Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мережних технологій факультету інформатики

РОЗРОБКА ВЕБ-САЙТУ ЗНАЙОМСТВ

Текстова частина до курсової роботи

за спеціальністю «Комп’ютерні науки»

Керівник курсової роботи:

к. ф.-м. н. Гречко А.В.

Виконала студентка:

Чурілова А.С.

Київ 2021

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мережних технологій факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри інформатики,

к. ф.-м. н. С. С. Гороховський

_____

(підпис)

“____”_____2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студента Чурілової Анни Сергіївни факультету інформатики 3 курсу

Тема: Розробка веб-сайту знайомств

Дата видачі “___” _____ 2021 р. Керівник _____

(підпис)

Завдання отримав _____

(підпис)

Календарний план виконання курсової роботи

Тема: Розробка веб-сайту знайомств

№ п/п	Назва етапу курсової роботи	Термін виконання
1.	Обрання теми та отримання завдання курсової роботи.	28.09.2020
2.	Ознайомлення з тематичною інформацією, вибір технологій розробки	20.01.2021
3.	Аналіз аналогів розробки	03.02.2021
4.	Розробка технічного завдання та алгоритму роботи веб-сайту	16.02.2021
5.	Створення архітектури баз даних	20.02.2020
6.	Програмування інтерфейсу веб-сайту	03.03.2021
7.	Подання першої версії інтерфейсу науковому керівник	14.03.2021
8.	Програмування бекенд частини веб-сайту	20.03.2021
9.	Подання першої версії роботи науковому керівнику	08.05.2021
10.	Коригування роботи згідно зауважень наукового керівника	09.05.2021
11.	Створення презентації	16.05.2021
12.	Захист курсової роботи	З 24.05.2021

Студент Чурілова А.С.

Керівник Гречко А.В.

“_____” _____

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ

1. **Чат** - засіб для обміну текстовими повідомленнями між користувачами сайту в реальному часі.
2. **Свайп-картка** - елемент веб-сторінки, що може бути прибраний шляхом перелистування вліво або вправо, в залежності від вибору користувача.
3. **Валідація** – перевірка введеної користувачем інформації на задоволення певних критеріїв.
4. **Месенджер** - функція, що дозволяє користувачам обмінюватися текстовими повідомленнями в режимі реального часу
5. **Дейтингові сервіси** - сервіси, які мають на меті організацію побачень (від англ. “date” - побачення) для своїх користувачів.
6. **Хук** - React - компонент, що дає змогу використовувати стани(state) та інші можливості цієї бібліотеки без необхідності створення класів.

ВСТУП

“Stars Aligned”- соціальна мережа знайомств, яка надає можливість знайти собі співрозмовника, друзів або партнера для романтичних відносин. Назва говорить сама за себе - “зірки зійшлись”, адже при користуванні додатками-аналогами у людей дійсно складається враження, що вони зустріли свого друга чи кохану людину випадково, що цьому сприяли зірки.

На сьогоднішній день існує декілька схожих платформ для знайомств, що мають свої переваги та недоліки, тому проекту “Stars Aligned” є куди розвиватися. “Stars Aligned” чітко підбирає потенційно цікавих людей, ґрунтуючись на інтересах кожного користувача.

Цільовою аудиторією сайту є люди будь-якого віку старше 18 років, що хочуть знайти своє кохання, або просто поспілкуватись із людьми, з якими вони мають спільні інтереси.

Обираючи тему курсової роботи я враховувала той факт, що людям потрібна друга половинка, вони хочуть кохати та відчувати себе коханими. Так завжди було і буде, а в епоху карантинних обмежень, коли можливості спілкування (особливо наживо) суттєво скоротилися, організація знайомств онлайн стає ще більш актуальним завданням.

Завданням курсової роботи є створення веб-сайту для всіх типів користувачів старших 18 років, які хочуть знайти справжнє кохання за допомогою застосунку з красивим, зручним та “інтуїтивним” інтерфейсом, яким можна користуватися без сторонньої допомоги.

Мета курсової роботи - розробити найбільш комфортний веб-сайт, завдяки якому користувачам інтернету буде надана можливість у зручний спосіб знайти споріднених духом людей.

Предметом дослідження є способи пошуку друзів та кохання в епоху соціальних мереж та інтернету. Увагу зосереджено на дослідженні можливостей того, як можна допомогти користувачеві розповісти про себе якомога краще та більш точно розподіляти людей за спільними інтересами, поглядами і захопленнями.

Об’єктом дослідження є наявність дейтингових застосунків, як-от “Tinder”, “Emilydates”, “OneAmour” тощо, глибоке вивчення сайтів на дану тематику, можливості просування і поліпшення якості роботи сайту, що допоможе привабити якомога більшу кількість інтернет-користувачів.

За результатами дослідження виявлено, що всі сервіси зі схожою тематикою мають подібні функції, проте залишається актуальною проблема зручності користування такими застосунками. Найбільш популярний аналог - мобільний додаток “Tinder”, що залишається в топі серед своїх конкурентів, проте все ж таки залишається мобільним додатком. “Stars Aligned” - веб-сайт, де у зручний спосіб, на великому екрані свого комп’ютера, люди можуть переглядати потенційні пари та спілкуватись один з одним.

Веб-сайт написаний на мові JavaScript, зокрема її бібліотеці React. Процес написання коду відбувався в середовищі розробки Microsoft Visual Studio Code. Інформація про користувачів зберігається документоорієнтованій СКБД для NoSql баз даних Firestore, а файли із зображеннями - в сховищі Firebase Cloud Storage. Робота з ними відбувалась за допомогою інструменту Firebase Console.

Текстова частина до курсової роботи має вступ, перелік термінів та умовних позначень, список використаних джерел, додатки та три розділи: "Аналіз предметної області. Постановка завдання курсової роботи", "Теоретичні відомості" та "Опис реалізації програмного продукту".

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАВДАННЯ КУРСОВОЇ РОБОТИ

1.1 Аналіз сучасного стану питання та обґрунтування теми

Способи зв'язку для спілкування, за весь час людства, ніколи не стояли на місці, навпаки, вони з кожним днем еволюціонують, методи зв'язку змінюються з розвитком сучасних технологій. Природно, до початку епохи інтернету, люди спілкувалися в основному наживо, далі почалася ера соціальних мереж, і люди почали комунікувати переважно в письмовій формі. На сьогоднішній день месенджери дають нам повну можливість доносити наші думки і слова в будь-якому вигляді, ми можемо як відправляти відео/голосові повідомлення, так і писати просто від себе, надсилати різноманітні емоджі та картинки. І це прекрасно, адже за лічені хвилини ми можемо зв'язатися з людьми з будь-якого куточку світу. У такому випадку постає питання: якою є ситуація з налагодженням нових контактів та знайомствами у наш час? Чи можна тут завдяки певним онлайн-сервісам замінити “живі” знайомства?

Якщо розбирати все по порядку, то, безумовно, сьогодні знайомства вживу стали відбуватися рідше ніж будь-коли. Добре це чи погано - кожен вирішує для себе сам. Хтось вважає це вторгненням в особистий простір, а хтось - сміливим вчинком. Проте якщо поглянути на ситуацію зі сторони технологічних відкриттів, то можна вважати, що дані сервіси для знайомств - це справжній прорив для людства. Практично кожній людині надана можливість познайомитися з іншою людиною зі схожими інтересами або з однодумцем з іншої країни.

Однією значною перевагою знайомств в дейтингових застосунках є те, що люди дізнаються про симпатію в той момент, коли вона є взаємною. Тобто дві людини в момент знайомства вже знають, що вони мають спільні інтереси і що вони один одному сподобались за критерієм зовнішності. Це пришвидшує розвиток відносин, одразу даючи зрозуміти партнеру, що він тобі симпатизує та навпаки.

Тож можемо зробити висновок, що дейтингові сервіси - найкращий спосіб заведення знайомств у наш час. Користувач відразу може відсіювати нецікавих йому людей, і не витрачати на це багато свого часу. Веб-сервіс “Stars Aligned” допоможе у зручний та швидкий спосіб знайти споріднену духом людину та/або завести довгострокові романтичні відносини чи міцну дружбу.

1.2 Огляд існуючих аналогів розробки

Безсумнівно, сайти знайомств стали невід'ємною частиною знайомств у інтернеті. Але так само варто згадати, що є різні типи сайтів знайомств, які відрізняються між собою дизайном, функціями або ж користувачами за різними категоріями (наприклад, вік, цілі спілкування тощо).

Роглянемо приклади:

- Love.ua [1]

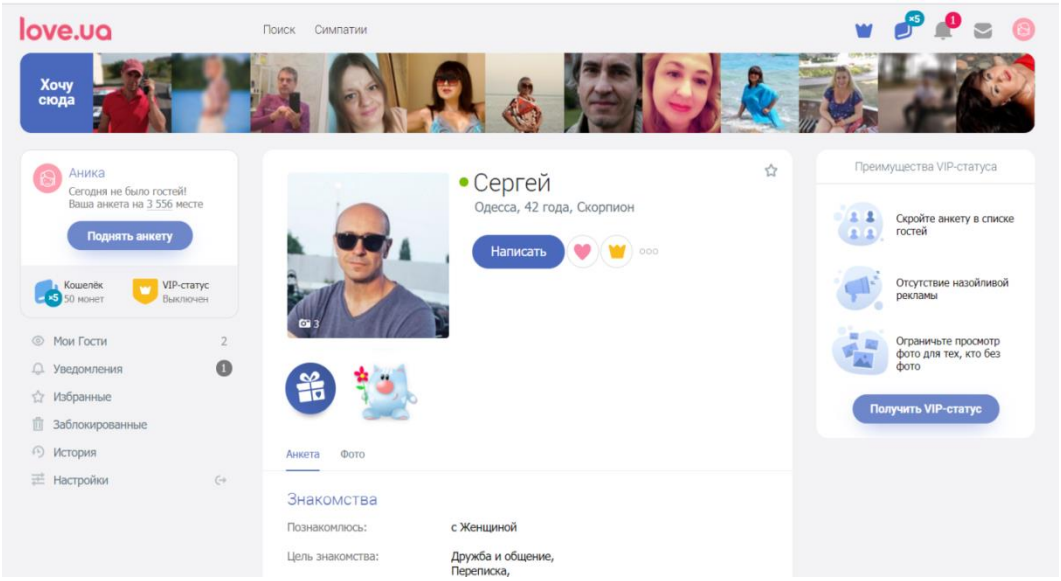

Рисунок 1.2.1 – Сайт знайомств Love.ua

Український сайт для знайомств, де можна зазначити свій настрій, а вже на основі цього настрою підбирати собі партнера по листуванню та, можливо, для романтичних відносин. Даний сервіс працює по всій Україні;

- Tinder [2]

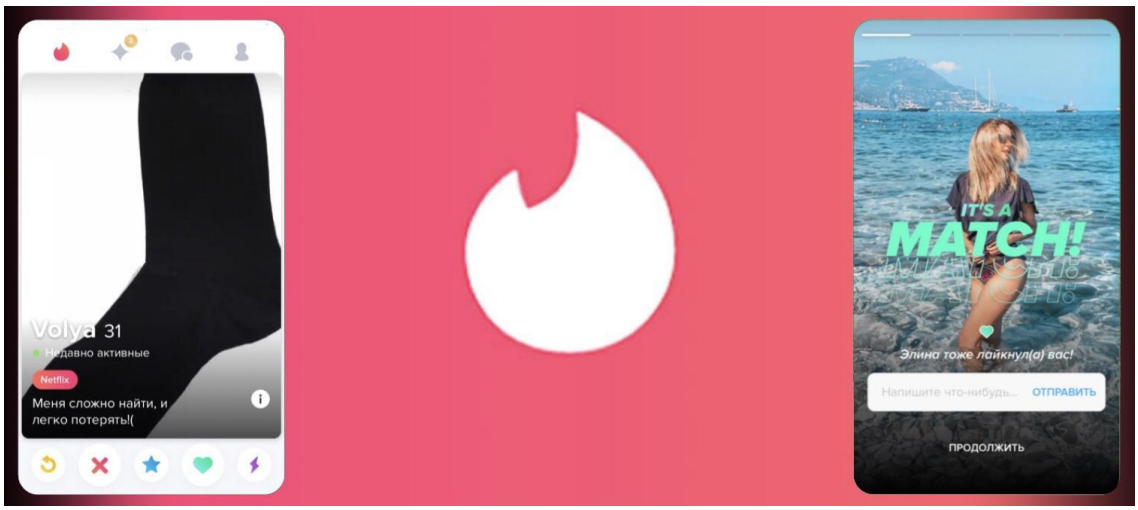

Рисунок 1.2.2 - Tinder

Tinder - це один із найпопулярніших сервісів для знайомств у світі, яким регулярно користуються близько 50 мільйонів людей. На відміну від попереднього сервісу знайомств, Tinder не може похвалитися такою ж статистикою пар, які пов'язали свої життя надовго. Проте за використання цього додатку шанс знайти співрозмовника набагато вищий, ніж на будь-якій іншій платформі. “Stars Aligned” найбільш наближений до інтерфейсу і функцій даного сервісу;

- Emilydates [3]

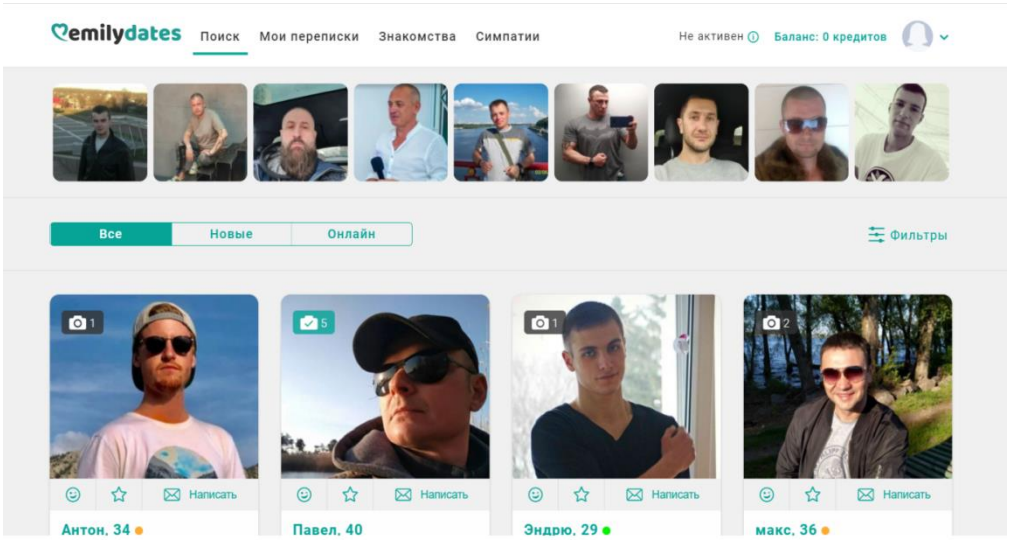

Рисунок 1.2.3 - Emilydates

На веб-сайті Emilydates присутня не зовсім звичайна система спілкування: після того, як користувач реєструється, йому буде доступно 15 контактів, після яких за кожен новий контакт йому буде надаватися ще по одній можливості починати новий діалог. Зазвичай, аудиторія цього сайту не потребує серйозних відносин, їх головна ціль - розкішне життя і незабутній відпочинок. За відгуками користувачів цієї платформи (як жінок, так і чоловіків), на цьому сайті у них часто виходить знайти однодумців;

- OneAmour [4]

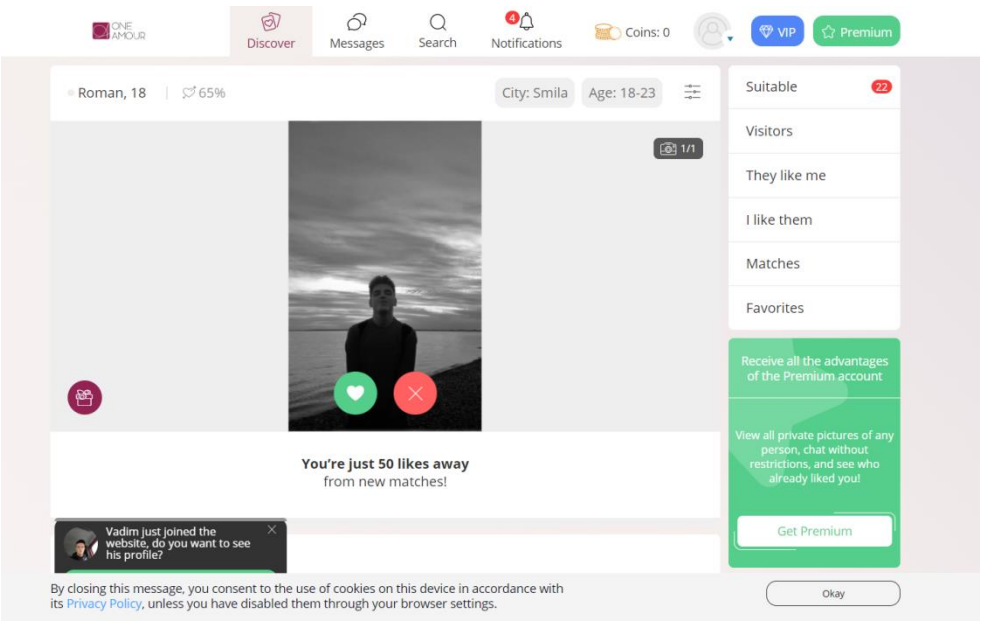

Рисунок 1.2.4 - OneAmour

Цей сайт для знайомств абсолютно звичайний, без будь-яких додаткових функцій, але відразу після реєстрації з'являються боти, від яких моментально надходять пропозиції про знайомство. У більшості випадків, - це лише спроба швидше змусити людину придбати Преміум-версію акаунту.

1.3 Постановка завдання

В сучасному світі стало дуже популярним та зручним знайомство в інтернеті, але, на жаль, веб-сервісів, де у зручний спосіб можна знайти собі друга або другу половинку, не так багато.

Тому ідея моєї курсової роботи полягає в тому, щоб створити веб-сайт, який став би популярним серед користувачів, які бажають знайти свою споріднену душу, та полегшив би для них цей процес.

Завдання моєї роботи в курсовому проекті:

1. Знайти та дослідити найпопулярніші у світі технології веб-програмування. Вибрати ті, які підходять найкраще для написання веб-сайту даного типу.

2. Знайти найпопулярніші аналоги розробки, проаналізувати зручність їх інтерфейсу, інтуїтивність користування, функціонал, сильні та слабкі сторони кожного. Визначити спільні риси кожного сервісу для полегшення розробки курсового проекту.

3. Розробити веб-сайт знайомств, який буде мати такі функції:

- реєстрація та авторизація на сайті;
- заповнення анкети користувача;
- завантаження своїх фотографій;
- перегляд профілю потенційного друга/партнера;
- перегляд потенційних пар за ознакою спільних інтересів та статтю;
- створення пар;
- перегляд списку пар;
- можливість спілкування зі своєю парою у месенджері;
- перегляд принципів роботи сайту;
- вихід з акаунту;
- видалення акаунту.

РОЗДІЛ 2: ТЕОРЕТИЧНІ ВІДОМОСТІ

2.1 Види веб-сайтів

Щодня ми використовуємо змістовну систему пов'язаних даних, що має особливу адресу, відому нам під назвою – веб-сайт. Завдяки йому, віртуальний світ для середньостатистичної людини став більш доступним. У ньому можна шукати, писати, читати різну інформацію, можна слухати музику, працювати, спілкуватися і навіть вести свій блог. Можливості використання веб-сайтів безмежні.

Існують безліч різних типів веб-сайтів, які можна класифікувати за способом застосування. Основні з них[5][6]:

1. Веб-сайти соціальних мереж

На сьогоднішній день кількість людей, що користуються соціальними мережами досягає близько 3 мільярдів. Це дає нам зрозуміти, що популярність та актуальність веб-сайтів такого виду постійно зростатиме. Соціальні мережі дають людям змогу ділитися своєю особистою інформацією, слідкувати за життям інших користувачів та спілкуватись один з одним.

2. Розважальні веб-сайти

Сайти розважального характеру дають змогу, наприклад, переглядати художні твори, музику, фільми і драматичні серіали, а також онлайн-ігри для того, щоб розважити користувача.

3. Онлайн-магазин

Це веб-сайт, розроблений з метою надання функцій здійснення покупок товарів та послуг в онлайн режимі. Користувачі можуть купувати необхідні їм речі в будь-який час, без безпосереднього звернення до продавця.

4. Веб-сайти портфоліо

Веб-сайт портфоліо дозволяє користувачам, що займаються різними видами творчої діяльності, показувати свої роботи. Цільовою аудиторією таких сайтів є художники, письменники, дизайнери, кінематографісти, фотографи тощо.

Веб-сайт “Stars Aligned” найбільше підходить під опис веб-сайту соціальної мережі. На цьому сайті можна викладати фотографії і створювати свій профіль, переглядати чужі профілі і відзначати ті, які сподобалися, а також спілкуватися з іншими користувачами. При цьому користувач сайту викладає там достатню кількість інформації про себе, щоб знайти однодумців і людей зі схожими інтересами та поглядами.

При використанні сайту «Stars Aligned», можна звернути увагу на те, що під час реєстрації свого профілю, годі й думати про те, що можна про себе розповісти в контексті захоплень та інтересів. Сайт сам запитує, які ви маєте хобі і чим цікавитеся. Таким чином, на основі ваших відповідей сайт буде влучно підбирати вам потенційних партнерів або друзів. Потрібно вибрати мінімум 3 захоплення і чим більше ви оберете, тим більший шанс того, що можете зустріти однодумця.

2.2 Бази даних та СКБД

База даних - це збережений в електронному вигляді, організований набір структурованої інформації або даних.[7] База даних може містити різноманітну інформацію з будь-якої предметної області в світі. Наприклад

назви деталей, квітів, імена та прізвища людей, інформація про сімейний стан, прогноз погоди тощо.

База даних NoSQL стала основою при розробці мого сайту. NoSQL - це метод для масштабованої реалізації інформаційного сховища (або бази), який має хибку модель даних. [8]

При додаванні інформації у базу даних NoSQL, як правило немає потреби заздалегідь визначати її жорстку схему. Як наслідок, такий тип даних можна змінити будь-коли, не порушуючи роботу всієї програми. Це не тільки забезпечує економію часу програмістам, а також надає можливість сервісам, які вони розробляють, ставати більш

СКБД (система керування базами даних) - це програмне забезпечення, яке використовується задля забезпечення широкого спектру дій з базами даних. До таких дій відносяться: визначення, створення, підтримка та здійснення контрольованого доступу до бази даних.

Види NoSQL-СКБД [9]:
1) “Ключ-значення”

Представляють собою асоціативний масив. За допомогою таких баз даних застосунок легко масштабувати. Відсутні зв'язки між сутностями, схеми баз даних, немає обмежень на кількість елементів та інших частин, окрім обмежень, які накладаються на асоціативні масиви. Природньо, що такі бази даних використовуються в хмарних технологіях.

Серед мінусів таких баз даних є те, що більшість простих та звичних операцій зі значеннями бази дуже ускладнена, адже, наприклад, виконання пошуку за значеннями займе доволі багато часу. Очевидно, що це матиме негативні наслідки для продуктивності аналізу даних у базі.

2) Документоорієнтовані

Документоорієнтована база даних - це система оперування та зберігання даних, яка має структуру у вигляді лісу або дерева.

Завдяки вузлам дерева, які містять дані, можна здійснювати швидкий пошук, навіть не дивлячись на складну структуру бази даних. Все завдяки тому, що при додаванні даних до сховища, вони вносяться в індекс бази.

Сама структура дерева дозволяє нам створювати окремі колекції документів, які об‘єднані за певними критеріями, а також присутні механізми пошуку надають нам змогу знаходити документи та його частини.

3) Графові

Графові бази даних передбачають графову структуру даних, тобто розробник має можливість отримати найкращі результати у продуктивності, а також забезпечують простоту внесення змін та подання інформації. Для полегшення роботи з великими графами застосовують алгоритми, які розраховують на часткове пересування їх в оперативну пам‘ять.

4) Bigtable-подібні

Bigtable-подібні бази даних включають в себе дані, які впорядковані у вигляді розрідженої матриці. Рядки та стовпці цієї матриці застосовуються у якості ключів. Вони використовуються задля вирішення веб-індексування та інших проблем. Загалом, їх застосовують там, де потрібні великі обсяги даних.

Отже, існує безліч СКБД, що використовують різні технології, зберігають інформацію в різних виглядах та призначені для задоволення різних інформаційних потреб користувача. Обрана мною СКБД Cloud Firestore відноситься до документоорієнтованого типу баз даних, тобто зберігає дані у вигляді дерева або лісу.

2.3 Переваги обраної СКБД

Сьогодні існує безліч систем керування базами даних, найвідоміші з них – Oracle, SQLite, PostgreSQL, MySQL, MongoDB та порівняно нова (створена в 2012 році компанією Firebase) Cloud Firestore . Однак, оскільки проект побудований на основі NoSQL бази даних, мій вибір впав на Cloud Firestore. Cloud Firestore - це хмарна база даних, яка дає змогу розробникам зберігати, діставати та синхронізувати інформацію з неї. Нею легко користуватись, тому що вона надає зручні засоби взаємодії.

Як було зазначено вище, Cloud Firestore є продуктом компанії Firebase. Firebase працює на платформі Google, що є одним із найбільш відомих та надійних брендів у світі технологій. За допомогою Firebase можна охопити всі етапи розробки, а сама платформа містить всі необхідні інструменти для розробки, випуску та підтримки додатків, що є величезною перевагою Firebase. Однією з доволі сильних переваг Firebase є те, що він забезпечує середовище, в якому відсутні сервери, тобто відсутня необхідність управління їх інфраструктурою. У зв’язку з цим майже не виникають проблеми з масштабуванням серверів. Firebase покращує безпеку даних за рахунок постійного резервного копіювання. Таким чином, Firebase забезпечує захист додатків від втрати інформації.

Переваги Cloud Firestore[10] :

- Дані зберігаються у вигляді документів, що організовані в певні колекції залежно від вибору класифікації. Такі документи можуть містити в собі не тільки колекції, а й доволі складні об'єкти, що є вкладеними.
- У Cloud Firestore присутня вкладена індексація запитів. Ця перевага значно підвищує продуктивність отримання даних зі сховища

- Cloud Firestore підтримує синхронізацію та оновлення даних в режимі реального часу.
- Cloud Firestore проводить автоматичне кешування даних, які використовуються в ході роботи програми. Ця перевага дає змогу додавати, переглядати і робити запит до даних навіть в офлайн-режимі. Проте в момент, коли пристрій починає працювати в онлайн-режимі, Cloud Firestore проводить синхронізацію всіх локальних змін.
- Cloud Firestore базується на потужній інфраструктурі, що робить розроблюваний продукт легко масштабованим горизонтально та вертикально. Як наслідок, додаток зможе прийняти та обробити навантаження у вигляді мільйонів користувачів.

РОЗДІЛ 3. ОПИС РЕАЛІЗАЦІЇ ПРОГРАМНОГО ПРОДУКТУ

3.1 Аналіз технічного завдання:

Суть проекту “Stars Aligned” в тому, щоб допомагати людям знаходити один одного, знайомитися, товаришувати та, можливо, заводити романтичні стосунки. Перевага такого веб-сайту в тому, що він допомагає знайти співрозмовника за критерієм за спільних інтересів та статтю. Останнє особливо важливо для людей із нетрадиційною сексуальною орієнтацією, адже при реєстрації на платформі користувач обирає стать людей, яких він хоче бачити у списку запропонованих пар, а також вказує, людям якої статі він хоче бути показаним. Таким чином багато “лишніх” кандидатів відсіюється ще до початку відбору.

Разом зі зручним інтерфейсом програми, в деяких місцях користувачам мають бути надані підказки та повідомлення про помилки, що допоможе покращити розуміння проблем, якщо вони виникають та повідомити про успішність певних операцій. У програмі це забезпечено тим, що присутні наступні функції:

- валідація форм реєстрації, авторизації та створення профілю – відображення причини, чому введені дані не підходять: “Приходьте, коли Вам буде 18 років!”, “Помилка! Неправильний пароль!”, “Помилка! Користувач не знайдений!” тощо;
- підказки на сторінці реєстрації та при заповненні анкети – “Введені паролі не співпадають”, “Невірний формат електронної пошти”, мінімальна кількість введених інтересів тощо;

- відображення повідомлення про успіх/невдачу збереження даних та створення пар – “Файл успішно завантажений”, “Ура! Ви зробили пару.”.

Користувачі, що пройшли авторизацію, мають доступ до таких функцій (для неавторизованих користувачів доступу до платформи не передбачається):

- здійснювати відбір потенційних пар;
- зв’язуватися з новоствореними парами у месенджері;
- видаляти свій профіль;
- переглядати інформацію про новостворених пар;
- переглядати інформацію про принцип роботи платформи;
- здійснювати вихід із аканту.

3.2 Обґрунтування алгоритму й структури програми

Сайт побудовано у вигляді односторінкового додатку (single page application). Додаток SPA[11] - це одна єдина сторінка сайту, що динамічно змінює свій вигляд залежно від дій користувача. Веб-сайт не завантажує щоразу додаткові сторінки з сервера, а динамічно змінює себе, що дозволяє заощадити час на повторне підтягнення однакових елементів. Щоб візуально зобразити структуру можливих виглядів сторінки застосунку, було побудовано схему, яку можна переглянути на рисунку 1 у додатку А. Кожен об’єкт блок-схеми - це вигляд сторінки, який з’являється у результаті певних дій користувача.

Загалом, користувачу доступні такі варіації сторінки:

- За шляхом “/login” відкривається сторінка “Авторизація”, на якій відбувається авторизація користувачів або переадресація на сторінку реєстрації (у випадку, якщо користувач ще не авторизований). Присутні 2 обов’язкових поля вводу даних - “Email адреса” та “Пароль”, кнопка “Вхід” та посилання “Створити акаунт”;
- За шляхом “/login#” відкривається сторінка “Реєстрація” з полями для введення логіну, паролю і повторення паролю, кнопкою “Створити” та посиланням “Увійти в акаунт”;
- За шляхом “/register” відкривається “Анкета користувача” з кількома обов’язковими для заповнення полями: ім’я користувача, прізвище, дата народження, сексуальна орієнтація, додавання фото (включно із перетягуванням картинки в область завантаження фото) тощо. Також присутня кнопка “Створити”. Варто зазначити, що у випадку, коли користувач не завершив авторизацію (тобто зареєстрував електронну пошту, але не заповнив анкету), при вході в акаунт йому буде відображатися сторінка “Анкета користувача” (переглянути реалізацію цієї логіки можна на рисунку 11 в додатку В);
- За шляхом “/” відкривається “Головна” - головна сторінка сайту, на якій відображені свайп-картки з інформацією про потенційних пар (фотографія та короткий опис). Також присутнє меню, яке містить такі пункти: “Головна”, “Пари”, “Вихід”, “Як це працює”, “Видалити акаунт”. Варто зазначити, що авторизованому користувачу при введенні будь-якого шляху відкриється сторінка “Головна”;
- За шляхом “/chat” відкривається сторінка “Пари”, на якій відображено список створених пар;
- За шляхом “/chat/:userId” відкривається сторінка “Месенджер” - чат з певною людиною. Праворуч присутня кнопка перегляду інформації про новостворену пару, ліворуч - меню;

- За шляхом “/howitworks” відкривається сторінка “Як це працює” із описом принципів роботи веб-сайту;

Варто зазначити, що неавторизованого користувача при введені будь-якого шляху в пошуковий рядок сайту буде перенаправлено на сторінку реєстрації.

3.3 Обґрунтування вибору засобів розробки

Проаналізувавши найзручніші, найпопулярніші у світі технології розробки веб-сайтів, я зупинилася на тих, що допоможуть найкраще реалізувати мою програму.

Процес створення платформи відбувався у IDE Microsoft Visual Studio Code, що є простим редактором коду та має достатньо сильні інструменти розробника. Показ сайту відбувається у браузері Google Chrome.

Проект поділяється на дві частини: клієнтську та базу даних.

Клієнтська частина:

- CSS - мова, яка відповідає за зовнішній вигляд сторінок HTML. Велика кількість сучасних сайтів реалізовані на основі технологій HTML та CSS.
- HTML - це мова гіпертекстової розмітки, використовуючи яку, верстальники створюють “каркас” для сторінок сайту.
- JavaScript – мова програмування, на якій реалізовано багато додатків з метою додавання їм інтерактивності і швидкості.
- React [12] [13] - бібліотека мофи JavaScript, що використовується для створення інтерфейсу користувача. React використовує JSX (JavaScript + XML) - легкозасвоюваний та зрозумілий метод визначення інтерфейсу програми. При використанні React програмісту достатньо описати, як різні частини інтерфейсу мають відображатись на екрані в

різних станах. React буде оновлювати їх щоразу, коли в програмі відбуваються якісь зміни. React дає змогу створювати та керувати (переміщувати між різними проектами та в різні місця одного проекту) компоненти, а також об’єднувати їх в складні інтерфейси користувача.

- Material-UI [14] - React - компонент, що пришвидшує та полегшує веб-розробку. Для використання Material-UI достатньо завантажити його через npm та зробити import потрібного компоненту без необхідності проведення будь-яких налаштувань.
- Ajax [15] - технологія JavaScript, що звільняє від необхідності щоразу, коли змінюється якась інформація на сайті оновлювати всю сторінку, оскільки при використанні ajax оновлюється лише потрібна частина.

База даних:

- Cloud Firestore - це хмарна база даних, яка дає змогу розробникам зберігати, діставати та синхронізувати інформацію з неї. Нею легко користуватись, тому що вона надає зручні засоби взаємодії.
- Firebase Cloud Storage - це ще один тип сховища платформи Firebase. Дані зберігаються в папках, зазвичай використовується для роботи із медіафайлами різних типів.
- Модуль авторизації Firebase. Використовуючи даний модуль можна реалізувати авторизацію користувача за допомогою анонімного входу, акаунтів GitHub, Google, Facebook, та, як у моєму випадку, за допомогою логіна та паролю.

3.4 Опис розробки програми.

В реалізації веб-сайту відсутня серверна частина. Це означає, що робота з базами даних відбувається через клієнта і спілкування проходить лише між клієнтом та сховищем.

Опис файлів вихідного коду:

- “data.js” - відбувається підготовка набору значень та перемішування їх довільним чином для створення фейкових користувачів та відправки їх у базу даних. Переглянути вихідний код даного файлу можна на рисунках 6-9 в додатку В;
- директорія “node_modules”, що зберігає всі бібліотеки, встановлені за допомогою команди npm install бібліотеки, фреймворки та різноманітні компоненти. У вихідному коді папка відсутня, тому необхідно її встановити, адже без неї програма не запрацює;
- директорія “public” містить автоматично згенеровані при створенні React-проекту файли;
- “firebase.js” зберігає конфігурації сховищ платформи Firebase. Із файлу експортуються об’єкти “db” і “storage”, за допомогою яких відбувається робота зі сховищами в усій програмі. Переглянути вихідний код даного файлу можна на рисунку 10 в додатку В;
- “useAuth.js” - хук, що містить функцію useProvideAuth, в якій відбувається реєстрація, авторизація, видалення та вихід із акаунту а також збереження авторизованого користувача в колекцію “users”. Переглянути частини вихідного коду даного файлу можна на рисунку 1 в додатку В;
- “useFirestoreQuery.js” - хук, що дозволяє провести швидку підписку на дані у сховищі Firestore, при цьому не переймаючись управлінням станами;

- “useMatchUser.js” - хук, що організовує контекст. Генерує провайдер, що може зберігати для всіх компонентів об’єкт із даними про пару та використовує хук, що ці дані дістає;
- “useMemoCompare.js” - хук, що допомагає зберігати дані зі сховища в кеш, після чого ці дані потрібно діставати із бази лише в тому випадку, якщо там відбулись зміни;
- директорія “styles” зберігає стилі для кожної сторінки сайту за допомогою CSS;
- “App.js” - задає маршрутизацію, де присутнє використання усіх компонентів програми. Описує дані, які відображаються при перегляді профайлу партнера, створює, наповнює та задає логіку правого і лівого drawer-меню, обробляє кожен його пункт а також адаптує їх на різні розширення екрану. Переглянути частини вихідного коду даного файлу можна на рисунках 11-14 в додатку В;
- “Cards.js” - відбувається фільтрація запропонованих користувачів за критеріями кількості інтересів (більше 3) а також полями “showMe” та “showToMe”. Ще в даному файлі реалізована ідентифікація свайпу - ліворуч - дізлайк, праворуч - лайк та обробка цієї інформації. Переглянути частини вихідного коду даного файлу можна на рисунках 2, 15 та 21 в додатку В;
- “Chats.js” - робить запит до бази даних на отримання списку створених пар даного користувача;
- “ChatScreen.js” - робить запит до бази даних на отримання усіх повідомлень з месенджера певної пари, додавання нових елементів діалогу до сховища в режимі реального часу. Переглянути частини вихідного коду даного файлу можна на рисунках 3 і 4 в додатку В;
- “HowItWorks.js” - сторінка зі статичною інформацією про принципи роботи сайту;

- “index.js” - точка входу в програму. Використовує 3 провайдери(компоненти, що створюють область контексту, тобто зберігають та росповсюджують дані на всі компоненти, що містяться всередині них) та викликає компонент “App.js”. Переглянути вихідний код даного файлу можна на рисунку 16 в додатку В;
- “MatchesList.js” - логіка наповнення сторінки “Пари”. Переглянути вихідний код даного файлу можна на рисунку 17 в додатку В;
- “RegisterForm.js” - створення та відображення форми сторінки “Анкета користувача”. Проводить відправку фотографій користувачів у сховище Firebase Cloud Storage, а решту даних форми в базу Cloud Firestore. Переглянути частини вихідного коду даного файлу можна на рисунку 5 в додатку В;
- “serviceWorker.js” - згенерований автоматично при створенні React - проекту скрипт, що запускається браузером в фоновому режимі, щоб мати доступ до тих функцій, використання яких не потребує взаємодії з користувачем;
- “SignForm.js” - форма введення та перевірки логіну і паролю користувача. Основою є робота з модулем авторизації Firestore. На цій сторінці асинхронно викликаються методи авторизації та реєстрації користувача. Переглянути частину вихідного коду даного файлу можна на рисунку 18 в додатку В;
- “package.json” - файл, що містить в форматі JSON список пакетів, що використовуються в програмі.

3.5 Створення об'єктів і розробка головної програми.

Пункти реалізації поставлених завдань у пункті “1.3 Постановка завдання”:

- Реєстрація користувача.

  Метод реєстрації користувача викликається асинхронно. Форма реєстрації має кілька перевірок на валідність полів - користувач з таким логіном вже зареєстрований у системі, введені паролі не співпадають і неправильний формат електронної пошти, що супроводжується виведенням відповідних повідомлень. Відображення таких повідомлень відбувається за допомогою компоненту Material-UI MuiAlert. Варто зазначити, що перевірка на валідність і асинхронний метод відбуваються за допомогою модулю авторизації Firebase. Переглянути частину вихідного коду даного файлу можна на рисунках 1 і 22 в додатку В;

- Авторизація користувача

  Метод авторизації, як і реєстрації користувача, викликається асинхронно. Форма авторизації має 2 перевірки на валідність полів - відсутність користувача з таким логіном і неправильно введений пароль, що супроводжується виведенням відповідних повідомлень за допомогою компоненту Material-UI MuiAlert. Переглянути частину вихідного коду даного файлу можна на рисунках 1 і 18 в додатку В;

- Заповнення анкети користувача

  При заповненні анкети користувача відбувається валідація обов'язкових полів на заповненість, поле вводу дати народження перевіряється на вік користувача (має бути старшим 18 років). Окрім використання MultiAlert, присутнє також підсвічування відповідних полів червоним кольором та напис про помилку на ньому. Створений користувач зберігається в базі даних за допомогою модулю авторизації Firebase;

- Завантаження фотографій

  При додаванні фотографії у форму анкети користувача, відбувається формування назви даного файлу для подальшого завантаження на Firebase Cloud Storage у папку “images”. Назва файлу зображення співпадає із унікальним ідентифікатором власника фото. За допомогою модуля авторизації Firestore відбувається оновлення даних в базі. У моїй реалізації при записі до сховища присутній атрибут merge: true, що проводить не перезаписування даних про користувача, а доповнення їх. Варто зазначити, що додавання фотографій на форму можна здійснити у 2 різні способи: перетягуванням картинки у область завантаження фото та звичайним вибором файлу через провідник файлів. Переглянути частину вихідного коду цієї логіки можна на рисунку 5 в додатку В;

- Перегляд потенційних пар

  По-перше, список потенційних пар формується з урахуванням фільтрації запропонованих користувачів за критеріями кількості інтересів (більше 3), полями “showMe” та “showToMe”. Також варто зазначити, що кожен претендент з’являється у списку пар тільки один раз. При перегляді пар користувач повинен робити свайпи праворуч або ліворуч в залежності від свого вибору. Обробка цієї дії заключається в тому, що при свайпі праворуч оновлюється масив “likes” даних користувача, а ліворуч - масив “dislikes”. Дане доповнення відбувається також і в базі даних за допомогою модуля авторизації Firebase. Переглянути частину вихідного коду цієї логіки можна на рисунку 21 в додатку В;

- Створення пар

Метод, що формує пару викликається при свайпі користувача праворуч. У цьому методі відбувається формування об’єкту match, що буде записано у сховище. Унікальний ідентифікатор пари формується із id учасників пари, літери яких довільним чином перемішуються і отриманий рядок “обрізається” до 28 символів. Тобто створення пари - це оновлення в базі даних масиву “likes” кожного учасника союзу та додавання новоствореної пари до колекції “matches” у Cloud Firestorage. Переглянути частину вихідного коду цієї логіки можна на рисунках 2 і 21 в додатку В;

- Перегляд списку утворених пар

  Відбувається підписка на оновлення пар в реальному часі за допомогою використання хука useFirestoreQuery. І в залежності від кількості пар генеруються компоненти pairs, що містять необхідну інформацію для передачі компоненту “MatchesList”. В цьому моменті можна проглянути основну перевагу реалізації односторінкового сайту, - при будь-яких змінах стану React моментально перезавантажує сторінку. Це означає, що новостворені пари з’являються на сторінці “Пари” в момент їх створення;

- Месенджер

  По-перше, перевіряється чи текст повідомлення не порожній, потім створюється об’єкт message, що містить пари ключ-значення відповідно до запису документу колекції “messages” у базі даних. По-друге, перед тим, як покласти повідомлення до бази, змінюється стан(state). Таким чином відправлене повідомлення з’являється на екрані моментально, та користувач візуально не бачить можливої затримки при додаванні даних до бази. По-третє, очищується поле вводу та відправляється повідомлення до бази. Врешті-решт, оновлюється поле “resentMessage”

у відповідного документу колекції “matches” бази даних для того, щоб на сторінці “Пари” відображалось останнє повідомлення із діалогу. Переглянути частину вихідного коду цієї логіки можна на рисунку 4 в додатку В;

- Перегляд профілю партнера

  Використано хук useEffect(), що реагує на зміни елементу пари. В цьому хуці прописаний запит до бази даних на отримання користувача, з яким утворено пару. Таким чином коли користувач переходить між різними діалогами, дані про поточну пару оновлюються. Переглянути частину вихідного коду цієї логіки можна на рисунку 20 в додатку В;

- Перегляд принципів роботи сайту

  Реалізація цієї функції представляє собо сторінку зі статичною інформацією про принципи роботи сайту;

- Вихід із акаунту

  При натисненні на кнопку меню “Вихід” відбувається виклик методу signout(), що використовує метод модулю авторизації Firestore signOut(). Після спрацювання цього методу відбувається перенаправлення користувача на сторінку реєстрації;

- Видалення акаунту

  Для того, щоб видалити акаунт, користувач має ввести коректні дані логіну та паролю у відповідну форму, що з’явиться після натиснення на пункт меню “Видалити акаунт”. Видалення акаунту відбувається за допомогою використання модулю авторизації Firebase. За допомогою методу цього модулю reauthenticateWithCredential(credential), що приймає параметр credentials - логін та пароль користувача,

відбувається повторна аутентифікація користувача, що проводить видалення інформації про нього із бази даних.

Варто зазначити, що сайт є адаптивним для різних розширень екрану. По великому рахунку це забезпечено використанням React-компоненту Material-UI, кожен елемент якого автоматично являється адаптованим. Єдиний елемент сторінок, що вимагає окремих налаштувань адаптації є drawer-меню. Його тип “temporary” або “persistent” змінюється в залежності від розміру екрану “sm” або “md” відповідно. Переглянути вихідний код даної логіки можна на рисунку 19 в додатку Д;

3.6 Опис файлів даних та інтерфейсу програми.

Веб-сайт використовує 2 типи сховищ - Cloud Firestore та Firebase Storage.

Firebase Storage містить фотографії профілю кожного користувача. У папці “images” даного сховища зберігаються фотографії, назва яких складається із унікального ідентифікатора власника(uid) та розширення файлу зображення.

Cloud Firestore зберігає колекцію документів із повним набором інформації про кожного користувача. Одним із полів такого документу є “img”, значення якого відповідає посиланню на відповідне зображення в сховищі Firebase Storage.

Схематичне зображення сутностей сховищ та зв’язків між ними представлене на рисунку 1 у додатку Б.

Колекція “users” зберігає інформацію про кожного зареєстрованого в системі користувача. Дана колекція містить документи, кожен з яких описує

певного користувача та має назву, що відповідає унікальному ідентифікатору особи(uid). Кожен документ в свою чергу має наступні поля:

- string uid - унікальний ідентифікатор користувача;
- string bday - дата народження користувача;
- string bio - короткий опис користувача;
- string city - місце проживання користувача;
- array dislikes – масив об’єктів - список інших користувачів, які “не сподобались” даному;
- string edu - інформація про освіту користувача;
- string email - електронна адреса користувача;
- string gender - стать користувача;
- string identity - сексуальна орієнтація користувача;
- string img – посилання на фотографію профілю користувача, що зберігається в Firebase Cloud Storage. Зберігається в форматі string;
- array inters – масив об’єктів - список інтересів користувача;
- array likes - масив об’єктів - список інших користувачів, які “сподобались” даному;
- array matches - масив об’єктів - список інших користувачів, у яких сталася взаємна симпатія із даним;
- string name - ім’я користувача. Зберігається в форматі string;
- string prof - професія користувача. Зберігається в форматі string;
- string showMe - інформація про стать користувачів, яким можна показувати даного користувача. Зберігається в форматі string;
- string showToMe - інформація про стать користувачів, яких хоче бачити даний користувач. Зберігається в форматі string;

Колекція “matches” зберігає інформацію про створену пару кожного користувача. Дана колекція містить документи, кожен з яких описує кожну

створену пару та має назву, що відповідає унікальному ідентифікатору пари(uid). Кожен документ в свою чергу має наступні поля:

- string id - унікальний ідентифікатор пари;
- number createdAt - інформація про дату створення пари;
- map resentMessage - мапа, що містить пари ключ-значення, що відповідають останньому повідомленню створеної пари в месенджері;
    - number sentAt - інформація про дату відправлення повідомлення;
    - string sentBy - інформація про унікальний ідентифікатор відправника повідомлення;
    - string text - інформація про текст повідомлення;
- array usersData – маств об’єктів - множина користувачів, що складається із 2 елементів - учасників створеної пари;
    - map 0 - мапа, що містить інформацію про одного із учасників створеної пари;
        - string id - інформація про унікальний ідентифікатор користувача;
        - string img - посилання на файл із зображенням профілю користувача в Firebase Cloud Storage;
        - string name - інформація про ім’я користувача
    - map 1 - мапа, що містить інформацію про іншого учасникіа створеної пари;
        - string id - інформація про унікальний ідентифікатор користувача;
        - string img - посилання на файл із зображенням профілю користувача в Firebase Cloud Storage;
        - string name - інформація про ім’я користувача

- array usersId - множина унікальний ідентифікаторів учасників створеної пари;
    - string 0 - інформація про унікальний ідентифікатор першого користувача;
    - string 1 - інформація про унікальний ідентифікатор другого користувача;

Колекція “dialogs” зберігає документи, назва яких є унікальним ідентифікатором документу і співпадає із унікальним ідентифікатором пари (документу колекції matches). Даний документ зберігає в собі колекцію “messages”, яка в свою чергу містить документи, що відповідають кожному повідомленню. Структура документів “messages”:

- number sentAt - дата відправлення повідомлення;
- string sentBy - id відправника;
- string text - текст повідомлення;

Робота із базами даних здійснюється асинхронно, за допомогою методів взаємодії зі сховищами Firebase. Приклади таких методів можна переглянути на рисунках 1-5 у додатку В.

3.7 Тестування програми і результати її виконання.

Перед початком тестування програми я провела автоматичну генерацію фейкових користувачів в базу даних. Для цього я створила декілька масивів із набором інформації по даним користувача: чоловічі, жіночі імена та зображення, міста проживання, інтереси, гендери, дати народження тощо. Також логіни формуються у вигляді рядка “male1@gmail.com” або “female2@gmail.com” в залежності від статі та порядкового номеру

користувача. Пароль у кожного фейкового користувача “123456”. Після цього написала декілька функцій перемішування рандомним чином цих даних, створення масиву користувачів та врешті-решт додавання їх в базу даних. Приклади коду даної реалізації представлені на рисунках 6-9 у додатку В.

Коли користувач вперше потрапляє на веб-сайт, перед ним постає сторінка “Авторизація”:

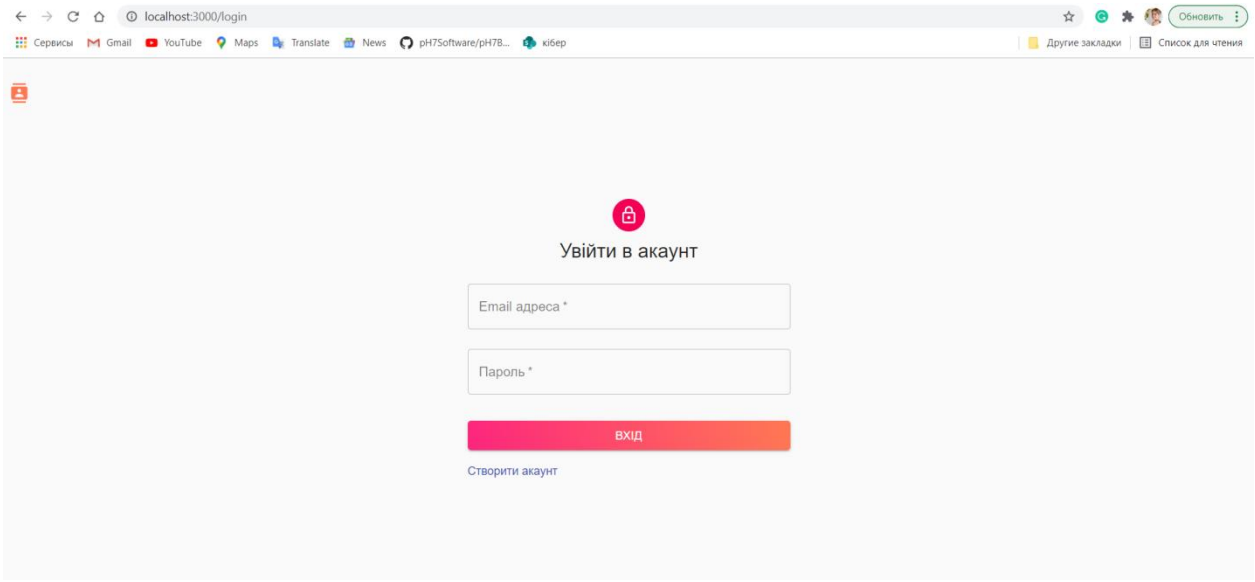

Рисунок 3.7.1. Сторінка “Авторизація”.

При натисканні на посилання “Створити акаунт”, користувач потрапляє на сторінку “Реєстрація”:

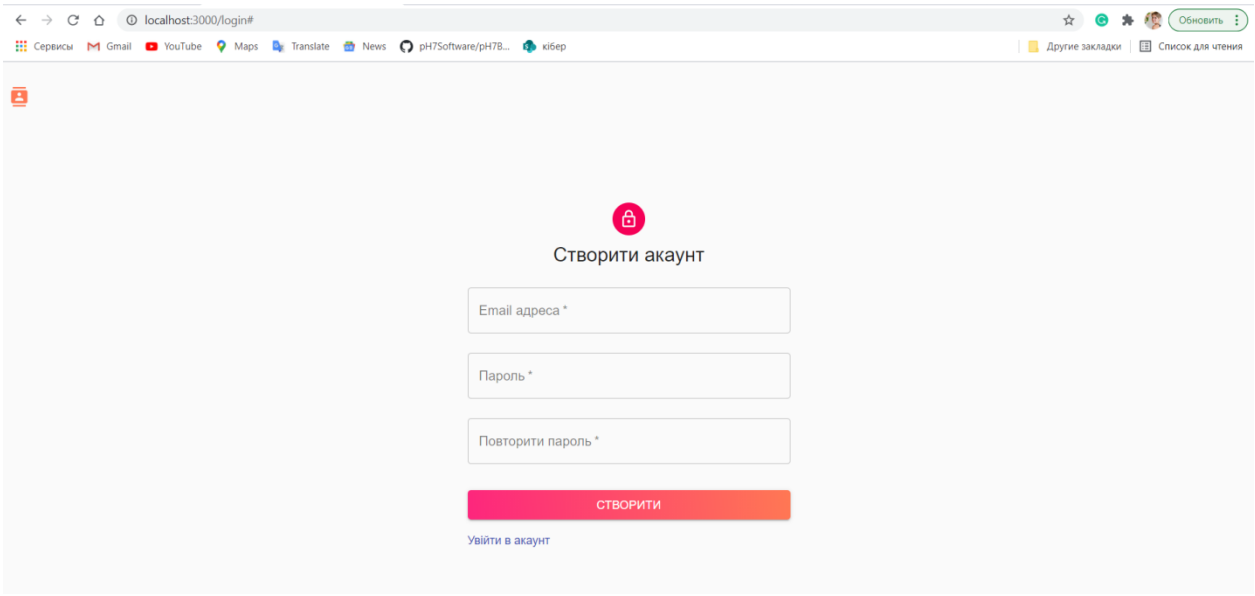

Рисунок 3.7.2. Сторінка "Реєстрація".

При заповненні полів "Пароль" та "Повторити пароль" на сторінці "Реєстрація" різними даними, введення невалідної електронної адреси або такої, що вже існує в системі, програма сповіщає користувача про це:

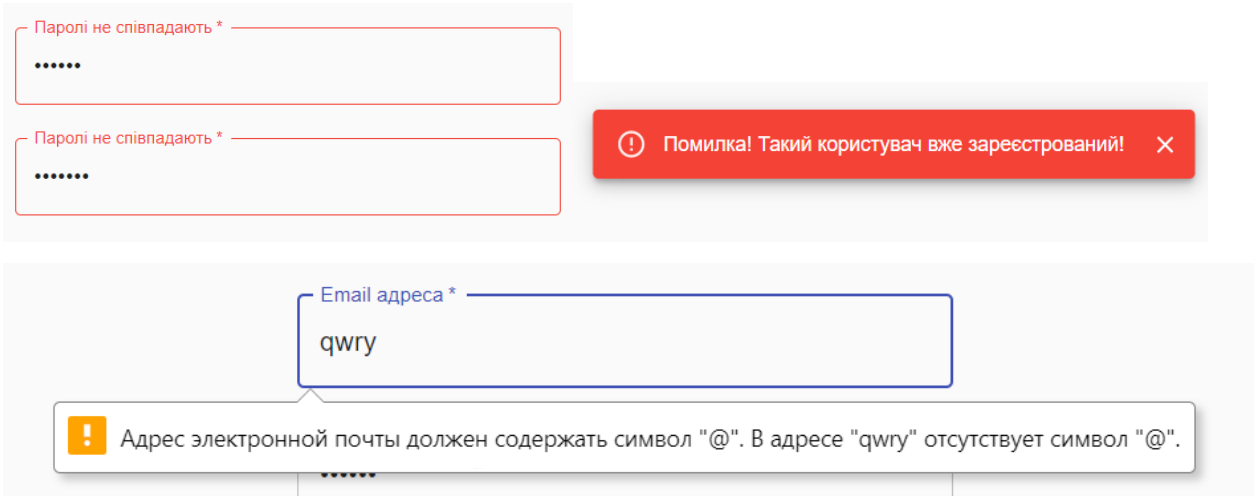

Рисунок 3.7.3. Сповіщення про помилки на сторінці "Реєстрація".

На сторінці “Авторизація” при введенні електронної адреси, що не зареєстрована в системі або неправильного паролю, програма сповіщає користувача про це:

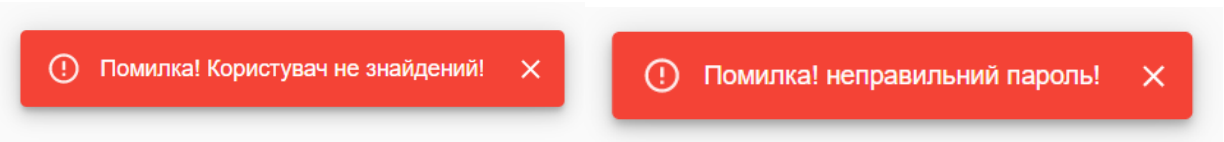

Рисунок 3.7.4. Сповіщення про помилки на сторінці “Авторизація”.

Після натискання на кнопку “Створити” на сторінці “Реєстрація”, користувач потрапляє на сторінку “Анкета користувача”, де повинен заповнити ряд обов’язкових полів (обов’язкові поля позначені знаком “*”).

На сторінці “Анкета користувача” присутні сповіщення для користувача про успішне завантаження фото, перевірка на повноліття та незаповненість обов’язкових полів :

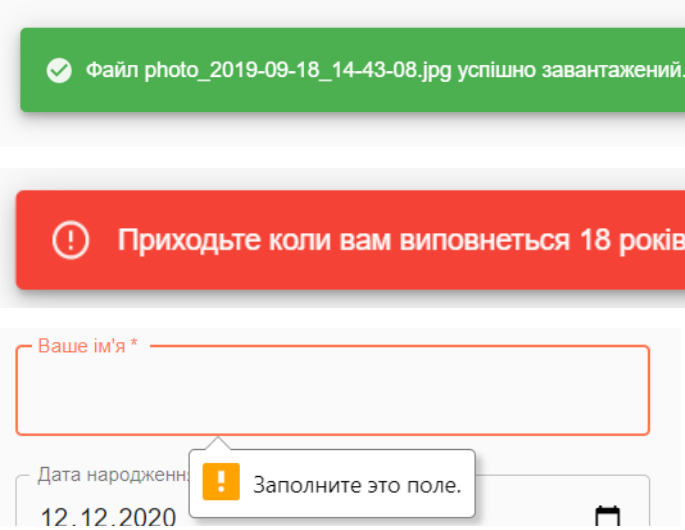

Рисунок 3.7.5. Сповіщення на сторінці “Анкета користувача”.

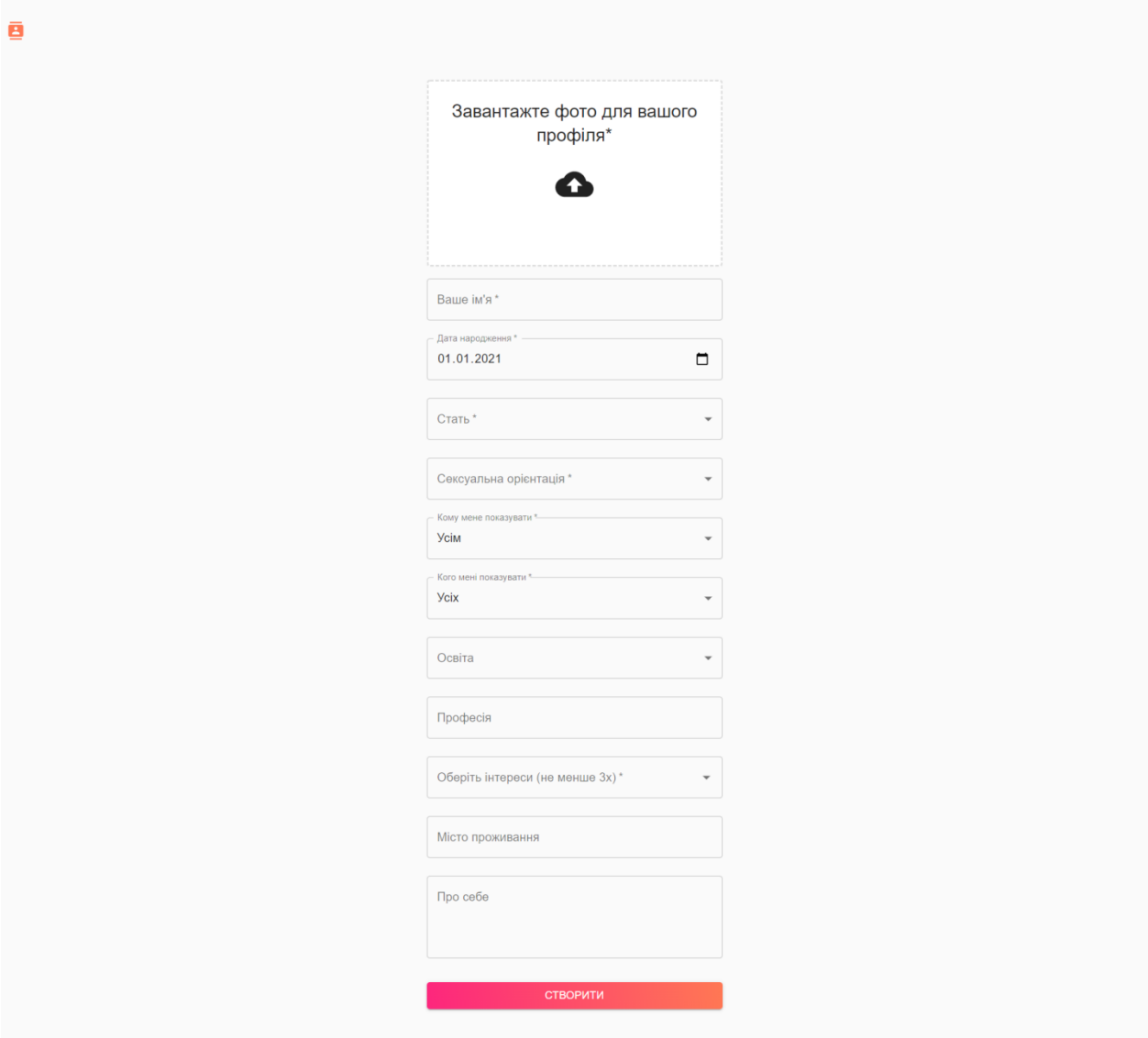

Рисунок 3.7.6. Сторінка "Анкета користувача".

Після створення профілю, користувач потрапляє на сторінку "Головна", де присутні свайп-картки та drawer-меню:

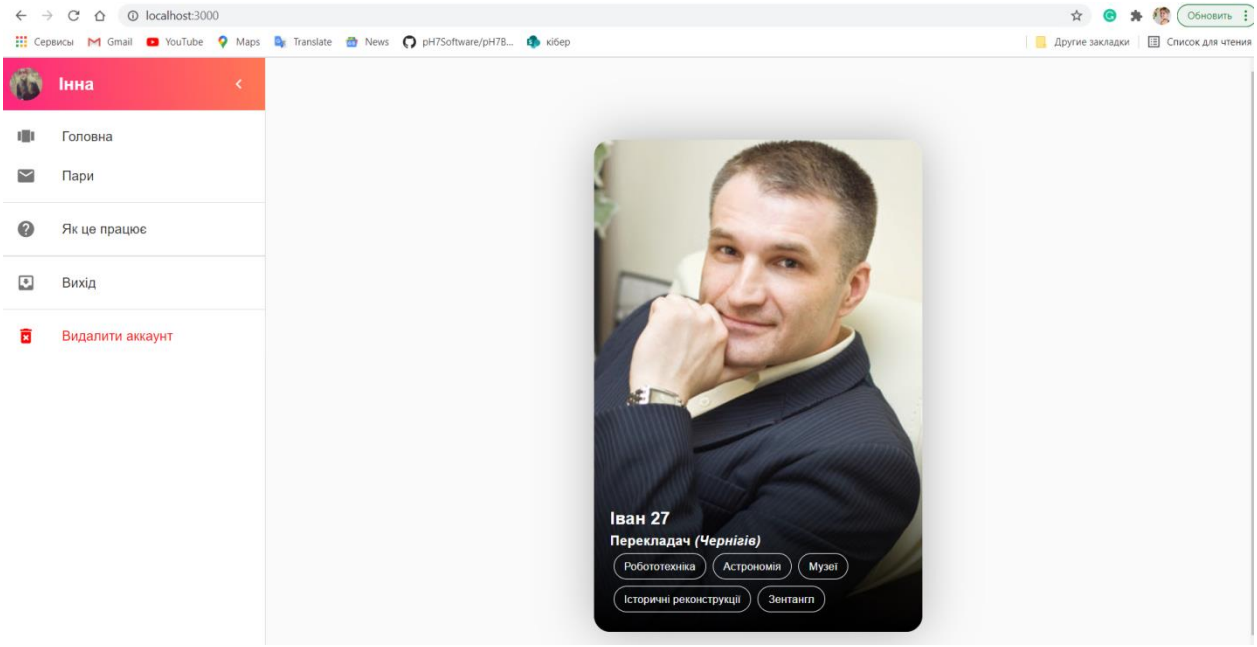

Рисунок 3.7.7.Сторінка "Головна".

При натисканні на пункт меню "Вихід", відбувається вихід з акаунту і перенаправлення на сторінку "Авторизація".

Свайп-картки можна переміщати праворуч, що означає, що людина вам симпатизує та ліворуч, щоб перейти до перегляду наступної пари:

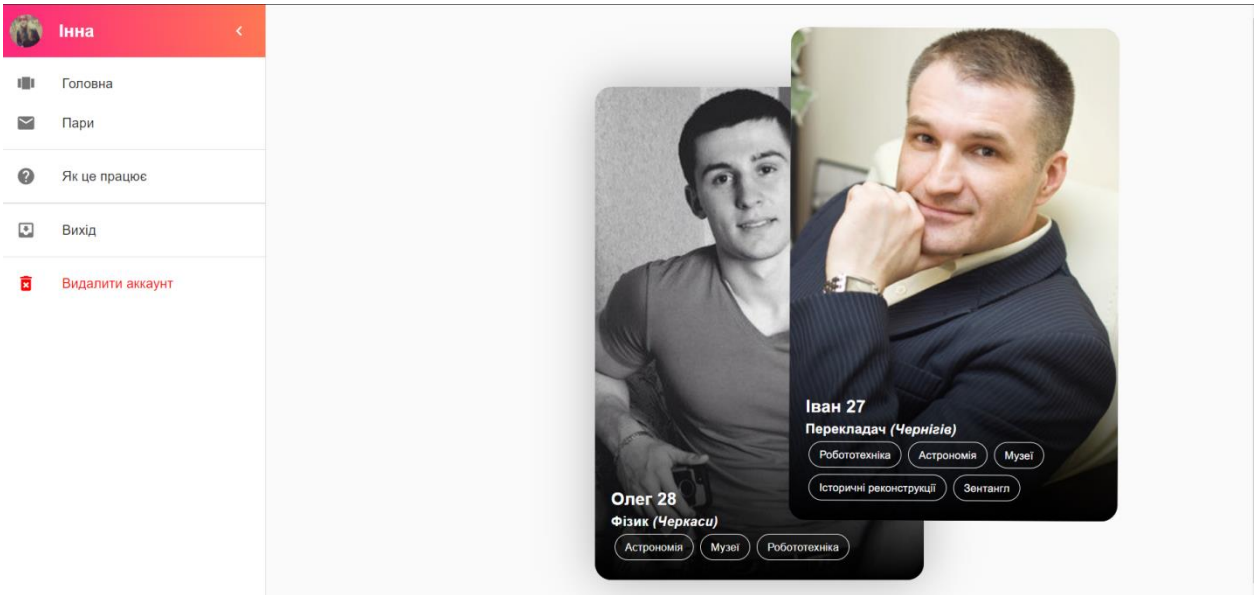

Рисунок 3.7.8. Переміщення свайп-картки праворуч.

Нехай користувачу Інні сподобався користувач Іван, як зображено на Рисунку 3.7.8, та навпаки. В такому випадку відбувається створення пари, і користувач отримує сповіщення про це:

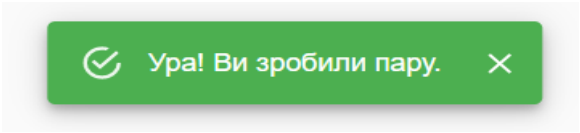

Рисунок 3.7.9. Сповіщення про створення пари.

Також користувачі з’являються один у одного на сторінці “Пари”:

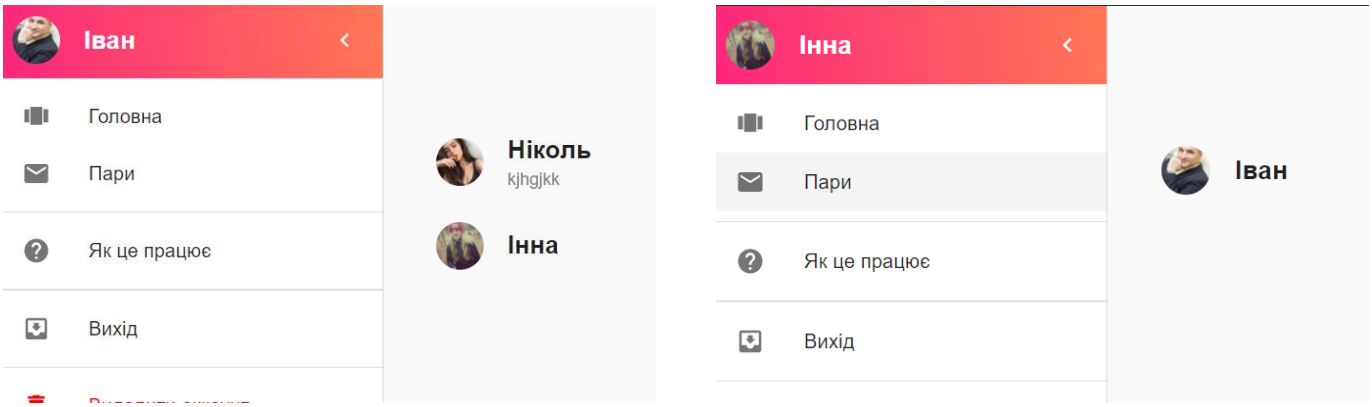

Рисунок 3.7.10. Відображення створеної пари.

Натиснувши на іконку своєї пари, користувач потрапляє на сторінку діалогу з партнером. До початку спілкування на цій сторінці відображається інформація про дату створення пари:

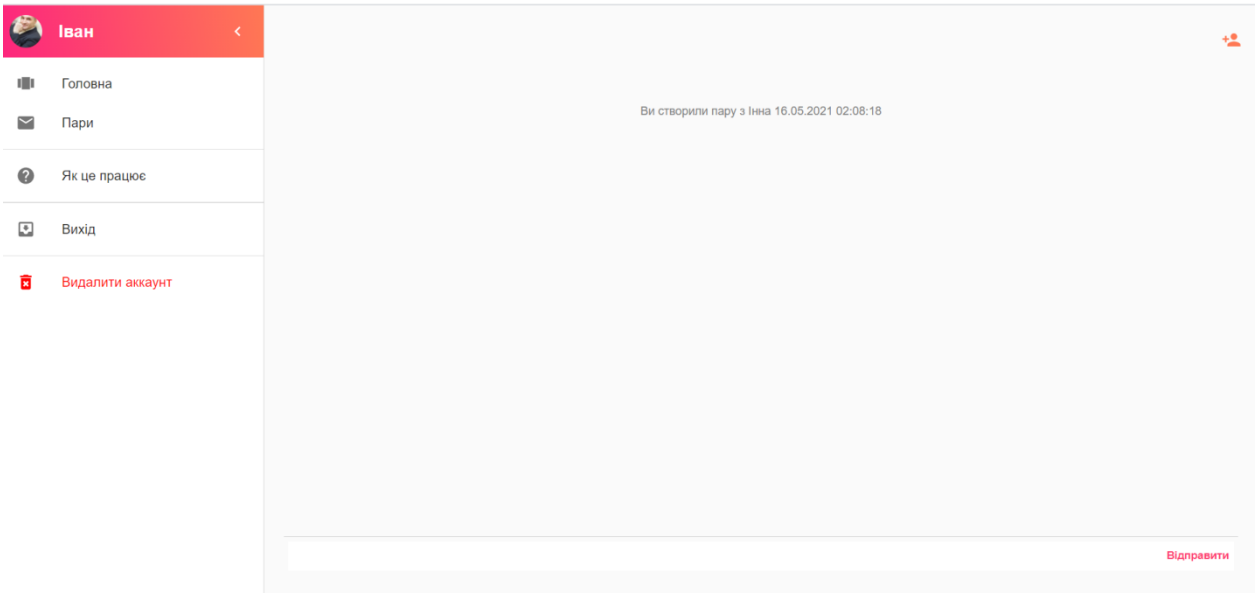

Рисунок 3.7.11. Діалог з новоствореною парою.

Відправлення повідомлення відбуваються в режимі реального часу, тому обидва користувача можуть спілкуватись у месенджері:

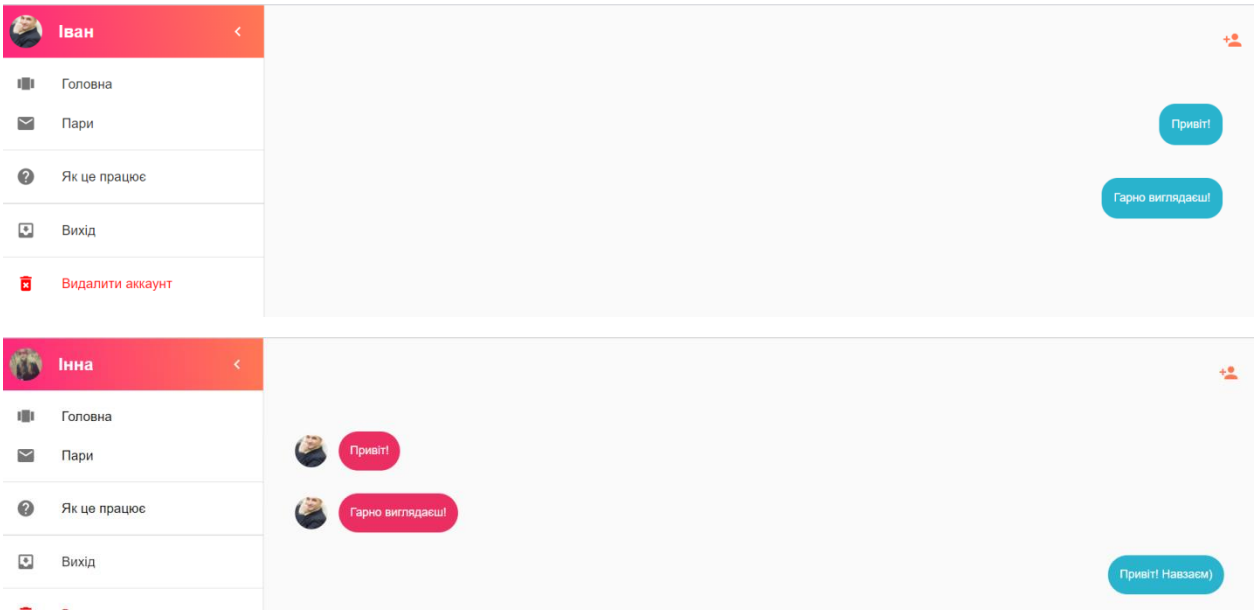

Рисунок 3.7.12. Діалог з парою.

До того ж, натиснувши на кнопку перегляду профілю пари у правому верхньому кутку чату, користувачу відкривається детальна інформація про свою пару (при цьому вікно з drawer-меню закривається):

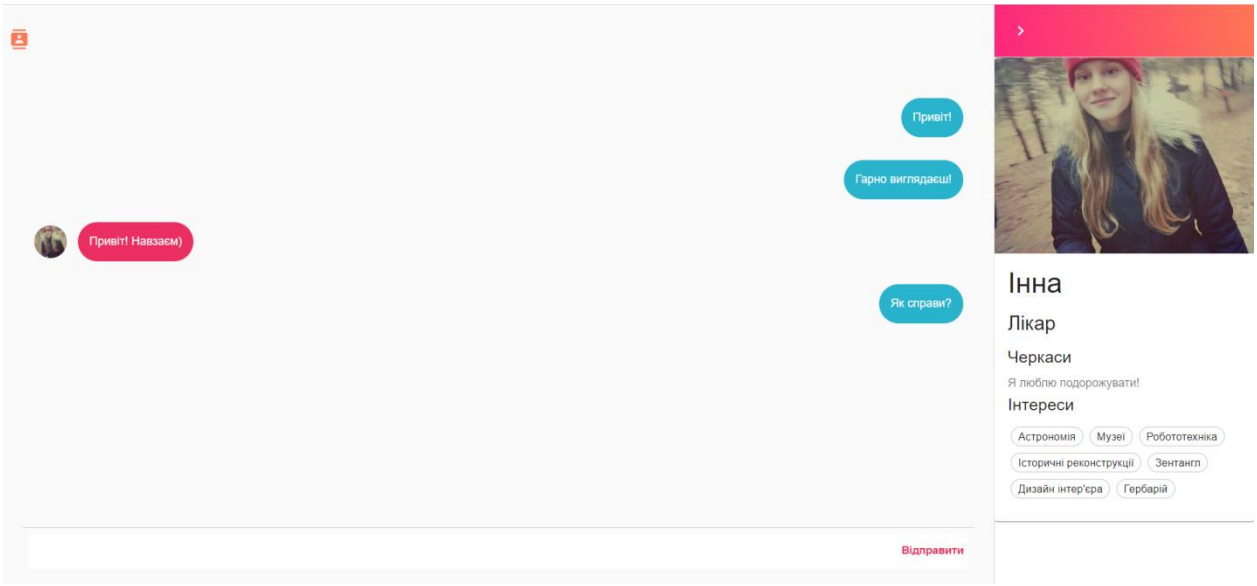

Рисунок 3.7.13. Перегляд детальної інформації про пару.

Також користувач може обрати пункт меню “Як це працює”, та потрапити на відповідну сторінку:

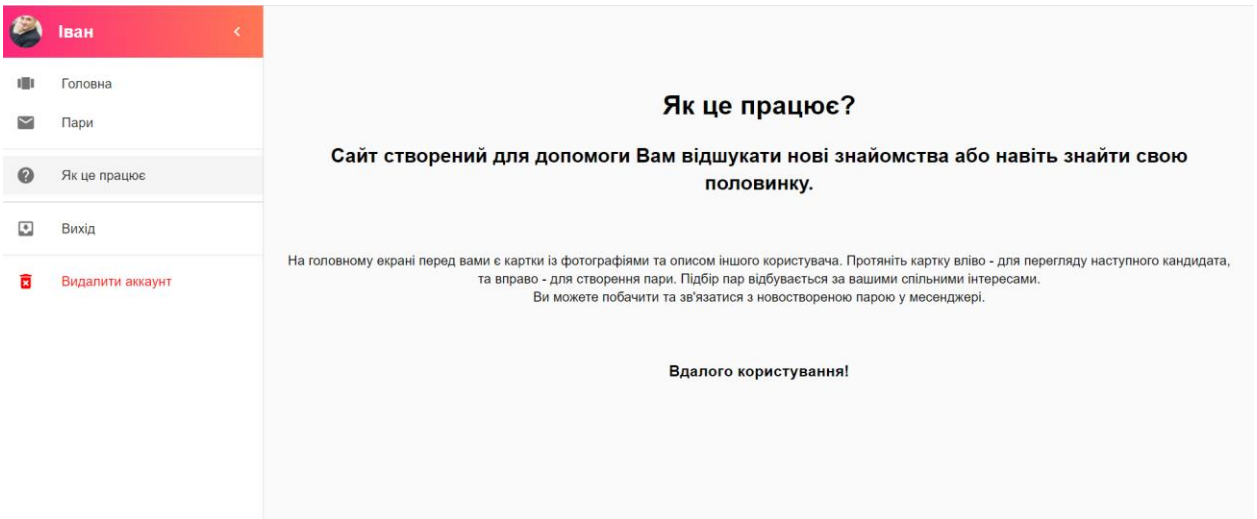

Рисунок 3.7.14. Перегляд принципів роботи сайту.

У випадку, якщо користувач бажає видалити свій акаунт, він натискає на кнопку “Видалити акаунт”, що відкриває йому форму для введення логіну та паролю. Після введення коректних даних та натиснення на кнопку “Видалити” відбувається видалення акаунту та перенаправлення на сторінку “Авторизація”:

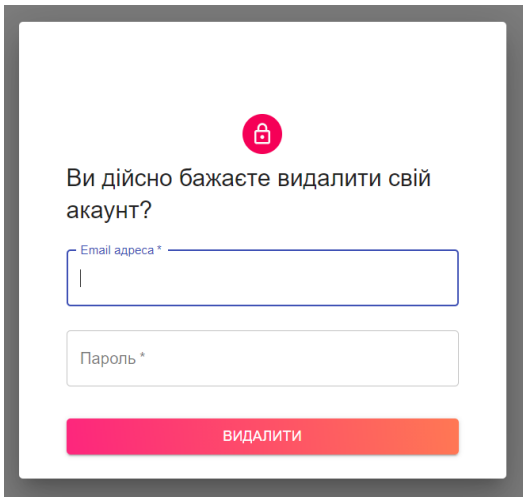

Рисунок 3.7.15. Видалення акаунту.

Варто зазначити, що веб-сайт є адаптивним для будь-яких розмірів екрану, а тому може бути використаним на будь-яких девайсах:

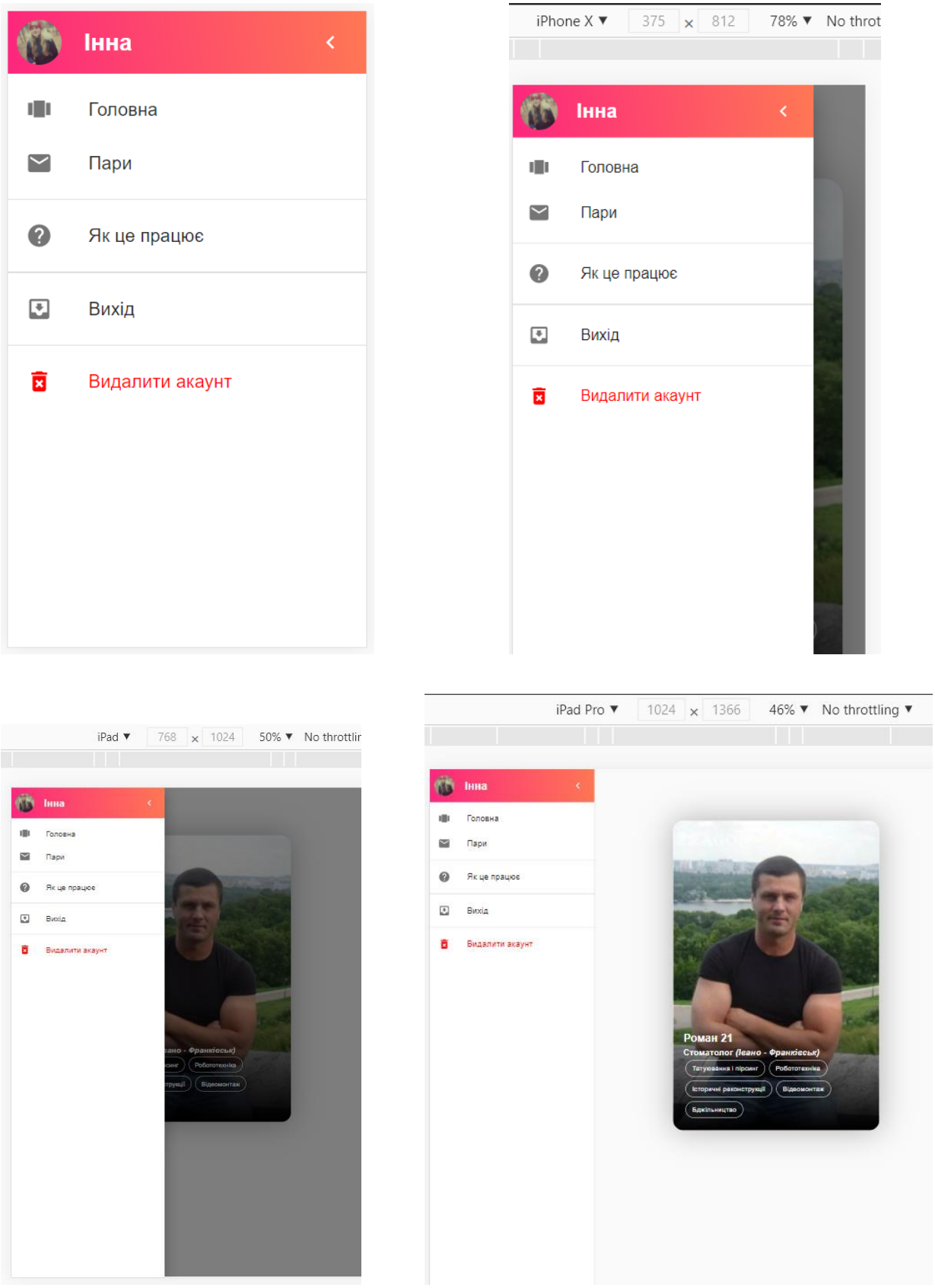

Рисунок 3.7.16. Адаптивність сайту на різні розширення екрану.

ВИСНОВОК:

В процесі розробки курсової роботи мною було проаналізовано процес створення застосувань, призначених для організації знайомств в мережі.

В ході роботи над курсовою було проаналізовано популярні та актуальні технології розробки веб-сайтів. В результаті було обрано саме такі, що найбільш зручним та доступним способом реалізують багатосторінковий веб-сайт.

Також проведено аналіз існуючих аналогів розробки. Визначено спільні риси всіх сервісів даної тематики та використано у проекті.

Найкращим способом реалізації подібної системи визначено розробку веб-сайту. Тож у результаті розроблено веб-сайт знайомств, де люди можуть знайти співрозмовників, друзів та свою половинку, в якому реалізовано підбір пар за інтересами і місцезнаходженням, можливість зв’язатися зі своєю парою та познайомитися поблище. Застосунок має зрозумілий інтерфейс, яким легко можуть користуватись навіть не обізнані у роботі веб-сайтів користувачі.

У проекті є ще багато функціоналу, який можна додати задля розвитку та вдосконалення платформи. По-перше, можна додати можливість бачити користувачів, яким ти сподобався до того моменту, поки вони не сподобаються тобі, що додасть людям більше інтересу в користуванні веб-сайтом, а власникам додаткову монетизацію за рахунок купівлі цих функцій. По-друге, додати “популярність” профілю, що позначатиме відображення кількості людей, які поставили “лайк” даному користувачу. По-третє, можна додати більше критеріїв для підбору запропонованих пар самою програмою, не тільки за предметом спільних інтересів, а й, наприклад, місцем роботи, освітою тощо. Тож проекту є куди рухатись та розвиватись, додавати нові

функції та покращувати його роботу, тому можна впевнено сказати, що це не кінець розвитку проекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Аналог розробки: Love.ua. [Електронний ресурс]. https://love.ua/
2. Аналог розробки: Tinder. [Електронний ресурс]. https://tinder.com/uk
3. Аналог розробки: Emilydates. [Електронний ресурс]. https://emilydates.com/
4. Аналог розробки: OneAmour. [Електронний ресурс]. https://oneamour.com/
5. Види веб-сайтів. [Електронний ресурс]. https://99designs.com/blog/web-digital/types-of-websites/
6. Види веб-сайтів. [Електронний ресурс]. https://www.gwsmedia.com/20-different-types-websites-1
7. База даних. [Електронний ресурс]. https://www.oracle.com/database/what-is-database/
8. NoSQL databases. [Електронний ресурс]. https://www.mongodb.com/nosql-explained
9. Види NoSQL-СКБД. [Електронний ресурс]. https://www.quality-assurance-group.com/nosql-perevagy-ta-nedoliky-nerelyatsijnyh-baz-danyh/
10. Переваги Firestore. [Електронний ресурс]. https://firebase.google.com/docs/firestore
11. Single page application. Електронний ресурс]. .https://wezom.com.ua/blog/chto-takoe-spa-prilozheniya
12. React.js. [Електронний ресурс]. https://ru.reactjs.org/
13. React.js. [Електронний ресурс]. https://metanit.com/web/react/1.1.php
14. Material - UI. [Електронний ресурс]. https://material-ui.com/ru/
15. Ajax. [Електронний ресурс]. https://habr.com/ru/post/14246/

ДОДАТКИ

Додаток А: блок-схема карти програмної частини курсової роботи

Рисунок 1. Блок-схема структури програмної частини курсової роботи.

Додаток Б: схематичне зображення сутностей та зв’язків між ними у використовуваних сховищах.

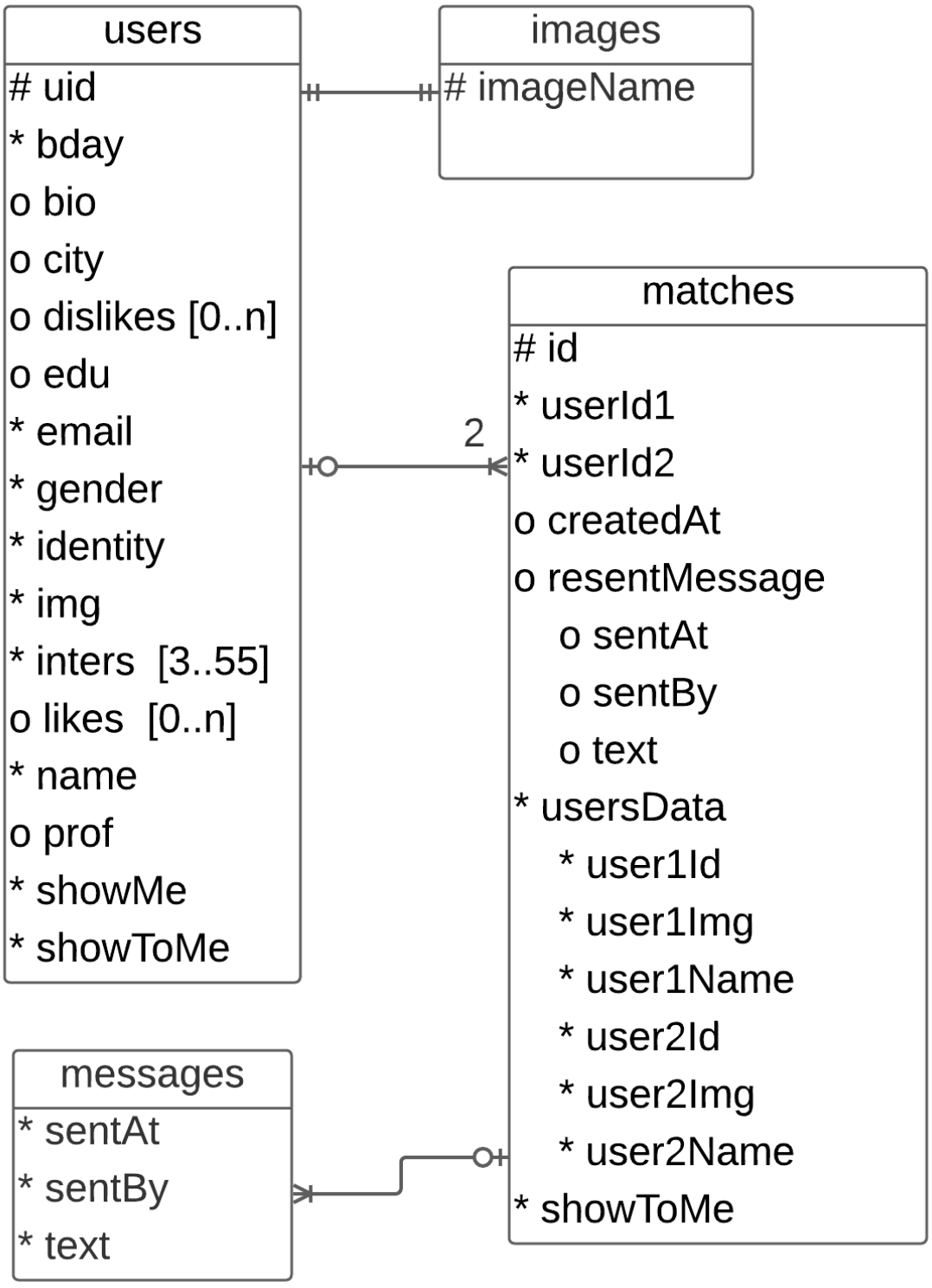

Рисунок 1. Схематичне зображення сутностей та зв’язків між ними

Додаток В: зображення вихідного коду деяких частин реалізації.

```
const saveUser = (user, mod) => {
    db.collection("users").doc(user.uid).set(user, { merge: true })
        .then(() => {
            setUserData(user)
            // !mod && history.push("/")
            console.log("Document successfully written!");
        })
        .catch((error) => {
            setUserData(null)
            console.error("Error writing document: ", error);
        });
};
const signin = (email, password) => {
    return firebase
        .auth()
        .signInWithEmailAndPassword(email, password)
        .then((response) => {
            setUser(response.user);
            setCredential(response)
            return response.user;
        });
};
const signup = (email, password) => {
    return firebase
        .auth()
        .createUserWithEmailAndPassword(email, password)
        .then((response) => {
            setUser(response.user);
            setCredential(response)
            return response.user;
        });
};
```

```
const signout = () => {
    return firebase
        .auth()
        .signOut()
        .then(() => {
            setUser(false);
        });
};
const deleteUser = (email, password) => {
    const credential = firebase.auth.EmailAuthProvider.credential(
        email,
        password
    );
    user.reauthenticateWithCredential(credential).then(function () {
        db.collection("users").doc(user.uid).delete().then(() => {
            user.delete().then(function () {
                // User deleted.
            }).catch(function (error) {
                // An error happened.
            });
        }).catch((error) => {
          console.error("Error removing document: ", error);
        });

    }).catch(function (error) {
        // An error happened.
    });
};
```

Рисунок 1. Методи взаємодії з базами даних. Робота з акаунтом користувача в базі даних Cloud Firestore.

```
const makeMatch = (matchUser) => {
  console.log('MATCH call!');
  setOpenAlert(true)
  const matchId = `${user.uid}${matchUser.uid}`.split('').sort(function () { return 0.5 - Math.random() }).join('').slice(0, 28);
  const match = {
    createAt: Date.now(),
    usersId: [user.uid, matchUser.uid],
    usersData: [{
      img: user.img,
      id: user.uid,
      name: user.name,
    }, {
      img: matchUser.img,
      id: matchUser.uid,
      name: matchUser.name,
    }],
    recentMessage: {

    },
    id: matchId
  }
  db.collection("matches").doc(matchId).set(match, { merge: true })
    .then(() => {
      console.log("Document successfully written! MATCH!!");

    })
    .catch((error) => {
      console.error("Error writing document: ", error);
    });
}
```

Рисунок 2. Методи взаємодії з базами даних. Додавання створеної пари в базу даних Cloud Firestore.

```
const { data, status, error } = useFirestoreQuery(
  db.collection('dialogs')
    .doc(matchId)
    .collection('messages')
    .orderBy('sentAt')
);
```

Рисунок 3. Методи взаємодії з базами даних. Отримання всіх повідомлень певної пари із бази даних Cloud Firestore.

```
const handleSend = (e) => {
  e.preventDefault();
  if (!input.trim()) {
    return
  }
  const message = {
    text: input.trim(),
    sentAt: Date.now(),
    sentBy: auth.user.uid
  }
  setMessages([...messages, message]);
  setInput("");
  db.collection('dialogs')
    .doc(matchId)
    .collection('messages')
    .add(message)
    .then(function (docRef) {
      db.collection("matches").doc(matchId).set({ recentMessage: message }, { merge: true })
        .then(() => {
          console.log("Document successfully written! RECENT!!");
        })
        .catch((error) => {
          console.error("Error writing document: ", error);
        });
    })
    .catch(function (error) {
      console.error("Error writing document: ", error);
    })
};
```

Рисунок 4. Методи взаємодії з базами даних. Відправка повідомлень в базу даних Cloud Firestore.

```
const ext = selectedPhoto.path.slice(selectedPhoto.path.lastIndexOf('.'))
const ref = storage.ref().child(`images/${auth.user.uid}${ext}`)
ref.put(selectedPhoto).then((snapshot) => {
    snapshot.ref.getDownloadURL().then(url => {
        const newUser = {
            "prof": selectedProf,
            "gender": selectedGender,
            "showMe": selectedShowMe,
            "identity": selectedIdentity,
            "uid": auth.user.uid,
            "dislikes": [],
            "name": selectedName,
            "likes": [],
            "img": url,
            "showToMe": selectedShowToMe,
            "inters": selectedInters,
            "bday": selectedDate,
            "edu": selectedEdu,
            "city": selectedCity,
            "bio": selectedBio,
            "matches": [],
            "created": Date.now()
        }
        auth.saveUser(newUser)
        setTimeout(() => {
            history.push('/')
        }, 1000);
    })
});
```

Рисунок 5. Методи взаємодії з базами даних. Формування імені файлу та відправка зображення у сховище Firebase Cloud Storage.

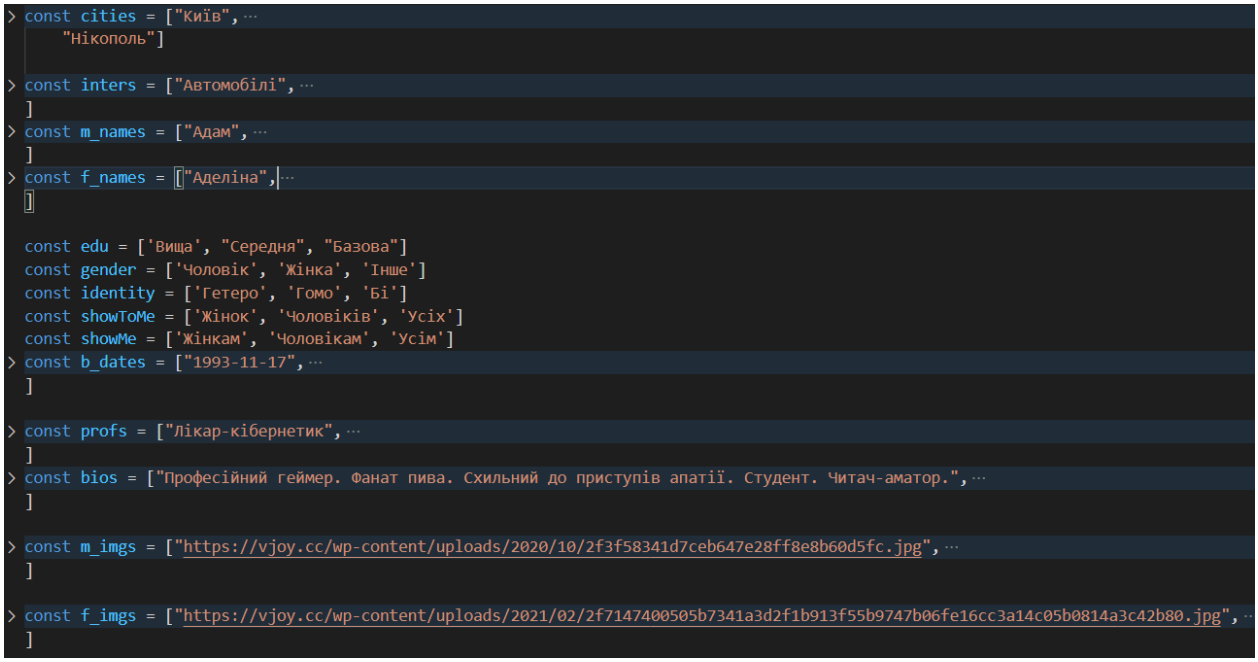

Рисунок 6. Наповнення масивів даними.

```
function generateInts() {
    const ints = []
    let i = 0
    while (i < rand(5, 15)) {
        const int = getInt()
        if (!ints.includes(int)) {
            ints.push(int)
            i++
        }
    }
    function getInt() {
        return inters[rand(0, inters.length - 1)]
    }
    return ints
}
```

Рисунок 7. Функція генерації інтересів.

```
function generateFakeUsers(gender) {
    let imgArray = []
    let names = []
    if (gender === 'Чоловік') {
        imgArray = m_imgs
        names = m_names
    } else if (gender === 'Жінка') {
        imgArray = f_imgs
        names = f_names
    }
    const users = imgArray.map((img, i) => {
        const user = {}
        user.uid = uuid.v4()
        user.email = `${gender === 'Чоловік' ? 'male' : gender === 'Жінка' ? 'female' : 'gender'}${i+1}@gmail.com`
        user.password = '123456'
        user.img = img
        user.gender = gender
        user.identity = identity[generateIdentity()]
        user.name = names[rand(0, names.length - 1)]
        user.bday = b_dates[rand(0, b_dates.length - 1)]
        user.edu = edu[rand(0, edu.length - 1)]
        user.city = cities[rand(0, cities.length - 1)]
        user.prof = profs[rand(0, profs.length - 1)]
        user.bio = bios[rand(0, bios.length - 1)]
        user.inters = generateInts()
        const displayPos = generateShowMe(gender, user.identity)
        user.showMe = showMe[displayPos]
        user.showToMe = showToMe[displayPos]
        user.matches = []
        user.likes = []
        user.dislikes = []
        return user
    })
    return users
}
```

Рисунок 8. Функція генерації даних та створення масиву користувачів.

```
function generateShowMe(gender, identity) {
    let pos = null
    let chance = rand(0,30)
    if (gender === 'Чоловік') {
        if (identity == 'Гетеро') {
            pos = chance > 25 ? 2 : 0
        } else if (identity == 'Гомо') {
            pos = chance > 25 ? 1 : 2
        } else {
            pos = chance > 10 ? 0 : chance > 20 ? 1 : 2
        }
    } else if (gender === 'Жінка') {
        if (identity == 'Гетеро') {
            pos = chance > 25 ? 2 : 1
        } else if (identity == 'Гомо') {
            pos = chance > 25 ? 0 : 2
        } else {
            pos = chance > 10 ? 0 : chance > 20 ? 1 : 2
        }
    } else if (gender === 'Інше'){
        pos = 2
    }
    return pos
}
```

Рисунок 9. Функція генерації даних "Кого мені показувати".

```
import firebase from 'firebase/app'
import "firebase/auth";
import "firebase/firestore";
import "firebase/storage";

firebase.initializeApp({
    apiKey: "AIzaSyA_rt6DDS-md6SD-XTzvV9ASHsB6ZeCKZo",
    authDomain: "tinder-34f9d.firebaseapp.com",
    projectId: "tinder-34f9d",
    storageBucket: "tinder-34f9d.appspot.com",
    messagingSenderId: "178854142403",
    appId: "1:178854142403:web:d9b741f3a00900a0419d76"
});
const db = firebase.firestore();
const storage = firebase.storage();
export { firebase, db, storage }
```

Рисунок 10. Використання конфігурацій сховищ у програмі.

```
useEffect(() => {
  console.log(auth);
  if (!auth.user && !loginPageMatch?.isExact) {
    history.push("/login");
  } else if (auth.user && !auth.userData && !registerPageMatch?.isExact) {
    history.push("/register");
  } else if (auth.user && auth.userData && (loginPageMatch?.isExact || registerPageMatch?.isExact)) {
    history.push("/");
  }
}, [auth, location]);
```

Рисунок 11. Деяка логіка маршрутизації.

```
<Switch>
  <Route path="/chat/:matchId">
    {
      auth.user &&
      <>
        {}
        <ChatScreen />
      </>
    }
  </Route>
  <Route path="/chat">
    {
      auth.user &&
      <>
        {}
        <Chats />
      </>
    }
  </Route>
  <Route path="/login">
    {
      !auth.user &&
      <>
        {}
        <SignForm />
      </>
    }
  </Route>
  <Route path="/register">
    {
      auth.user &&
      <>
        {}
        <RegisterForm />
      </>
    }
  </Route>
  <Route exact path="/howitworks">
    {
      auth.user &&
      <>
        {}
        <HowItWorks />
      </>
    }
  </Route>
  <Route exact path="/">
    {
      auth.user &&
      <>
        {}
        <Cards />
      </>
    }
  </Route>
</Switch>
```

Рисунок 12.Задання маршрутизації.

```
{auth.user && <><List>
  <Link to="/">
    <ListItem button>
      <ListItemIcon>{<ViewCarouselIcon />}</ListItemIcon>
      <ListItemText primary={'Головна'} />
    </ListItem></Link>
  <Link to="/chat">
    <ListItem button>
      <ListItemIcon>{<MailIcon />}</ListItemIcon>
      <ListItemText primary={'Пари'} />
    </ListItem></Link>
</List>
  <Divider />
  <List>
  <Link to="/howitworks">
    <ListItem button>
      <ListItemIcon><HelpIcon /></ListItemIcon>
      <ListItemText primary={'Як це працює'} />
    </ListItem>
    </Link>
  </List>
  <Divider />
  <List>
    <ListItem button onClick={e => auth.signout()}>
      <ListItemIcon><InboxIcon /></ListItemIcon>
      <ListItemText primary={'Вихід'} />
    </ListItem>
  </List>
  <Divider />
  <List>
    <ListItem button onClick={e => setOpenDeleteDialog(true)} >
      <ListItemIcon><DeleteForeverIcon style={{ color: 'red' }} /></ListItemIcon>
      <ListItemText primary={'Видалити акаунт'} style={{ color: 'red' }} />
    </ListItem>
  </List></>}
```

Рисунок 13. Обробка кожного пункту drawer-меню

```
{MatchUser && <Card className={classes.card}>
  <CardMedia
    className={classes.media}
    image={MatchUser?.data?.img}
    title={MatchUser?.data?.name}
  />
  <CardContent>
    <Typography gutterBottom variant="h4" component="h2">
    {MatchUser?.data?.name}
    </Typography>
    <Typography gutterBottom variant="h5" component="h3">
    {MatchUser?.data?.prof}
    </Typography>
    <Typography gutterBottom variant="h6" component="h4">
    {MatchUser?.data?.city}
    </Typography>
    <Typography variant="body2" color="textSecondary" component="p">
    {MatchUser?.data?.bio}
    </Typography>
    <Typography gutterBottom variant="h6" component="h4" >
      Інтереси
    </Typography>
    <Typography variant="body2" color="textSecondary" component="div" className={classes.chips}>
    {MatchUser?.data?.inters.map(inter => <Chip variant="outlined" key={inter} size="small" label={inter} />)}
    </Typography>
  </CardContent>
</Card>}
```

Рисунок 14. Відображення на екрані профайлу партнера.

```
function filterUsers(me, users) {
  console.log('filters', me, users);
  let filteredUsers = []
  let showToMe = ''
  if (me.showToMe === "Жінок") {
    showToMe = 'Жінка'
  } else if (me.showToMe === "Чоловіків") {
    showToMe = 'Чоловік'
  }
  filteredUsers = users.filter(user => {
    let intersMatches = 0
    me.inters.forEach(inter => {
      user.inters.includes(inter) && intersMatches++
    })
    let showToMeMatch = !showToMe ? true : user.gender === showToMe ? true : false
    let myGender = ''
    if (user.showMe === "Жінкам") {
      myGender = 'Жінка'
    } else if (user.showMe === "Чоловікам") {
      myGender = 'Чоловік'
    }
    let showMeMatch = !myGender ? true : me.gender === myGender ? true : false
    let notViewed = !me.dislikes.includes(user.uid) && !me.likes.includes(user.uid)
    return showMeMatch && showToMeMatch && intersMatches >= 3 && notViewed && me.uid !== user.uid
  })
```

Рисунок 15. Фільтрація запропонованих кристувачів.

```
ReactDOM.render(
  <React.StrictMode>
    <ProvideAuth>
      <BrowserRouter>
        <MatchUserProvider>
          <App />
        </MatchUserProvider>
      </BrowserRouter>
    </ProvideAuth>
  </React.StrictMode>,
  document.getElementById('root')
);

serviceWorker.unregister();
```

Рисунок 16. Точка входу в програму.

```
const MatchesList = ({ id, userName, msg, avatar, pairId, sentAt }) => {
  const time = sentAt && `${new Date().toLocaleDateString(sentAt)} ${new Date().toLocaleTimeString(sentAt)}`


  return (
    <Link to={`/chat/${id}`}>
      <div className="matchesPage">
        <Avatar src={avatar} className="avatar"/>
        <div className="title">
          <h1>{userName}</h1>
          <h4>{msg}</h4>
        </div>
      </div>
    </Link>
  );
};

export default MatchesList;
```

Рисунок 17. Логіка наповнення сторінки “Пари”.

```
async function formAction(event) {
    event.preventDefault();
    setTrySend(true)
    if (deleteUser) {
        auth.deleteUser(selectedEmail, selectedPassword)
        closeDialog(false)
        return
    }
    if (formType === "signup" && selectedPassword !== selectedPasswordCheck) {
        return
    }
    try {
        if (formType === "signup") {
            await auth.signup(selectedEmail, selectedPassword);
        } else if (formType === "signin") {
            await auth.signin(selectedEmail, selectedPassword);
        }
    } catch (error) {
        console.warn(error);
        let text = ''
        if (error.code === "auth/user-not-found") {
            text = 'Помилка! Користувач не знайдений!'
        } else if (error.code === "auth/email-already-in-use") {
            text = 'Помилка! Такий користувач вже зареєстрований!'
        } else if (error.code === "auth/wrong-password") {
            text = 'Помилка! неправильний пароль!'
        } else {
            text = 'Упс! Невідома помилка!'
        }
        setOpenAlert(true)
        setAlertText(text)
    }
}
```

Рисунок 18. Асинхронне проведення авторизації та реєстрації користувача.

```
useEffect(() => {
  console.log(matchesSM, 'sm');
  matchesSM ? setdrawerType('temporary') : setdrawerType('persistent')
}, [matchesSM])
useEffect(() => {
  console.log(matchesMD, 'md');
}, [matchesMD])
useEffect(() => {
  console.log(MatchUser);
}, [MatchUser])
```

Рисунок 19. Адаптація drawer-меню на різні розширення екрану.

```
useEffect(() => {
  if (match?.usersData) {
    db.collection("users").doc(match.usersData.find(user => user.id !== auth.user.uid).id).get().then((doc) => {
      if (doc.exists) {
        dispatchMatchUser({ type: 'SET', payload: doc.data() })
        console.log("Document data:");
      } else {
        console.log("No such document!");
      }
    }).catch((error) => {
      console.log("Error getting document:", error);
    });
  }
}, [match])
```

Рисунок 20. Хук для відображення інформації про партнера.

```
const outOfFrame = (character, direction) => {
  console.log(character.name + ' left the screen!', direction)
  let modUser = null
  if (direction === 'left') {
    modUser = { ...user, dislikes: [...new Set([...user.dislikes, character.uid])] }
  } else if (direction === 'right') {
    modUser = { ...user, likes: [...new Set([...user.likes, character.uid])] }
    console.log('MATCH likes', character.likes.includes(user.uid));
    character.likes.includes(user.uid) && makeMatch(character)
  }
  console.log(modUser);
  setUser(modUser)
  modUser && auth.saveUser(modUser, true)
}
```

Рисунок 21. Обробка свайпу.

```
async function formAction(event) {
    event.preventDefault();
    setTrySend(true)
    if (!selectedPhoto) {
        setAlertText('Завантажте фото профіля!')
        setOpenAlert(true)
        return
    }
    if (calculate_age(new Date(selectedDate)) < 18) {
        setAlertText('Приходьте коли вам виповнеться 18 років!')
        setOpenAlert(true)
        return
    }
    const ext = selectedPhoto.path.slice(selectedPhoto.path.lastIndexOf('.'))
    const ref = storage.ref().child(`images/${auth.user.uid}${ext}`)
    ref.put(selectedPhoto).then((snapshot) => {
        snapshot.ref.getDownloadURL().then(url => {
            const newUser = {
                "prof": selectedProf,
                "gender": selectedGender,
                "showMe": selectedShowMe,
                "identity": selectedIdentity,
                "uid": auth.user.uid,
                "dislikes": [],
                "name": selectedName,
                "likes": [],
                "img": url,
                "showToMe": selectedShowToMe,
                "inters": selectedInters,
                "bday": selectedDate,
                "edu": selectedEdu,
                "city": selectedCity,
```

Рисунок 22. Асинхронний метод реєстрації користувача..