та покращена підтримка форм.

CSS (Cascading Style Sheets)

CSS - це мова стилів, яка використовується для оформлення веб-сторінок. Вона дозволяє задавати кольори, шрифти, розміри та інші параметри елементів HTML. CSS також дозволяє створювати макети сторінок, розміщуючи елементи в потрібному порядку.

CSS використовує селектори для вибору елементів HTML, до яких потрібно застосувати стилі. Селектори можуть бути різними - за назвою тега, за класом, за ідентифікатором тощо.

CSS3 - це остання версія мови CSS, яка включає в себе багато нових можливостей, таких як підтримка тіней, градієнтів, анімації та інших ефектів.

Bootstrap - це популярний CSS-фреймворк, який значно спрощує розробку веб-сайтів. Він надає готовий набір стилів та компонентів, які можна використовувати для створення адаптивних та сучасних веб-сторінок.

Bootstrap включає в себе сітку, яка дозволяє легко розміщувати елементи на сторінці, а також набір компонентів, таких як кнопки, форми, навігаційні панелі та інші. Всі компоненти Bootstrap адаптивні, тобто вони автоматично підлаштовуються під розмір екрану пристрою, на якому відображається сайт.

Bootstrap також включає в себе JavaScript-плагіни, які додають інтерактивність на сайт. Наприклад, за допомогою JavaScript-плагінів можна створювати модальні вікна, випадаючі списки та інші елементи.

2.3 Організація зберігання та обробки даних засобами СУБД SQLite

SQLite — це легка, вбудована система управління базами даних, яка не потребує окремого серверного процесу і працює безпосередньо у вигляді звичайного файлу. Завдяки своїй простоті, вона ідеально підходить для

невеликих або середніх проєктів, а також для розробки прототипів і навчальних проєктів, зокрема сайтів знайомств.

Модель та схема даних

Для сайтів знайомств зазвичай використовується реляційна модель даних, яка реалізується у SQLite за допомогою таблиць, що складаються зі стовпців і рядків. Кожна таблиця представляє певну сутність, наприклад:

- Користувачі: зберігає ідентифікатори, імена, паролі (у хешованому вигляді), електронну пошту тощо.
- Профілі: містить додаткову інформацію про користувача — вік, стать, місто, інтереси, фото тощо.
- Повідомлення: зберігає історію листування між користувачами.
- Вподобання: фіксує інформацію про вподобання (лайки/свайпи).
- Друзі/збіги: може відображати взаємні вподобання або зв’язки.
- Зв'язки між таблицями та SQL-запити
- SQLite підтримує основні типи зв'язків: "один до одного", "один до багатьох" та "багато до багатьох". Наприклад:
- "Користувачі" ↔ "Профілі" — зв’язок "один до одного".
- "Користувачі" ↔ "Повідомлення" — зв’язок "один до багатьох", адже користувач може відправити багато повідомлень.

Запити до бази даних виконуються через мову SQL (Structured Query Language). Наприклад:

```
SELECT * FROM users WHERE gender = 'female' AND city = 'Kyiv';
```

або для отримання історії повідомлень між двома користувачами:

```
SELECT * FROM messages WHERE
(sender_id = 1 AND recipient_id = 2)
OR (sender_id = 2 AND recipient_id = 1)
ORDER BY timestamp ASC;
```

SQLite підтримує транзакції, що дозволяє виконувати кілька змін у базі як одну атомарну операцію. Наприклад, при реєстрації нового користувача можна в одному блоці вставити дані в таблицю users і profiles, щоб гарантувати цілісність даних.

Приклад транзакції в SQLite:

```
BEGIN TRANSACTION;
INSERT INTO users (username, email, password) VALUES (...);
INSERT INTO profiles (user_id, age, gender, city) VALUES (...);
COMMIT;
```

Безпека в SQLite реалізується переважно на рівні самої програми, оскільки СУБД не має вбудованої підтримки користувачів і прав доступу. Тому важливо реалізувати:

- хешування паролів (наприклад, через bcrypt у Django),
- захист від SQL-ін’єкцій (Django ORM автоматично захищає від них),
- контроль доступу через рівень логіки застосунку.

Обмеження та застосування

SQLite добре підходить для локальної розробки, тестування або невеликих сайтів, як-от MVP знайомств. Однак її масштабованість обмежена: вона не підтримує одночасну роботу великої кількості клієнтів та реплікацію на кілька серверів, як це можливо у PostgreSQL. Тому в майбутньому, при зростанні навантаження, можна мігрувати базу даних на PostgreSQL або іншу повноцінну серверну СУБД.

2.4 Вибір хмарної платформи для хостингу web-сайту – порівняння платформ Railway, Render, Vercel

Вибір відповідної хмарної платформи для хостингу веб-сайту є важливим етапом розробки та розгортання проєкту. Залежно від вимог до продуктивності,

масштабованості, зручності розгортання та технічної підтримки, розробники обирають найбільш відповідне рішення. У цьому підрозділі буде розглянуто три популярні хмарні платформи – Railway, Render та Vercel, які активно використовуються для хостингу веб-додатків, зокрема тих, що розроблені з використанням Django, PostgreSQL та інших сучасних технологій.

Railway: автоматизація та простота

Railway позиціонується як платформа, що дозволяє швидко та без зайвих налаштувань розгорнути веб-застосунок. Її основною перевагою є простота використання: розробники можуть легко підключати репозиторії, автоматично виконувати збірку додатку та розгортати його у виробничому середовищі. Railway підтримує різноманітні технології, включаючи Python/Django, Node.js, PostgreSQL та інші бази даних.

Railway пропонує безкоштовний тарифний план, який дозволяє тестувати невеликі проєкти, але для комерційного використання потрібно переходити на платні плани. Важливою перевагою Railway є автоматичне керування ресурсами – розробник не потребує глибоких знань DevOps, щоб розгорнути застосунок. Недоліком можна вважати обмеження безкоштовного плану та залежність від інфраструктури Railway, що може бути критичним у разі високого навантаження.

Render: баланс між простотою та гнучкістю

Render є ще однією популярною платформою для хостингу веб-додатків. Її ключовою особливістю є підтримка різноманітних технологій та простий процес CI/CD (безперервної інтеграції та доставки). Render дозволяє запускати як веб-сайти, так і бекенд-сервіси, роблячи його універсальним рішенням для розробників.

Однією з важливих переваг Render є можливість автоматичного масштабування, що особливо актуально для веб-додатків із високим навантаженням. Також платформа підтримує PostgreSQL, Redis та інші бази даних, що робить її зручною для використання з Django. Недоліком можна вважати те, що хоча безкоштовний план досить щедрий, для стабільної роботи у виробничому середовищі часто необхідно використовувати платні тарифи.

Render відрізняється простим процесом деплою: користувач підключає GitHub-репозиторій, а система автоматично збирає та запускає додаток. Однак, порівняно з Railway, Render може потребувати більше часу на конфігурацію, якщо проєкт має специфічні вимоги.

Vercel: оптимальне рішення для фронтенд-додатків

Vercel спеціалізується на хостингу фронтенд-застосунків, зокрема тих, що розроблені на основі React, Next.js, Vue.js та інших сучасних JavaScript-фреймворків. Основною перевагою Vercel є його оптимізація для статичних сайтів та серверних рендерингових рішень. Завдяки автоматичній підтримці GitHub-репозиторіїв розробники можуть швидко оновлювати свій веб-додаток без складних налаштувань.

Однією з головних переваг Vercel є його швидкість і оптимізованість для фронтенду. Він підтримує CDN (Content Delivery Network), що забезпечує мінімальні затримки під час завантаження сторінок користувачами з різних регіонів. Vercel також пропонує безкоштовний план, який підходить для невеликих проєктів, однак для серйозного комерційного використання може знадобитися платний тарифний план.

Основним недоліком Vercel є те, що він не підходить для хостингу бекенд-додатків, таких як Django або Flask. Тому для повноцінного веб-застосунку розробникам часто доводиться комбінувати Vercel із іншими платформами, такими як Railway або Render, які підтримують серверну логіку.

3 ПРОЕКТУВАННЯ ТА РОЗРОБКА WEB-САЙТІВ ЗНАЙОМСТВ

3.1 Проектування структури сайту

Проектування структури сайту — це перший етап розробки, який формує фундамент майбутньої системи знайомств. На цьому етапі визначається загальна архітектура, список необхідних сторінок, логіка переходів між ними, а також основна функціональність, яку має реалізовувати сайт. Основна мета — створити логічну, зручну та інтуїтивно зрозумілу навігацію, що відповідає очікуванням користувачів.

Початкове планування розробки сайту знайомств з аналізу потреб цільової аудиторії. Типовий користувач очікує просту реєстрацію, можливість заповнити профіль, переглядати інших користувачів, лайкати анкети та отримувати матчі. На основі цього визначаються такі ключові функції:

- реєстрація та авторизація користувачів;
- створення й редагування особистого профілю;
- перегляд анкет інших користувачів (у вигляді swipe або списку);
- взаємодія між користувачами (вподобання, матчі);
- чат або інша форма спілкування (можливо, як заготовка на майбутнє);
- налаштування профілю (вік, місто, інтереси, хобі тощо).

Після завершення етапу визначення ключового функціоналу веб-застосунку доцільно перейти до створення структури сайту у вигляді карти основних сторінок. Така карта дозволяє сформувати цілісне уявлення про архітектуру інтерфейсу користувача, забезпечити логічну навігацію та визначити шляхи взаємодії користувача із системою. Зазвичай структура сайту у сфері онлайн-знайомств включає низку обов’язкових сторінок, кожна з яких виконує конкретну роль у загальному функціонуванні платформи. У цьому проєкті реалізація сторінок здійснюється за допомогою системи шаблонів Django, що дає змогу ефективно комбінувати HTML-розмітку з логікою,

написаною на мові Python, та дозволяє динамічно формувати контент на основі даних із серверної частини.

Головна сторінка є обличчям сайту. Це перше, що бачить користувач при заході на платформу. Основне призначення цієї сторінки — надати загальну інформацію про сервіс, пояснити, які переваги має реєстрація, а також спонукати користувача долучитися до спільноти. У більшості випадків тут розміщується привабливий заголовок, короткий опис функціоналу сайту, а також заклики до дії (наприклад, кнопки «Зареєструватися» або «Увійти»).

Дизайн головної сторінки має бути візуально привабливим та інтуїтивно зрозумілим, з можливим використанням анімацій, слайдерів або відео на фоні, як це часто реалізується в сучасних сайтах знайомств. У Django ця сторінка реалізується за допомогою окремого шаблону home.html, який може бути розширенням базового шаблону, що включає загальні елементи інтерфейсу, як-от шапка та підвал (header, footer).

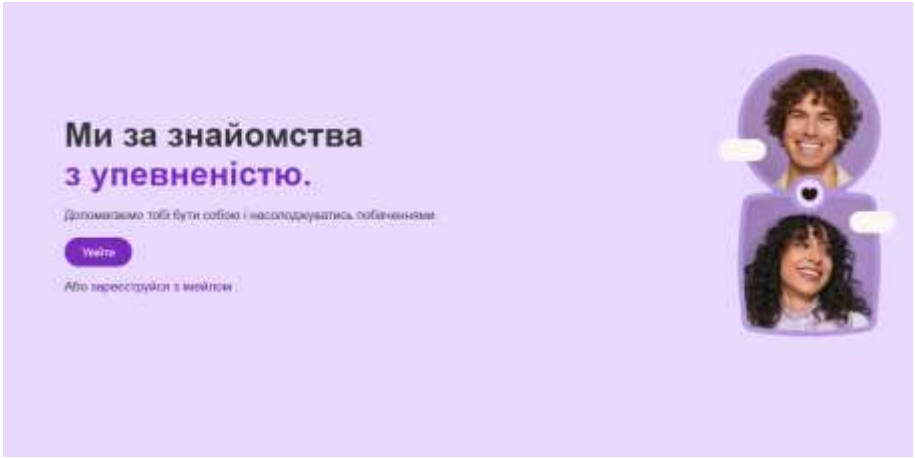


Рисунок 1.1 - Головна сторінка сайту у браузері.

Сторінка реєстрації та входу

Сторінка реєстрації та входу відіграє надзвичайно важливу роль у будь-якому сучасному веб-додатку, особливо у сервісах соціального типу, таких як сайт знайомств. Вона виконує критичну функцію — забезпечує процес

авторизації (тобто надання доступу до особистого кабінету користувача) та автентифікації (перевірки справжності введених даних). Без цієї сторінки неможливо гарантувати захист персональної інформації та належний рівень персоналізації взаємодії з системою.

З технічної точки зору сторінка складається з двох основних частин: форма реєстрації для нових користувачів та форма входу для тих, хто вже має обліковий запис у системі. Форма реєстрації зазвичай включає поля для введення електронної пошти, паролю, підтвердження паролю та, за потреби, інших базових даних (наприклад, ім’я, вік, стать тощо). Важливо також реалізувати валідацію введених даних — як на стороні клієнта, так і на стороні сервера — для забезпечення коректності та безпеки. Після успішної реєстрації користувач автоматично або вручну переходить до входу в систему.

Форма входу, у свою чергу, дозволяє авторизованим користувачам отримати доступ до функціоналу, що передбачає збереження сесії, перегляд персональних даних або взаємодію з іншими користувачами платформи. Введені під час входу дані звіряються із збереженими у базі — і лише у разі збігу користувач отримує доступ до свого профілю.

У середовищі розробки Django реалізація цієї функціональності може бути здійснена як за допомогою стандартних можливостей фреймворку, так і через створення кастомізованих рішень. Django містить вбудований модуль django.contrib.auth, який пропонує готові механізми для управління користувачами, а також шаблони для реєстрації, входу та виходу з системи. Цей модуль значно спрощує реалізацію базової логіки автентифікації.

У більш складних випадках або коли необхідна індивідуалізація процесу, розробники можуть створювати власні форми на основі класу forms.Form або forms.ModelForm, доповнюючи їх додатковими полями та логікою перевірки. Це дозволяє реалізувати, наприклад, перевірку унікальності електронної пошти, використання капчі або надсилання електронного листа з підтвердженням.

Шаблони register.html та login.html, відповідно, відповідають за відображення форм реєстрації та входу на стороні користувача. Вони мають бути

інтуїтивно зрозумілими, адаптивними для мобільних пристроїв і містити підказки, що допомагають користувачеві швидко та без помилок заповнити форму. Додатково можна додати повідомлення про помилки при неправильному введенні пароля, некоректному форматі електронної пошти чи вже існуючому акаунті.

Таким чином, сторінка реєстрації та входу є не просто технічним елементом, а важливою частиною користувацького досвіду, що має бути водночас функціональною, захищеною та зручною для взаємодії.

З метою покращення користувацького досвіду реєстрація на сайті була реалізована у вигляді покрокової (step-by-step) форми. Такий підхід дозволяє уникнути перевантаження користувача великою кількістю полів на одному екрані, натомість послідовно веде його через ключові етапи створення профілю. Кожен крок фокусує увагу на окремому аспекті — від введення електронної пошти та пароля до вибору статі, інтересів, міста та завантаження фото.

Форма реалізована на стороні клієнта за допомогою HTML, CSS та JavaScript, а обробка запиту здійснюється через Django. Нижче наведено уривок шаблону register.html, який відповідає за виведення першого кроку реєстрації:

```
<form id="registrationForm" method="POST" enctype="multipart/form-
data" action="{% url 'registration' %}">
  {% csrf_token %}
  <div class="step active" id="step1">
    <h2>Вітаю! Для реєстрації укажи свій email та пароль</h2>
    <p>Це необхідно для підтвердження аккаунта</p>
    <input type="email" id="email" name="email" placeholder="Твій
email" required />
    <input type="password" id="password" name="password"
placeholder="Пароль" required />
    <button type="button" class="btn"
onclick="validateStep1()">Продовжити</button>
  </div>
</form>
```

Після успішної валідації першого кроку користувача переводять до наступного етапу, де він обирає свою стать, ім’я та дату народження. У JavaScript реалізована логіка, яка перемикає кроки та перевіряє, чи заповнені обов’язкові поля перед переходом далі. Наприклад, функція validateStep1() перевіряє коректність заповнення полів email та пароля:

```
function validateStep1() {
  const email = document.getElementById('email').value.trim();
  const password = document.getElementById('password').value.trim();
  if (!email) { alert("Вкажи email"); return; }
  if (!password) { alert("Вкажи пароль"); return; }
  showStep(2);
}
```

Весь процес реєстрації охоплює п’ять кроків, кожен з яких відповідає за певну групу інформації. Це не лише спрощує сприйняття, а й дозволяє поступово збирати важливі персональні дані. Після завершення всіх кроків форма надсилається на сервер, де за допомогою Django відбувається створення нового користувача, збереження даних у базі, а також можливе надсилання листа-підтвердження.

Цей підхід забезпечує зручність та водночас гнучкість, оскільки дозволяє легко модифікувати або розширювати форму новими кроками в майбутньому, не порушуючи загальної структури взаємодії користувача з інтерфейсом.

Реалізація сторінки профілю користувача

У межах реалізації сайту знайомств особливу увагу було приділено створенню повноцінної, зручної та інтуїтивно зрозумілої сторінки профілю користувача. Сторінка профілю виконує надзвичайно важливу функцію у системі – вона є «вітриною» особистості користувача, дозволяє іншим учасникам сервісу краще його пізнати, а також є основною точкою керування особистою інформацією. У сучасних додатках для знайомств, таких як Badoo або Tinder, сторінка профілю не лише містить базову інформацію, а й забезпечує динамічну

взаємодію користувача з інтерфейсом, що значно покращує загальний досвід користування.

Під час реалізації цієї частини системи використовувалися такі технології як HTML, CSS, JavaScript, а також можливості шаблонізації Django. Весь інтерфейс створювався з урахуванням мобільного дизайну, що відповідає актуальним вимогам адаптивності. Це означає, що сторінка відображається коректно на різних пристроях – від смартфонів до планшетів, із забезпеченням зручного перегляду інформації.

З технічної точки зору, шаблон profile.html побудований таким чином, щоб у максимально естетичній формі відобразити як основну, так і додаткову інформацію про користувача. Верхня частина сторінки містить ім’я, вік, аватар (тобто основне фото), а також поточний статус. Також присутній прогрес-бар, який показує ступінь заповнення профілю – це поширена практика в подібних системах, яка мотивує користувача додавати більше інформації про себе.

Користувач має змогу натиснути кнопку «Редагувати профіль», яка відкриває повноекранне модальне вікно. Цей підхід дозволяє уникнути перенаправлень і підтримує динамічність інтерфейсу. Модальне вікно містить вкладки з полями, які можна редагувати: ім’я, дата народження, ріст, біографія, хобі, інтереси, а також інформація про освіту, роботу та тип відносин, які шукає користувач. Під час натискання на поля у формі відбувається інтерактивна обробка даних – вони тимчасово оновлюються на сторінці без збереження до бази. Це дозволяє протестувати зміну інформації перед її фактичним збереженням.

Особливу увагу приділено галереї фото. Реалізовано можливість завантаження до п’яти фотографій, які одразу ж виводяться в інтерфейсі. Кожне зображення відображається у форматі прев’ю, що дає змогу користувачеві оцінити зовнішній вигляд сторінки до збереження змін. Функціональність побудована на основі FileReader API у JavaScript, який дозволяє читати локальні файли та миттєво відображати їх у DOM.

Нижня частина сторінки містить мобільну панель навігації, яка повторює стиль додатків для знайомств: три основні іконки – домашня сторінка, пошук партнерів і профіль. Вони інтерактивні, змінюють активну іконку залежно від поточного розділу та забезпечують швидку навігацію між ключовими сторінками.

. У вкладці «Безпека» присутні опції зміни паролю, налаштування двофакторної автентифікації (2FA) та видалення облікового запису.

Для реалізації стилістичного оформлення всі стилі були винесені у окремий CSS-файл, що забезпечує чистоту шаблону та полегшує подальше обслуговування проекту. Іконки реалізовані за допомогою бібліотеки Font Awesome. Весь інтерфейс оформлений у сучасному фіолетово-білому кольоровому рішенні, яке є приємним для ока та створює асоціації з подібними мобільними додатками для знайомств.

Для прикладу, нижче наведено ілюстрацію зовнішнього вигляду сторінки профілю користувача з відкритим та закритим модальним вікном:

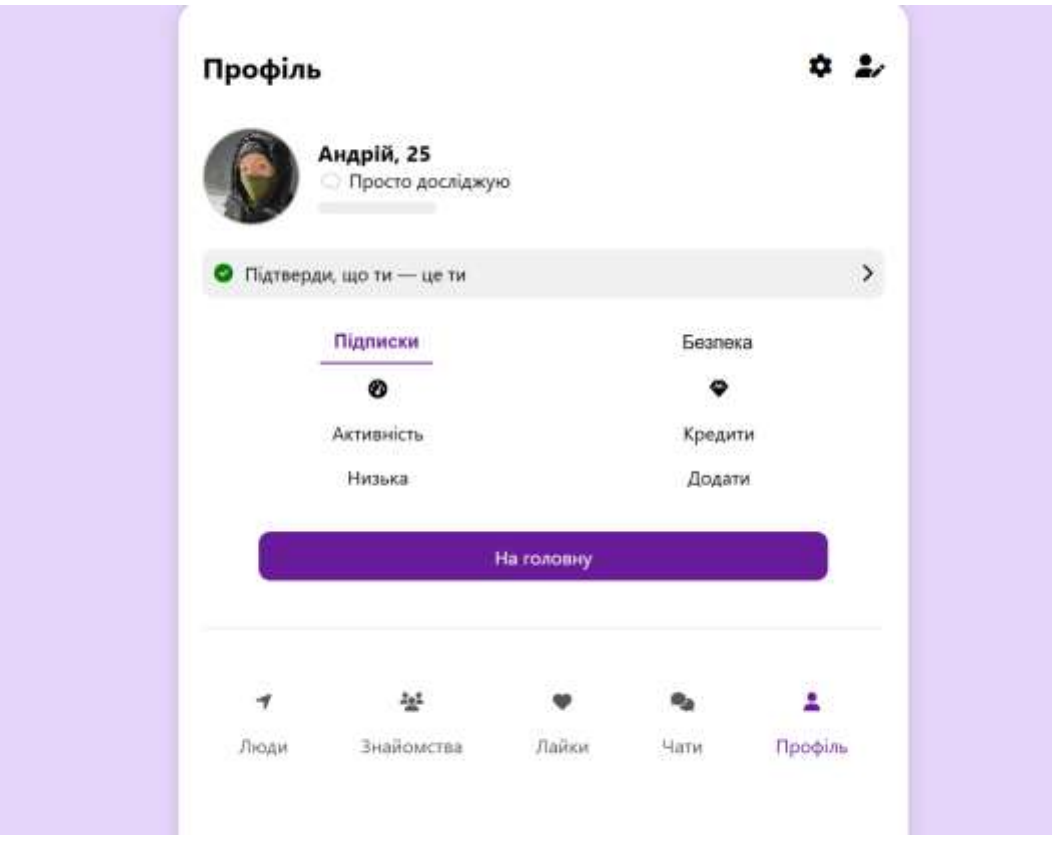


Рисунок 1.2 - Профіль користувача

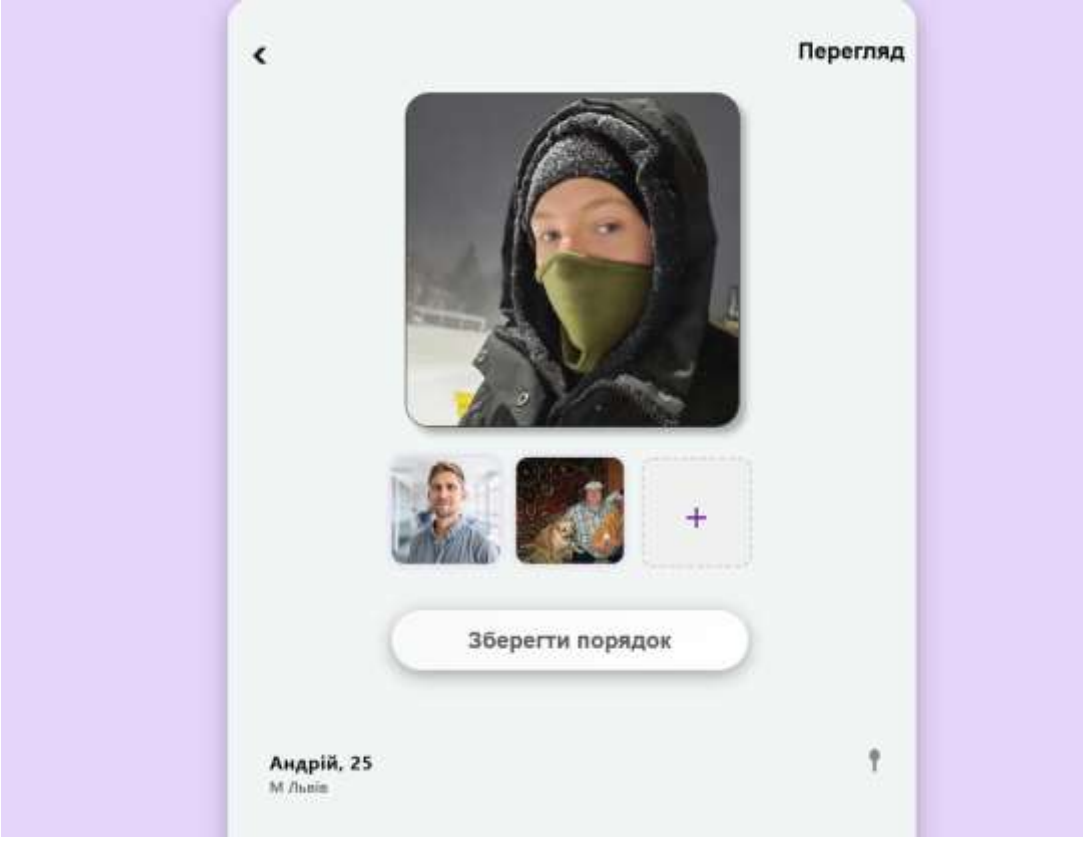


Рис 1.3 - Профіль користувача з відкритою сторінкою редагування профілю

Swipe або перегляд анкет

Сторінка свайпу є ключовим елементом взаємодії користувача із системою знайомств. Її головна мета — надати швидкий та інтуїтивний інтерфейс для перегляду потенційних партнерів. Ця сторінка реалізована у форматі карток, які можна "свайпати" — тобто, користувач має змогу відкинути анкету ліворуч (відмова) або вправо (симпатія), що забезпечує легкий і швидкий спосіб виявлення зацікавленості.

З технічної точки зору, сторінка реалізується як динамічне представлення на базі Django, HTML-шаблонів та JavaScript. Під час завантаження сторінки відбувається генерація набору анкет, які ще не були оцінені поточним користувачем. Вигрузка даних відбувається у view swipe_view, де за допомогою ORM Django виключаються анкети, які вже отримали лайк або дизлайк від користувача. Для цього формується список ідентифікаторів excluded_ids, який включає користувачів, вже оцінених (через модель Like), а також самого

користувача. Таким чином, у результаті запиту користувач отримує лише нові, ще не переглянуті анкети.

Ці анкети серіалізуються у вигляді JSON-структури й передаються до шаблону swipe.html, де далі опрацьовуються на стороні клієнта за допомогою JavaScript. Для кожного профілю генерується картка, яка містить фотографію, ім’я, вік, коротку біографію, а також додаткову інформацію: статус, місто, професію, хобі та (за можливості) відстань до користувача. Виведення здійснюється через динамічну побудову DOM-елементів за допомогою функції renderCard.

Користувач взаємодіє з карткою за допомогою двох основних кнопок — "лайк" та "дизлайк" . Кожна дія активує функцію swipe, яка не лише візуально анімує картку (плавне зміщення вліво чи вправо з обертанням), але й надсилає запит до backend-серверу через fetch. Запит надсилається на маршрут swipe_action, де реалізована обробка дій користувача.

Обробник swipe_action (POST-метод) приймає ідентифікатор користувача та напрям свайпу (left або right). Якщо користувач поставив "лайк", у базі створюється новий запис у таблиці Like, що зберігає пару from_user -> to_user. Додатково перевіряється, чи існує вже зворотний лайк від іншого користувача — у разі взаємної симпатії система повідомляє клієнта про успішний збіг (match), і для обох користувачів може бути створений приватний чат (через метод get_or_create_private_chat).

Для захисту запитів використовується механізм CSRF-токенів, які автоматично витягуються JavaScript-скриптом через функцію getCookie. Всі запити на бекенд мають необхідні заголовки, зокрема X-CSRFToken.

На візуальному рівні, сторінка оформлена у стилі мобільного застосунку — використовується модальне вікно, яке займає майже весь екран, адаптивна верстка та іконки. Картки з анкетами з’являються посередині, тоді як

навігаційне меню внизу дозволяє швидко перемикатися між іншими розділами сайту — профілем, чатом, лайками та пошуком людей поруч.

Важливо зазначити, що користувачеві, який переглянув усі доступні анкети, виводиться повідомлення "Немає нових користувачів для свайпу", що завершує сеанс перегляду і не допускає спроби свайпу порожніх даних.

Таким чином, реалізація свайпу є прикладом гармонійної інтеграції фронтенд- і бекенд-частин у рамках веб-системи знайомств. Вона поєднує зручність використання, естетичний вигляд та функціональність, що робить її критично важливою частиною платформи.

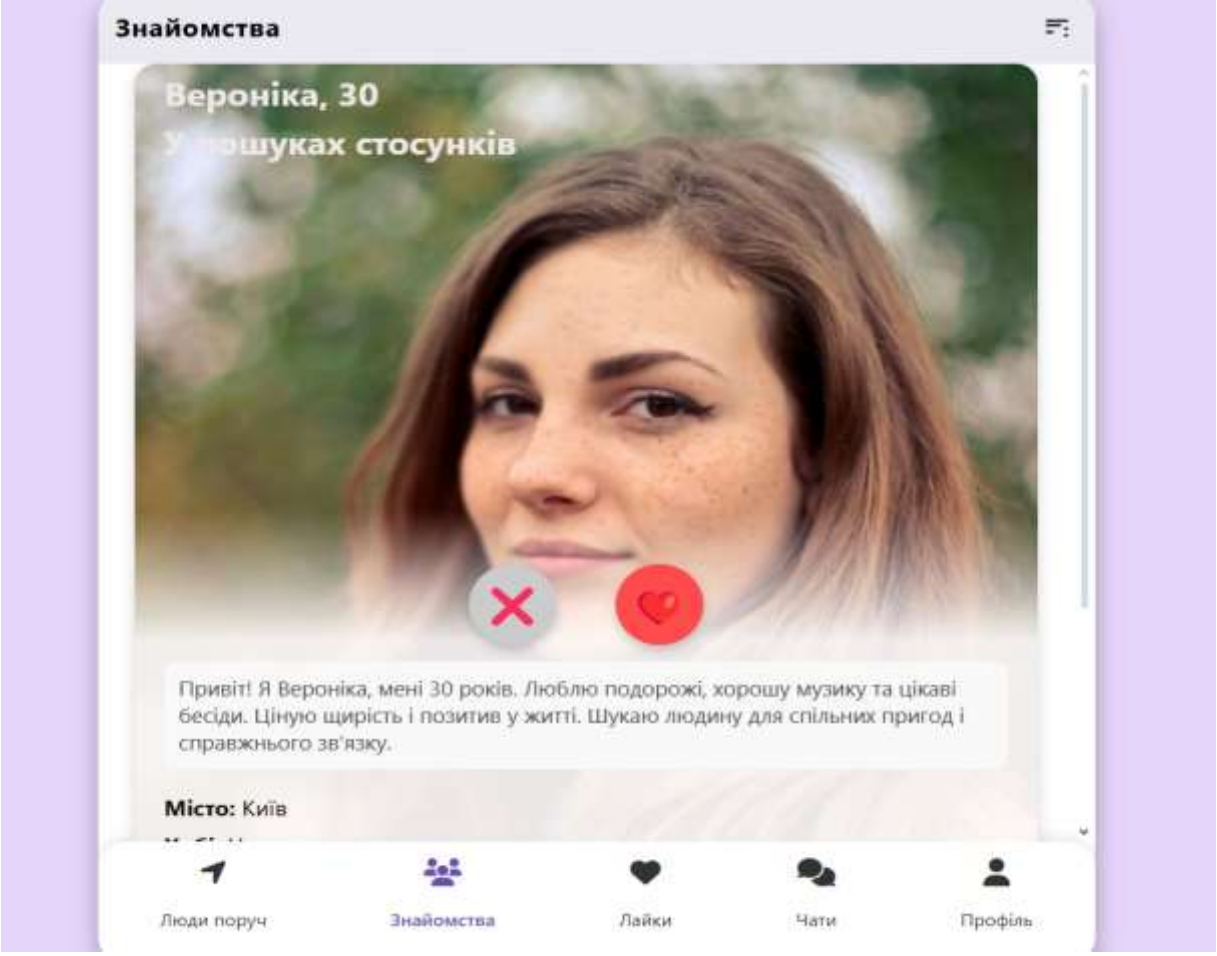


Рисунок 1.4 - Вигляд сторінки перегляду анкет.

3.2 Реалізація функціоналу створення профілю та пошуку партнерів

Для збереження та управління користувацькими даними в системі була створена кастомна модель User, яка розширює базову модель AbstractUser, надану фреймворком Django. Такий підхід дозволив адаптувати структуру даних під специфіку сайту знайомств, додавши необхідні поля для формування повноцінного профілю користувача.

Модель User містить розширені поля, що охоплюють різні аспекти інформації про користувача:

Персональні дані: до основних належать поля display_name, що відповідає за відображуване ім’я користувача, а також gender — стать, яка вибирається з обмеженого набору значень (чоловік або жінка). Важливим елементом є поле birth_date, яке зберігає дату народження. Для зручності роботи з віком користувача додано властивість age, реалізовану через декоратор @property. Ця властивість автоматично обчислює вік, враховуючи поточну дату, що дозволяє уникнути зберігання зайвих даних та гарантує актуальність інформації.

Інтереси та статус стосунків: у моделі передбачені поля interests та relationship, які мають набір фіксованих варіантів (choices). Це дозволяє стандартизувати введені дані та полегшує їх подальшу обробку. Користувач може позначити свій інтерес (наприклад, пошук спілкування або серйозних стосунків) та поточний статус у стосунках (одинокий, у стосунках тощо).

Додаткові характеристики: для розширеного опису особистості та вподобань користувача передбачені поля city (місто проживання), job (професія), education (освіта), hobbies (хобі), personality (опис особистості) та goals (життєві цілі). Ці дані допомагають формувати більш точні рекомендації та покращують релевантність пошуку партнерів.

Медіа: у моделі передбачено два поля для збереження зображень — avatar та profile_image. Вони відповідають за завантаження та збереження фотографій профілю користувача в окремі директорії на сервері, що дозволяє підвищити зручність використання та організувати структуру файлів.

Модель також містить стандартний метод __str__(), який повертає відображуване ім’я користувача або, у разі його відсутності, логін. Це полегшує роботу з об’єктами моделі у консолі адміністратора та у коді.

Для забезпечення функціоналу взаємодії між користувачами була реалізована модель Like. Вона фіксує інформацію про те, який користувач кому поставив "лайк". Основними полями є from_user — автор лайка, to_user — отримувач, а також поле created_at з датою і часом створення запису. Це дозволяє не тільки відслідковувати симпатії, але і реалізовувати логіку парування користувачів на основі взаємних лайків.

Крім того, для організації обміну повідомленнями користувачів між собою створено дві взаємопов’язані моделі — Chat та Message. Модель Chat підтримує зв’язок типу "багато-до-багатьох" між учасниками, що дозволяє формувати приватні діалоги між двома або більше користувачами. Модель Message містить поля для збереження тексту повідомлення, відправника (sender), отримувача (recipient), часу відправлення та статусу прочитання. Така архітектура забезпечує гнучкість і масштабованість системи обміну повідомленнями.

Для реалізації функціоналу створення і редагування профілю використовується view-функція edit_profile, яка обробляє HTTP-запити користувача. Ця функція інтегрується з формою UserProfileForm, що відповідає за валідацію та збереження даних профілю.

Обробка запитів поділена на дві гілки:

- **GET-запит:** при завантаженні сторінки редагування профілю у форму підставляються поточні дані користувача (instance=user), що дає змогу користувачеві побачити і змінити свої наявні дані.
- **POST-запит:** після відправлення форми з новими або зміненими даними, у тому числі з можливістю завантаження файлів, форму ініціалізують даними з запиту request.POST і файлами request.FILES. Виконується валідація полів, яка перевіряє коректність введеної інформації, зокрема, формат дати народження, відповідність

полів допустимим значенням, а також перевірку типів і розмірів завантажених файлів.

У разі успішної валідації форма зберігає оновлені дані у базі (form.save()).

Додатково, реалізована підтримка множинного завантаження зображень профілю — через отримання списку файлів з поля profile_images (request.FILES.getlist('profile_images')). Для кожного файлу створюється окремий запис у моделі Photo, пов’язаний з користувачем, що дозволяє зберігати кілька фотографій профілю.

Після успішного збереження профілю користувача відбувається перенаправлення на сторінку перегляду профілю (redirect('profile')).

```
@login_required
def edit_profile(request):
    user = request.user
    if request.method == 'POST':
        form = UserProfileForm(request.POST, request.FILES, 
instance=user)
        if form.is_valid():
            form.save()

            # Обробка множинного завантаження фотографій профілю
            files = request.FILES.getlist('profile_images')
            for f in files:
                Photo.objects.create(user=user, image=f)

            return redirect('profile')  # Переадресація на сторінку 
профілю користувача
    else:
        form = UserProfileForm(instance=user)

    return render(request, 'accounts/edit_profile.html', {'form': 
form})
```

Інтерфейс для свайп-карток реалізовано як модальне вікно, яке відразу відкривається при завантаженні сторінки. Основна область – це контейнер для карток користувачів, який має id cards-container. Самі картки створюються

динамічно за допомогою JavaScript на основі масиву користувачів, який передається з бекенду у вигляді JSON-рядка.

Для кожного користувача генерується окрема картка, де відображається аватар, ім'я, вік, статус, біо, місто, хобі, професія, відстань і інша додаткова інформація. Важливо, що для безпеки вивід тексту відбувається через функцію екранування HTML, аби уникнути можливих XSS-уразливостей.

Під карткою розташовані дві кнопки — «Dislike» та «Like» . Коли користувач натискає на будь-яку з них, відбувається анімація свайпу карти вліво або вправо, відповідно. Після цього JavaScript відправляє асинхронний POST-запит на сервер із інформацією про вибір користувача (ідентифікатор профілю і напрям свайпу).

Серверна частина обробляє цей запит, зберігає оцінку у базі даних, і у випадку взаємної симпатії повертає у відповідь дані про користувача, з яким стався матч. Клієнт у цей момент показує спеціальне модальне вікно з повідомленням «Це збіг!» та ім’ям і аватаром співпадіння. Це підвищує емоційну залученість користувача і спонукає розпочати спілкування.

Якщо натиснути кнопку закриття на вікні збігу, воно зникає, і користувач може продовжувати свайпати інших кандидатів.

Крім того, реалізовано окрему модалку для перегляду фотогалереї користувача. Коли користувач клікає на фото, відкривається слайдер із можливістю перегляду фотографій за допомогою стрілок «вперед» і «назад», а також кнопки закриття. Всі ці елементи керуються чистим JavaScript без використання сторонніх бібліотек.

Для захисту від CSRF-атак при відправці POST-запитів використовується стандартний механізм Django з отриманням токена із куків.

Загалом цей фронтенд забезпечує плавний, інтерактивний та безпечний досвід користувача, який максимально наближений до сучасних мобільних додатків знайомств.

Для реалізації функції свайп-карток у вебдодатку було використано комбінацію HTML, CSS та JavaScript, що забезпечує зручний та інтуїтивно

зрозумілий інтерфейс для користувачів. Основна структура картки побудована на базі HTML-блоку, який містить усі необхідні елементи: фотографію, ім’я, вік та додаткову інформацію, таку як статус, хобі та інші персональні дані. Фото виводиться в спеціальному контейнері, де застосовані стилі, які забезпечують правильне відображення зображення без спотворень, за допомогою властивості CSS object-fit: cover. Завдяки цьому фото завжди виглядає акуратно та гармонійно.

Всі елементи картки стилізовані за допомогою CSS, де окремо налаштовані розміри, відступи та заокруглення кутів, що надає блокам м’якості та сучасного вигляду. Особливу увагу приділено анімаціям, які відповідають за плавний перехід і переміщення карток при свайпі. Для цього використано CSS-властивості transform і transition, які дозволяють створити ефект «кидання» картки вправо чи вліво з легким обертанням, що імітує природний рух. Такий підхід робить взаємодію більш живою і приємною.

У JavaScript реалізована логіка, яка відстежує, яка картка зараз активна — це здійснюється за допомогою змінної currentIndex. Вона зберігає індекс поточного користувача в масиві всіх профілів і оновлюється після кожного свайпу, завдяки чому можна послідовно показувати користувачів без дублювань. Перед кожним свайпом перевіряється, чи залишилися ще картки — якщо всі профілі переглянуті, то подальші свайпи просто не виконуються, що запобігає виникненню помилок.

Для відправки інформації про вибір користувача (наприклад, лайк чи дизлайк) на сервер використовується технологія fetch із POST-запитами. У запитах обов’язково передається CSRF-токен — це спеціальний захисний код, який потрібен для запобігання шкідливим атакам типу CSRF (Cross-Site Request Forgery). Токен отримується з cookies браузера за допомогою функції, яка шукає відповідний запис у списку кукісів, і потім включається у заголовок HTTP-запиту. Це дозволяє серверу впевнитися, що запит зроблено дійсно з нашого сайту, а не стороннім ресурсом.

Окрім цього, у JavaScript передбачена обробка помилок, які можуть виникнути під час запиту — наприклад, якщо з'єднання з сервером не вдалося встановити або сервер повернув помилку. У таких випадках користувач не отримує незручних збоїв у роботі, а помилка логиться у консоль розробника для подальшого аналізу. Це робить додаток більш стійким до непередбачуваних ситуацій.

Таким чином, поєднання чіткої HTML-структури, продуманого CSS-дизайну та динамічної логіки на JavaScript дозволяє створити плавний і приємний інтерфейс свайп-карток, який забезпечує ефективну взаємодію користувача із сайтом знайомств. Особливо важливо, що всі операції взаємодії із сервером захищені від потенційних атак, що гарантує безпеку і стабільність роботи вебдодатку.

3.3 Архітектура серверної частини

Серверна частина вебсайту знайомств реалізована з використанням фреймворку Django, який побудований за архітектурною моделлю MVT (Model–View–Template). Це означає, що логіка додатку поділяється на моделі (що відповідають за роботу з базою даних), представлення (що обробляють запити користувача) і шаблони (що відповідають за відображення інтерфейсу).

Проєкт реалізований за класичною структурою Django-проєкту. Головний каталог містить основний конфігураційний файл, утиліту керування, а також модулі додатків. Ключовим є додаток accounts, у якому зосереджена вся логіка роботи з користувачами, повідомленнями, лайками та чатами. Додатково є директорії для зберігання завантажених файлів (media), статичних ресурсів (static) і шаблонів.

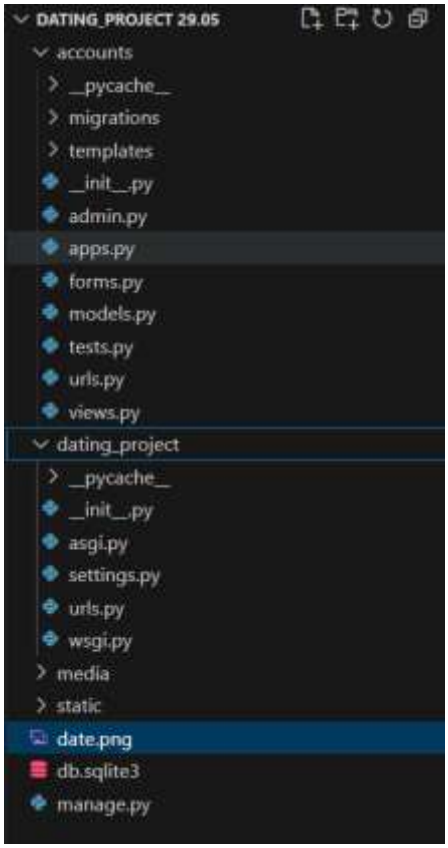


Рисунок 1.5 - Архітектура каталогу, відкритого у VS Code

Модель User є розширеною версією стандартної моделі Django AbstractUser. Вона містить поля для біографії, аватара, дати народження, хобі, міста, інтересів,

Вона зберігає не лише базові поля для аутентифікації, а й містить додаткові поля для персоналізації профілю користувача:

- interests — інтереси користувача (перелік вибору),
- bio — текстова біографія,
- avatar — аватар користувача,
- display_name — ім’я для відображення у інтерфейсі,
- gender — стать (чоловік/жінка),
- relationship — статус стосунків,
- profile_image — додаткове фото профілю,
- city — місто проживання,
- phone — контактний телефон,

- birth_date — дата народження,
- job, height, goals, education, status, personality, hobbies — додаткові поля для розширеного опису користувач

Вік користувача не зберігається у базі даних як окреме поле. Натомість він динамічно обчислюється за допомогою властивості @property, що враховує поточну дату і дату народження користувача.

```
@property
def age(self):
    if self.birth_date:
        today = date.today()
        age = today.year - self.birth_date.year
        if (today.month, today.day) < (self.birth_date.month,
self.birth_date.day):
            age -= 1
        return age
    return None
```

Окрім цього, система підтримує завантаження декількох фото, кожне з яких має свою модель — Photo.

```
class User(AbstractUser):
    RELATIONSHIP_CHOICES = [
        ('single', 'Одинокий'),
        ('in_a_relationship', 'У стосунках'),
        ('married', 'Одружений/Заміжня'),
        ('complicated', 'Складні стосунки'),
    ]

    INTEREST_CHOICES = [
        ('communication', 'Шукаю спілкування'),
        ('relationship', 'У пошуках стосунків'),
        ('explore', 'Просто досліджую'),
    ]

    interests = models.CharField(max_length=20,
choices=INTEREST_CHOICES, blank=True, null=True)
    bio = models.TextField(blank=True)
```

```
    avatar = models.ImageField(upload_to='avatars/', blank=True,
null=True)
    display_name = models.CharField(max_length=150,
verbose_name="Ім’я", blank=True)

    gender = models.CharField(
        max_length=10,
        choices=[('M', 'Чоловік'), ('F', 'Жінка')],
        blank=True
    )

    relationship = models.CharField(
        max_length=20,
        choices=RELATIONSHIP_CHOICES,
        blank=True
    )

    # age = models.PositiveIntegerField(null=True, blank=True)

    profile_image = models.ImageField(upload_to='profiles/',
blank=True, null=True)

    city = models.CharField(max_length=100, blank=True)
    phone = models.CharField(max_length=20, blank=True)
    birth_date = models.DateField(null=True, blank=True)  #

    job = models.CharField(max_length=100, blank=True)
    height = models.PositiveIntegerField(null=True, blank=True)
    goals = models.TextField(blank=True)
    education = models.CharField(max_length=100, blank=True)
    status = models.CharField(max_length=255, blank=True,
null=True)
    personality = models.TextField(blank=True)
    hobbies = models.TextField(blank=True)

    @property
    def age(self):
        if self.birth_date:
            today = date.today()
            age = today.year - self.birth_date.year
            if (today.month, today.day) < (self.birth_date.month,
self.birth_date.day):
                age -= 1
            return age
        return None
```

```
    def __str__(self):
        return self.display_name or self.username
```

Основні характеристики:

– Модель містить багато додаткових полів, які описують особисті дані користувача: інтереси, біографія, фото, стать, статус стосунків, місто, телефон, дата народження тощо.

– Вік користувача (age) не зберігається у базі, а обчислюється динамічно через властивість @property на основі дати народження.

– Поле display_name дозволяє задавати ім’я, яке буде відображатися у інтерфейсі замість логіну.

Модель Like реалізує функціонал лайків між користувачами. Якщо два користувачі взаємно лайкнули один одного, створюється екземпляр Chat, що об’єднує їх у приватному чаті. Повідомлення між користувачами зберігаються у моделі Message, яка містить вказівку на чат, відправника, отримувача, текст повідомлення, мітку часу та статус прочитання.

```
class Like(models.Model):
    from_user = models.ForeignKey(settings.AUTH_USER_MODEL, 
on_delete=models.CASCADE, related_name='likes_given')
    to_user = models.ForeignKey(settings.AUTH_USER_MODEL, 
on_delete=models.CASCADE, related_name='likes_received')
    created_at = models.DateTimeField(auto_now_add=True)


    def __str__(self):
        return f"{self.from_user} liked {self.to_user}"
```

- Відображає факт, що один користувач, постав лайк іншому.
- Взаємний лайк (коли один користувач лайкнув іншого) може ініціювати створення приватного чату.

Лістинг усіх моделей наданий у додатку А

ER-діаграма структури надана у додатку Б

Міграції бази даних

Після створення або зміни моделей Django використовує систему міграцій для застосування цих змін у структурі бази даних. Міграції — це набір автоматично згенерованих файлів, які описують, як створити або змінити таблиці у базі.

Приклад команд для створення та застосування міграцій:

```
python manage.py makemigrations
python manage.py migrate
```

У випадку з моделями в нашому прикладі, ці команди утворюють таблиці для User, Photo, Like, Chat та Message зі всіма відповідними полями і з’вязками.

Так, приклад файлів міграцій — це фактичні Python-файли, які Django автоматично створює у папці migrations всередині кожного додатку (app). Вони містять код, який описує зміну структури бази даних.

Ось приклад простого файлу міграції, який Django може згенерувати для додатку, що створює модель User із кількома полями:

Generated by Django 5.2 on 2025-05-15 22:30

```
import django.contrib.auth.models
import django.contrib.auth.validators
import django.utils.timezone
from django.db import migrations, models

class Migration(migrations.Migration):

    initial = True

    dependencies = [
        ('auth', '0012_alter_user_first_name_max_length'),
    ]

    operations = [
```

```
        migrations.CreateModel(
            name='User',
            fields=[
                ('id', models.BigAutoField(auto_created=True,
primary_key=True, serialize=False, verbose_name='ID')),
                ('password', models.CharField(max_length=128,
verbose_name='password')),
                ('last_login', models.DateTimeField(blank=True,
null=True, verbose_name='last login')),
                ('is_superuser',
models.BooleanField(default=False, help_text='Designates that this
user has all permissions without explicitly assigning them.',
verbose_name='superuser status')),
                ('username',
models.CharField(error_messages={'unique': 'A user with that
username already exists.'}, help_text='Required. 150 characters or
fewer. Letters, digits and @/./+/-/_ only.', max_length=150,
unique=True,
validators=[django.contrib.auth.validators.UnicodeUsernameValidato
r()], verbose_name='username')),
                ('first_name', models.CharField(blank=True,
max_length=150, verbose_name='first name')),
                ('last_name', models.CharField(blank=True,
max_length=150, verbose_name='last name')),
                ('email', models.EmailField(blank=True,
max_length=254, verbose_name='email address')),
                ('is_staff', models.BooleanField(default=False,
help_text='Designates whether the user can log into this admin
site.', verbose_name='staff status')),
                ('is_active', models.BooleanField(default=True,
help_text='Designates whether this user should be treated as
active. Unselect this instead of deleting accounts.',
verbose_name='active')),
```

```
                ('date_joined',
models.DateTimeField(default=django.utils.timezone.now,
verbose_name='date joined')),
                ('bio', models.TextField(blank=True)),
                ('avatar', models.ImageField(blank=True,
upload_to='avatars/')),
                ('gender', models.CharField(blank=True,
choices=[('M', 'Чоловік'), ('F', 'Жінка')], max_length=10)),
                ('age', models.PositiveIntegerField(blank=True,
null=True)),
                ('interests', models.TextField(blank=True)),
                ('profile_image', models.ImageField(blank=True,
upload_to='profiles/')),
                ('groups', models.ManyToManyField(blank=True,
help_text='The groups this user belongs to. A user will get all
permissions granted to each of their groups.',
related_name='user_set', related_query_name='user',
to='auth.group', verbose_name='groups')),
                ('user_permissions',
models.ManyToManyField(blank=True, help_text='Specific permissions
for this user.', related_name='user_set',
related_query_name='user', to='auth.permission',
verbose_name='user permissions')),
            ],
            options={
                'verbose_name': 'user',
                'verbose_name_plural': 'users',
                'abstract': False,
            },
            managers=[
                ('objects',
django.contrib.auth.models.UserManager()),
            ],
        ),
    ]
```

Це — перша міграція, яка створює таблицю User у базі даних із відповідними полями.

Переваги використання міграцій полягають у тому, що вони автоматизують процес оновлення структури бази даних, позбавляючи розробника необхідності вручну писати SQL-код для внесення змін. Кожна міграція є зафіксованим кроком, що дозволяє контролювати версії та відслідковувати історію змін структури бази. Крім того, Django підтримує можливість відкату міграцій, що дуже корисно під час тестування або виправлення помилок. Міграції створюються у нейтральній формі, завдяки чому вони сумісні з різними системами управління базами даних, такими як PostgreSQL, MySQL, SQLite тощо.

Типові операції, які описують міграції, включають створення нової таблиці для моделі (наприклад, User), додавання нового поля до існуючої таблиці (наприклад, поле phone), зміну властивостей поля (наприклад, збільшення довжини текстового поля) або видалення поля чи таблиці під час рефакторингу.

В контексті нашого проєкту вебсайту знайомств міграції відповідають за створення таблиць для користувачів із розширеними профілями, формування таблиць для збереження фото, лайків, чатів та повідомлень, а також забезпечення зв’язків між таблицями, наприклад, через зовнішні ключі між моделями Like і User. Система міграцій дозволяє безболісно додавати нові поля або моделі у майбутньому без втрати даних. Таким чином, вона є фундаментальним інструментом для підтримки цілісності та еволюції структури бази даних протягом усього життєвого циклу проєкту.

Опис функціоналу Django views для сайту знайомств.

1. Реєстрація користувача (registration_view)

Функція registration_view відповідає за обробку реєстрації нового користувача на сайті. Вона працює з методом POST, отримуючи з форми дані: email, пароль, ім'я, дату народження, стать, інтереси, місто і фото профілю. Валідація включає перевірку обов'язкових полів — email та пароль. Якщо

користувач з таким email вже існує, функція повертає повідомлення про помилку.

Для збереження даних застосовується кастомна модель користувача User, де ім’я користувача відповідає email (username). Дата народження конвертується у формат datetime.date. Стать кодується за допомогою словника gender_map, де "Чоловік" відповідає 'M', "Жінка" — 'F'. Фото профілю, якщо воно додане, зберігається у полі avatar.

Після успішного створення користувача, відбувається перенаправлення на сторінку профілю.

2. Вхід користувача (login_view)

Функція login_view обробляє аутентифікацію користувача. Вона використовує стандартну форму Django AuthenticationForm для перевірки коректності введеного логіну та паролю. Якщо користувач успішно авторизувався, то виконується логін сесії через login(request, user).

Особливість — реалізація функції «Запам’ятати мене» (remember_me). Якщо ця опція не вибрана, сесія користувача буде дійсна лише під час відкриття браузера (set_expiry(0)), в іншому випадку — два тижні (1209600 секунд).

При невдалій аутентифікації користувачу виводиться повідомлення про помилку.

3. Профіль користувача (our_profile та profile_view)

Функція our_profile показує персональний профіль залогіненого користувача. Для візуалізації на сторінці в контекст передаються ім'я користувача, стать (через метод get_gender_display()), дата народження (поки що заглушка), а також URL фото профілю, якщо воно є.

Інша функція — profile_view — також відображає профіль, але з додатковою обробкою інтересів користувача. Тут використовується словник для відображення текстового опису інтересів у вигляді іконок та підписів (наприклад, " Шукаю спілкування"). Вік користувача обчислюється динамічно через властивість-модель age.

4. Логіка свайпів (swipe_view та swipe_action)

swipe_view відповідає за відображення списку користувачів для свайпів. З метою уникнення повторів, користувачі, яких поточний користувач вже лайкнув, а також він сам, виключаються зі списку. Для кожного користувача формується JSON-об’єкт із детальною інформацією: ім'я, вік, інтереси, посилання на аватар та додаткові фото.

Функція swipe_action приймає AJAX-запит із дією свайпу: вправо (лайк) чи вліво (дизлайк). Якщо користувач свайпає вправо, створюється запис у таблиці Like. Якщо лайк взаємний (два користувачі лайкнули один одного), створюється приватний чат, а клієнту повертається відповідь про матч.

5. Чати та повідомлення (chat_list_view, chat_detail_view, chat_view)

chat_list_view показує список чатів, у яких бере участь користувач, упорядкованих за датою останнього оновлення. Для кожного чату визначається співрозмовник, останнє повідомлення та дата.

chat_detail_view відображає конкретний чат з усіма повідомленнями, що відображаються у хронологічному порядку. При отриманні POST-запиту створюється нове повідомлення, і користувач перенаправляється назад у чат.

Функція chat_view відповідає за виведення повідомлень між двома користувачами. Вона також відмічає непрочитані повідомлення як прочитані.

6. Редагування профілю (edit_profile)

У функції edit_profile обробляється форма редагування профілю користувача. Вона працює із кастомною формою UserProfileForm, яка дозволяє оновити особисті дані та завантажити кілька фотографій одночасно. Після валідації даних користувач перенаправляється на сторінку свого профілю.

7. Додатковий функціонал

- like_user — функція лайку конкретного користувача. При взаємному лайку створюється чат, якщо його не було раніше.
- filtered_profiles — фільтрація профілів за параметрами (стать, вік, відстань) з отриманням параметрів через GET-запит.

– mark_messages_as_read — позначення повідомлень як прочитаних у чаті через AJAX-запит.

– upload_photo — приймає файл фото через POST-запит та зберігає його, повертаючи ID та URL зображення у форматі JSON.

– update_photo_order — оновлює порядок фотографій, що зберігаються у профілі, за допомогою списку з ID і порядком.

3.4 Тестування функціоналу та безпеки системи

З метою перевірки надійності реалізованої веб-системи було виконано базове тестування функціональності та сканування на потенційні вразливості безпеки.

Було реалізовано набір автоматичних тестів за допомогою вбудованого фреймворку Django TestCase. Основна мета — перевірка базових функцій: створення користувачів, механізму лайків, надсилання повідомлень.

Фрагмент коду тестів:

```
from django.test import TestCase
from django.contrib.auth import get_user_model
from dating.models import Like, Message

User = get_user_model()

class BasicTests(TestCase):
    def setUp(self):
        self.user1 = User.objects.create_user(username='user1',
password='pass1234')
        self.user2 = User.objects.create_user(username='user2',
password='pass1234')

    def test_user_creation(self):
        self.assertEqual(User.objects.count(), 2)

    def test_like_logic(self):
        Like.objects.create(from_user=self.user1,
to_user=self.user2)
        self.assertEqual(Like.objects.count(), 1)

    def test_send_message(self):
```

```
        Message.objects.create(sender=self.user1,
receiver=self.user2, content='Привіт!')
        self.assertEqual(Message.objects.count(), 1)
```

Всі тести пройшли успішно, це свідчить про коректну роботу ключових частин системи.

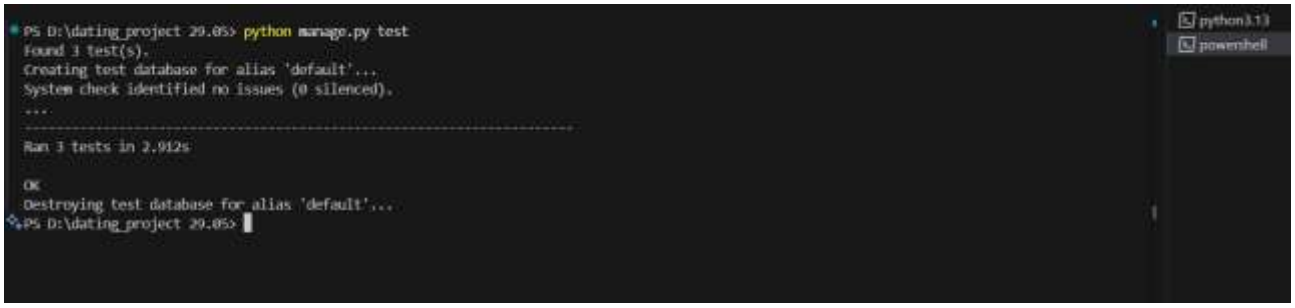


Рисунок 1.6 - Тести у терміналі VS Code

Сканування безпеки за допомогою OWASP ZAP

У рамках оцінки захищеності веб-застосунку було проведено тестування на наявність базових вразливостей з використанням сучасного інструменту OWASP ZAP (Zed Attack Proxy). Це один із найпопулярніших і рекомендованих засобів для автоматизованого тестування безпеки веб-додатків, який активно підтримується спільнотою OWASP (Open Web Application Security Project).

Для аналізу застосунку використовувалась версія 2.16.1 інструменту OWASP ZAP. Сканування проводилося у локальному середовищі розробки — веб-сайт був запущений на адресі http://127.0.0.1:8000, що дозволило протестувати всі основні маршрути та компоненти без потреби публічного розгортання.

Було обрано пасивний режим сканування, що дозволяє оцінити потенційні ризики, не змінюючи структуру запитів чи даних, які обробляються застосунком. Такий підхід є безпечним і рекомендується під час первинного аудиту — особливо на етапі розробки, коли продуктивна база даних ще не використовується, а функціональність активно змінюється.

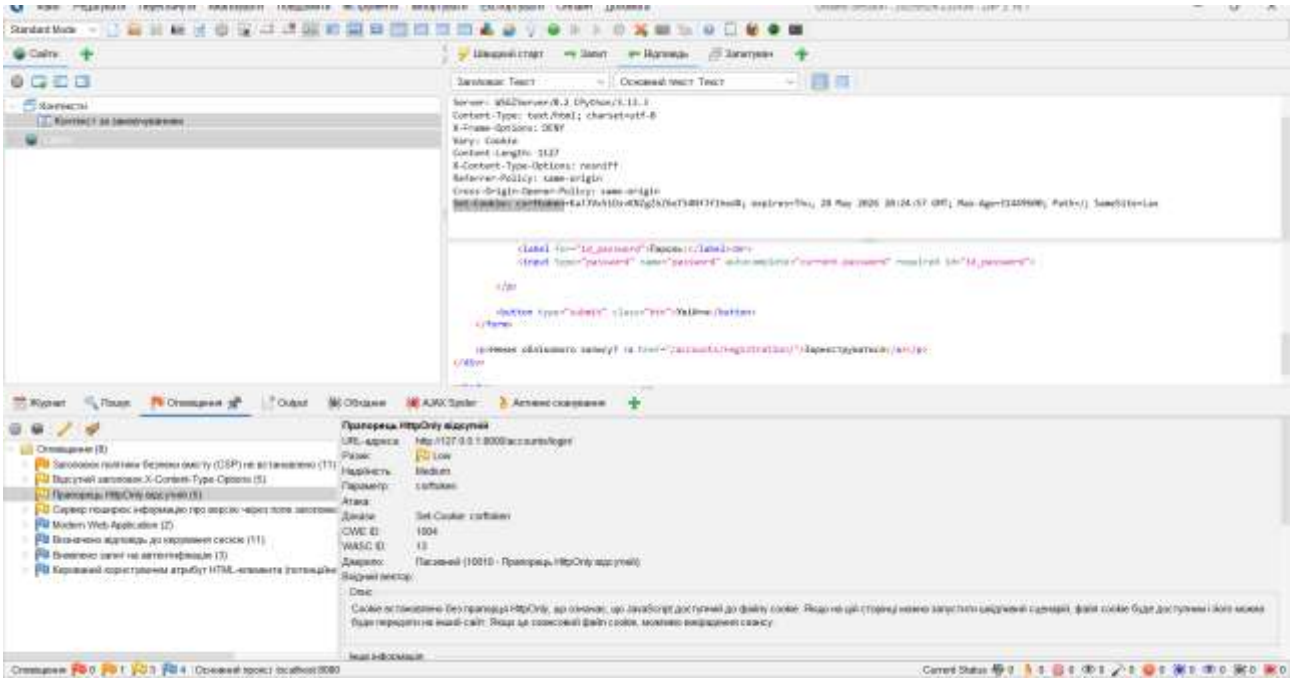


Рисунок 1.7 - Вікно користувача у OWASP ZAP

Сканування охопило ключові сторінки та маршрути, зокрема головну сторінку, сторінки реєстрації та входу, функціонал облікових записів, сторінки профілів та обробки повідомлень, статичні ресурси (CSS, JS, зображення).

Усього було виявлено 8 попереджень, які були розподілені за рівнями ризику наступним чином:

- 1 попередження середнього ризику
- 3 попередження низького ризику
- 4 попередження інформаційного характеру

Такий розподіл є типовим для веб-застосунків, які перебувають на етапі активної розробки та ще не мають повного комплекту налаштувань безпеки, притаманних продуктивному середовищу. Нижче наведено детальний опис кожного з виявлених пунктів.

Content Security Policy (CSP) not set:

Рівень ризику: середній

Заголовок Content-Security-Policy (CSP) дозволяє чітко визначити, з яких джерел можуть завантажуватись скрипти, стилі, зображення тощо. Його відсутність відкриває потенційну можливість для атак типу XSS (Cross-Site Scripting), за яких зловмисник може вставити шкідливий скрипт у веб-сторінку.

Задля зменшення ризиків варто вказати політику безпеки у HTTP-заголовках.

Наприклад:

Content-Security-Policy: default-src 'self';

Або більш гнучко:

Content-Security-Policy: default-src 'self'; img-src *; script-src 'self' 'unsafe-inline';

Витік інформації про сервер через заголовок Server

- Рівень ризику: низький
- Опис: HTTP-відповіді сервера містили заголовок Server, у якому було зазначено інформацію про використовувану технологію або версію вебсервера (наприклад, "WSGIServer/0.2"). Це, на перший погляд, нешкідлива інформація, однак вона може бути використана зловмисниками для точного підбору вектору атаки (наприклад, експлойтів під певну версію).
- Рекомендація: Приховати або замінити значення заголовка Server через налаштування вебсервера. Наприклад, у Gunicorn або Nginx можна вимкнути його або змінити на умовне значення ("Secure").
- Фрагмент звіту:

Повідомлення з’явилось під час завантаження ресурсу GET /static/img/hands.png.

3. Відсутній заголовок X-Content-Type-Options

- Рівень ризику: низький
- Опис: У сучасних браузерах при відсутності цього заголовка може виконуватись MIME-sniffing — тобто спроба "вгадати" тип переданого контенту, навіть якщо він вказаний. Це відкриває шлях для потенційного завантаження та виконання шкідливих ресурсів.

Рекомендація: Додати заголовок:

X-Content-Type-Options: nosniff

Це просте налаштування дозволяє браузеру строго дотримуватись вказаного MIME-типу.

– Фрагмент звіту: Вразливість зафіксовано під час завантаження CSS-файлу GET /static/css/registration_steps.css.

4. Відсутній прапорець HttpOnly для cookie

- Рівень ризику: низький
- Опис: Якщо cookie, що використовується для збереження сесії користувача, не має прапорця HttpOnly, до нього може отримати доступ JavaScript-код на сторінці. Це створює ризик викрадення сесії при XSS-атаці.
- Рекомендація: Увімкнути параметр HttpOnly для всіх сесійних cookie в налаштуваннях Django, зокрема:

SESSION_COOKIE_HTTPONLY = True

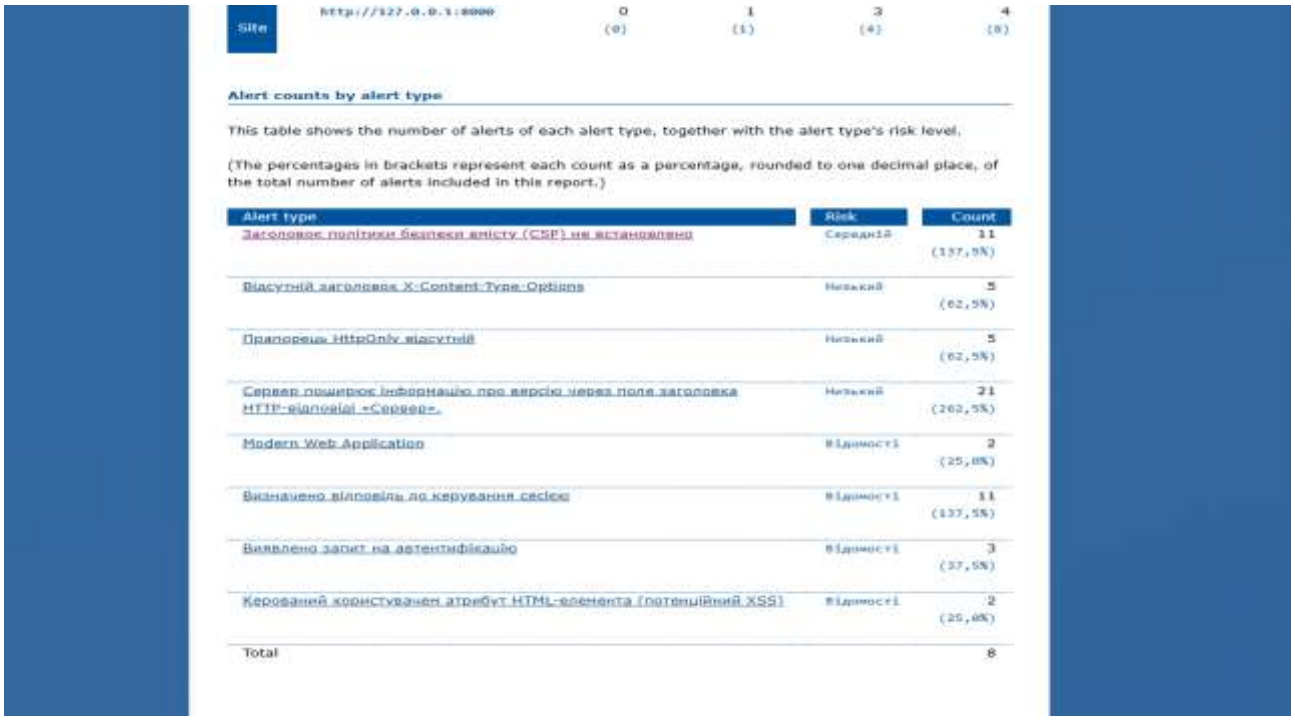
– Фрагмент звіту: Попередження отримано під час запиту на сторінку авторизації GET /accounts/login/.

Інформаційні попередження (4 шт.)

Під час сканування вебзастосунку було виявлено низку інформаційних попереджень, які не є критичними уразливостями, але свідчать про певні особливості його роботи та можуть бути використані в подальшому аналізі. Зокрема, зафіксовано сесійні відповіді, що свідчать про активне використання механізмів керування сесією, а також запити автентифікації, які підтверджують обробку логін-форм на стороні сервера. Крім того, на окремих сторінках виявлено HTML-атрибути, що можуть приймати дані користувача, що потенційно створює умови для XSS-атак. Також встановлено, що застосунок є сучасним SPA або MVC вебдодатком — ця характеристика не є уразливістю, однак вона враховується сканером для точнішої інтерпретації результатів перевірки.

Проведене тестування підтвердило базову стабільність та працездатність основного функціоналу веб-застосунку. Ручна перевірка ключових сценаріїв користувацької взаємодії (реєстрація, авторизація, редагування профілю, свайп профілів, чат) засвідчила коректну роботу системи згідно з поставленими вимогами. Усі функції виконувалися без критичних помилок, а також із належним збереженням і обробкою даних.

Додатково, за допомогою інструменту OWASP ZAP було проведено пасивне сканування безпеки, що дозволило виявити низку типовий конфігураційних вразливостей, характерних для етапу розробки. Важливо підкреслити, що критичних загроз, таких як SQL-ін’єкції, XSS з можливістю виконання скриптів, витік паролів чи порушення аутентифікації — виявлено не було.



Site	http://127.0.0.1:8000	0 (0)	1 (1)	3 (4)	4 (8)

Alert counts by alert type

This table shows the number of alerts of each alert type, together with the alert type's risk level.

(The percentages in brackets represent each count as a percentage, rounded to one decimal place, of the total number of alerts included in this report.)

Alert type	Risk	Count
Заголовок політики безпеки вмісту (CSP) не встановлено	Середній	11 (137,5%)
Відсутній заголовок X-Content-Type-Options	Низький	5 (62,5%)
Прапорець HttpOnly відсутній	Низький	5 (62,5%)
Сервер поширює інформацію про версію через поля заголовка HTTP-відповіді «Сервер».	Низький	21 (262,5%)
Modern Web Application	Відомості	2 (25,0%)
Визначено відповідь по керуванню сесією	Відомості	11 (137,5%)
Виявлено запит на автентифікацію	Відомості	3 (37,5%)
Керований користувачем атрибут HTML-елемента (потенційний XSS)	Відомості	2 (25,0%)
Total		8

Рисунок 1.8 - Фрагмент звіту сканера OWASP ZAP

Основні зафіксовані попередження стосувалися відсутності захисних HTTP-заголовків, недостатньо жорсткої політики cookie, а також незначного витоку інформації про середовище виконання. Всі ці недоліки можуть бути

усунені на етапі підготовки системи до продуктивного розгортання і не впливають на функціональність застосунку.

Таким чином, можна зробити висновок, що розроблений веб-застосунок на момент завершення основного етапу реалізації є функціонально завершеним і таким, що відповідає базовим вимогам безпеки. Подальша оптимізація системи має включати реалізацію рекомендованих заходів із покращення захисту, а також впровадження автоматизованого тестування для забезпечення стабільності при подальших оновленнях.

ВИСНОВКИ

У ході виконання дипломної роботи було розроблено, протестовано та частково перевірено на безпеку веб-застосунок типу «сайт знайомств», що реалізує базову функціональність для організації взаємодії між користувачами. Основною метою проєкту було створення системи, яка дозволяє людям знайомитися, взаємодіяти через лайки та обмінюватися повідомленнями в режимі реального часу. Усі поставлені завдання були успішно виконані.

У результаті роботи:

- було створено сучасний веб-інтерфейс з особистими анкетами користувачів, можливістю перегляду профілів інших людей та функцією свайпу;
- реалізовано механізм взаємних вподобань (матчів), який відкриває можливість для комунікації через чат;
- додано функціонал реєстрації, авторизації, редагування профілю, надсилання повідомлень і виходу з облікового запису;
- забезпечено збереження даних користувачів за допомогою бази даних та створено просту, але ефективну логіку обробки взаємодій.

Для перевірки працездатності системи було проведено ручне функціональне тестування ключових сценаріїв, а також базове пасивне сканування безпеки за допомогою інструменту OWASP ZAP. Отримані результати підтвердили коректну роботу реалізованих компонентів та дозволили виявити низку незначних вразливостей, характерних для середовища розробки. Їх усунення можливе в межах наступних ітерацій проєкту.

Таким чином, виконана дипломна робота не лише продемонструвала практичні навички розробки веб-застосунків, а й дозволила ознайомитися з основами забезпечення інформаційної безпеки та підтримки зручного користувацького досвіду. Подальший розвиток системи може включати масштабування функціоналу, адаптацію до мобільних пристроїв, впровадження рекомендаційних алгоритмів та більш глибокий захист персональних даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bootstrap. Офіційна документація [Електронний ресурс]. – Режим доступу: https://getbootstrap.com (дата звернення: 22.04.2025).
2. Content Security Policy (CSP). MDN Web Docs [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP (дата звернення: 30.04.2025).
3. CSS: Cascading Style Sheets. MDN Web Docs [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 01.05.2025).
4. Django documentation [Електронний ресурс]. – Режим доступу: https://docs.djangoproject.com (дата звернення: 12.04.2025).
5. HTML: HyperText Markup Language. MDN Web Docs [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 29.04.2025).
6. Hunt, T. Shhh… don’t let your response headers talk too much [Електронний ресурс]. – Режим доступу: https://www.troyhunt.com/shhh-dont-let-your-response-headers/ (дата звернення: 03.05.2025).
7. JavaScript. MDN Web Docs [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 07.05.2025).
8. JavaScript Event Loop: Як працює механізм асинхронності. MDN Web Docs [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript/EventLoop (дата звернення: 10.05.2025).
9. OWASP Cheat Sheet Series [Електронний ресурс]. – Режим доступу: https://cheatsheetseries.owasp.org (дата звернення: 18.04.2025).
10. OWASP Top Ten Web Application Security Risks [Електронний ресурс]. – Режим доступу: https://owasp.org/www-project-top-ten/ (дата звернення: 14.05.2025).

11. OWASP ZAP: Documentation [Електронний ресурс]. – Режим доступу: https://www.zaproxy.org/docs (дата звернення: 20.04.2025).

12. Python documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org (дата звернення: 27.04.2025).

13. SQLite documentation [Електронний ресурс]. – Режим доступу: https://sqlite.org/docs.html (дата звернення: 08.05.2025).

14. Web security for developers. MDN Web Docs [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/Security (дата звернення: 05.05.2025).

15. Online Dating Market Overview [Електронний ресурс]. – Режим доступу: https://www.marketresearchfuture.com/reports/online-dating-market-11692 (дата звернення: 27.12.2024).

16. Online Dating Services Market Size and Forecast 2025 to 2034 [Електронний ресурс]. – Режим доступу: https://www.precedenceresearch.com/online-dating-services-market (дата звернення: 1.01.2025).

17. Dating App Stats, Revenue & Usage 2025 Overview [Електронний ресурс]. – Режим доступу: https://prioridata.com/data/dating-app-stats/ (дата звернення: 30.12.2024).

ДОДАТКИ

Додаток А Лістинг коду файлу з моделями models.py

```
from django.contrib.auth.models import AbstractUser
from django.db import models
from django.conf import settings
from datetime import date

class User(AbstractUser):
    RELATIONSHIP_CHOICES = [
        ('single', 'Одинокий'),
        ('in_a_relationship', 'У стосунках'),
        ('married', 'Одружений/Заміжня'),
        ('complicated', 'Складні стосунки'),
    ]

    INTEREST_CHOICES = [
        ('communication', 'Шукаю спілкування'),
        ('relationship', 'У пошуках стосун
ків'),
        ('explore', 'Просто досліджую'),
    ]

              interests      =      models.CharField(max_length=20,
choices=INTEREST_CHOICES, blank=True, null=True)
    bio = models.TextField(blank=True)
     avatar  =  models.ImageField(upload_to='avatars/',  blank=True,
null=True)
            display_name     =     models.CharField(max_length=150,
verbose_name="Ім’я", blank=True)

    gender = models.CharField(
        max_length=10,
        choices=[('M', 'Чоловік'), ('F', 'Жінка')],
        blank=True
    )

    relationship = models.CharField(
        max_length=20,
        choices=RELATIONSHIP_CHOICES,
        blank=True
    )

    # Видаляємо поле age, воно буде через @property
    # age = models.PositiveIntegerField(null=True, blank=True)
```

```
        profile_image   =   models.ImageField(upload_to='profiles/',
blank=True, null=True)

    city = models.CharField(max_length=100, blank=True)
    phone = models.CharField(max_length=20, blank=True)
    birth_date = models.DateField(null=True, blank=True)  # Лише тут
залишаємо поле

    job = models.CharField(max_length=100, blank=True)
    height = models.PositiveIntegerField(null=True, blank=True)
    goals = models.TextField(blank=True)
    education = models.CharField(max_length=100, blank=True)
        status   =   models.CharField(max_length=255,   blank=True,
null=True)
    personality = models.TextField(blank=True)
    hobbies = models.TextField(blank=True)

    @property
    def age(self):
        if self.birth_date:
            today = date.today()
            age = today.year - self.birth_date.year
             if (today.month, today.day) < (self.birth_date.month,
self.birth_date.day):
                age -= 1
            return age
        return None

    def __str__(self):
        return self.display_name or self.username

class Like(models.Model):
        from_user   =   models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE, related_name='likes_given')
         to_user    =    models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE, related_name='likes_received')
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"{self.from_user} liked {self.to_user}"

class Message(models.Model):
      chat  =  models.ForeignKey('Chat',  on_delete=models.CASCADE,
related_name='messages', null=True, blank=True)
```

```
        sender = models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE, related_name='sent_messages')
        recipient = models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE, related_name='received_messages')
    content = models.TextField()
    timestamp = models.DateTimeField(auto_now_add=True)
    is_read = models.BooleanField(default=False)

    def __str__(self):
                return f"{self.sender} → {self.recipient}:
{self.content[:20]}"

class Chat(models.Model):
    participants = models.ManyToManyField(settings.AUTH_USER_MODEL,
related_name='chats')
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
            return f"Chat: {' & '.join([u.username for u in
self.participants.all()])}"

    @classmethod
    def get_or_create_private_chat(cls, user1, user2):
                                                chat =
cls.objects.filter(participants=user1).filter(participants=user2).
first()
        if not chat:
            chat = cls.objects.create()
            chat.participants.add(user1, user2)
        return chat, not chat.messages.exists()

from django.db import models
from django.contrib.auth import get_user_model

User = get_user_model()

class Photo(models.Model):
       user = models.ForeignKey(User, on_delete=models.CASCADE,
related_name='photos')
    image = models.ImageField(upload_to='user_photos/')
    uploaded_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"Photo {self.id} for {self.user.username}"
```

Додаток Б ER-Діаграма взаємодії бази даних

