Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему *Спосіб оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Освітня програма *Комп'ютерні науки*
Назва освітньої програми

Виконав: *студент 4 курсу, група КН-19-1* _____ *Д.І. Жук*
Курс, група виконавця | Підпис | Ініціали, прізвище

Керівник: *викладач кафедри КН* _____ *О.В. Собко*
Науковий ступінь, посада | Підпис | Ініціали, прізвище

Нормоконтроль: *к.т.н., доцент кафедри КН* _____ *Р.О. Багрій*
Науковий ступінь, посада | Підпис | Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КН, д.т.н., професор _____ *О.В. Бармак*
Підпис | Ініціали, прізвище

01 06 2023 р.

Хмельницький 2023

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *інформаційних технологій*
Кафедра *комп'ютерних наук*
Освітній ступінь *бакалавр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*
Освітня програма *освітньо-професійна програма підготовки бакалавра*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук*

_____
(підпис)
д.т.н., професор О.В. Бармак
«06» 03 2023 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

1. Тема кваліфікаційної роботи бакалавра: *«Спосіб оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств»*

2. Завдання видано студенту *Жуку Дмитру Івановичу*
(прізвище, ім'я, по батькові)

3. Керівник роботи *викладач кафедри КН Собко Олена Віталіївна*
(посада, прізвище, ім'я, по батькові)

4. Затверджено наказом університету від «01» 03 2023 р. № 5

5. Дата видачі завдання студенту: «03» 03 2023 р.

6. Зміст пояснювальної записки (перелік задач) та вихідні дані:

Провести аналіз предметної області, визначити особливості застосування фреймового подання знань для задач класифікації та пошуку відповідності. Виконати аналіз існуючих рішень вирішення подібних задач оцінювання сумісності анкет користувачів для сайтів знайомств. Розробити спосіб оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств. Спроєктувати структуру інформаційної системи, що використовує спосіб, реалізувати її і виконати її тестування.

7. Календарний план виконання кваліфікаційної роботи бакалавра:

№	Назва етапів (розділів) кваліфікаційної роботи бакалавра	Термін виконання	Примітка
1	*Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи бакалавра з керівником*	*грудень 2022*	виконано
2	*Ознайомлення з предметною областю, формулювання мети та задач дослідження, визначення об'єкта та предмета дослідження*	*січень 2023*	виконано
3	*Робота над розділом 1 – Характеристика предметної області та постановка задачі*	*січень 2023*	виконано
4	*Робота над розділом 2 – Спосіб оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств*	*березень 2023*	виконано
5	*Робота над розділом 3 – Програмна реалізація соціальної інтелектуальної вебсистеми для знайомств*	*квітень 2023*	виконано
6	*Оформлення пояснювальної записки згідно вимог*	*травень 2023*	виконано
7	*Попередній захист кваліфікаційної роботи бакалавра*	*травень 2023*	виконано
8	*Захист кваліфікаційної роботи бакалавра на засіданні Екзаменаційної комісії*	*червень 2023*	виконано

Виконавець: *студент 4 курсу, група КН-19-1* (ZDM) *Д.І. Жук*
Курс, група виконавця — Підпис — Ініціали, прізвище

Керівник: *викладач кафедри КН* *О. Собко* *О.В. Собко*
Науковий ступінь, посада — Підпис — Ініціали, прізвище

Анотація

Тема кваліфікаційної роботи бакалавра: *Спосіб оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств*

Виконавець кваліфікаційної роботи бакалавра: *студент групи КН-19-1 Жук Дмитро Іванович*

Керівник кваліфікаційної роботи бакалавра: *викладач кафедри КН Собко Олена Віталіївна*

Кваліфікаційна робота бакалавра містить:

Пояснювальна записка				Кількість додатків
Сторінок	Рисунків	Таблиць	Джерел інформації	
71	39	26	39	13

Кваліфікаційна робота бакалавра присвячена розробці способу оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств та відповідної інформаційної системи у вигляді соціальної інтелектуальної вебсистеми для знайомств, що використовує розроблений спосіб оцінювання сумісності анкет користувачів за фреймовим поданням знань.

Результатом виконання кваліфікаційної роботи бакалавра є створений та програмно реалізований спосіб порівняння сумісності анкет на прикладі сайту знайомств. Після порівняння анкет за наявними критеріями, система відображає відсоток сумісності анкет, що обчислюється на основі співпадінь заданих критеріїв.

Ключові слова: сайт знайомств, анкета, фреймове поданням знань, фрейм, поданням знань, сумісність анкет, соціальна інтелектуальна система.

Виконавець: *студент 4 курсу, група КН-19-1* (ZDm) *Д.І. Жук*

Курс, група виконавця — Підпис — Ініціали, прізвище

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
БД	База даних
КРБ	Кваліфікаційна робота бакалавра
КН	Комп’ютерні науки
ПК	Персональний комп’ютер
UML	Unified Modeling Language
XML	Extensible Markup Language
AWS	Amazon Web Services

Вступ

Кваліфікаційна робота бакалавра присвячена розробці способу оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств та відповідної інформаційної системи у вигляді соціальної інтелектуальної вебсистеми для знайомств, що використовує розроблений спосіб оцінювання сумісності анкет користувачів за фреймовим поданням знань.

Актуальність. Раніше люди знайомились на вулиці, у барах чи на інших громадських заходах, тобто у місцях де присутні інші люди. Сьогодні, до цього методу можна додати ще один – це знайомства онлайн. У знайомств онлайн є певні переваги. У першу чергу, якщо людина не сподобалась, можна просто ігнорувати її. Також, великий наплив та кількість активних користувачів таких сервісів надає великий вибір кандидатів на спілкування, що призводить до необхідності їхньої фільтрації. У XIX столітті, ця процедура відбувалась, на рівні батьків. Зазвичай просто говорили, за кого ти повинна вийти заміж чи на якій дівчині одружитись, але зараз ця процедура змінилась, і у багатьох випадках віддається свобода у спілкуванні та обранні партнера.

Зараз люди проводять багато часу у гаджетах, ПК, ноутбуках чи телефонах. Багато часу витрачається на спілкування, а ще більше, на пошуки співрозмовників. Отже, актуальність сайтів знайомств залишається на високому рівні, адже для пошуку партнера люди розуміють якими вони хочуть бачити свого ідеального партнера (зріст, колір очей, хоббі, робота).

Об’єкт дослідження – процес автоматизованого оцінювання сумісності анкет користувачів на прикладі сайту знайомств.

Предмет дослідження – моделі, методи, алгоритми та засоби для визначення рівня сумісності анкет користувачів для соціальної інтелектуальної вебсистеми знайомств.

Мета кваліфікаційної роботи бакалавра – розробка та програмна реалізація способу оцінювання сумісності анкет користувачів за фреймовим

поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств.

Завдання кваліфікаційної роботи бакалавра – Провести аналіз предметної області, визначити особливості застосування фреймового подання знань для задач класифікації та пошуку відповідності. Виконати аналіз існуючих рішень вирішення подібних задач. Розробити спосіб оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств. Розробити модель фрейму для узагальненого подання анкет користувачів та критеріїв пошуку користувачів для порівняння. Спроєктувати структуру інформаційної системи у вигляді соціальної інтелектуальної вебсистеми для знайомств, що використовує спосіб оцінювання сумісності анкет користувачів за фреймовим поданням знань. Створити соціальну інтелектуальну вебсистему для знайомств, що використовує розроблений спосіб, виконати її тестування.

Розділ 1 Характеристика предметної області та постановка задачі

1.1 Аналіз предметної області

Сайт знайомств – це онлайн-ресурс, який дозволяє людям шукати однодумців для спілкування або потенційних партнерів для створення романтичних стосунків [1]. Цей вебсайт надає користувачам можливість створити профіль, де вони можуть розмістити свої особисті дані, фотографії та інформацію про свої інтереси та уподобання. Вони можуть переглядати профілі інших користувачів та надсилати повідомлення тим, у кому вони зацікавлені. Сайти знайомств є зручним інструментом для знаходження людей, які мають спільні інтереси і цілі у відносинах, і можуть допомогти утворити нові знайомства та відкрити нові можливості для особистого життя [2]. В XXI столітті більшість людей сильно залежить від різних гаджетів, таких як смартфони, планшети, ноутбуки та ПК. Ці пристрої стали невіддільною частиною їх повсякденного життя і дозволяють проводити значну кількість часу в Інтернеті.

Для того, щоб розпочати пошук людей на сайтах знайомств, необхідно створити акаунт та заповнити анкету. У цій анкеті користувачі надають інформацію про себе, що допомагає знайти відповідність інтересів та потреб. Зазвичай, у анкеті зазначаються особисті дані, такі як ім'я, дата народження, зріст, вага, статура, а також інформація про хобі, роботу та шкідливі звички. Крім того, в анкеті можна вказати свої цілі щодо знайомств, наприклад, пошук серйозних стосунків або просто спілкування.

Ця інформація допомагає підібрати потенційні знайомства, що відповідають особистим вимогам та уподобанням користувача. Вона сприяє покращенню шансів зустріти відповідну людину, з якою можна буде розвивати стосунки на основі спільних інтересів та цілей.

Щоб мати змогу користуватись сайтом знайомств потрібно створити акаунт на обраному сайті. Акаунт – це інформаційний запис, що містить дані про користувача та призначений для ідентифікації та надання йому відповідних прав

доступу [3]. Для наведення прикладів акаунтів користувачів було обрано сайт «juliadates» (рисунки 1.1, 1.2) [4].

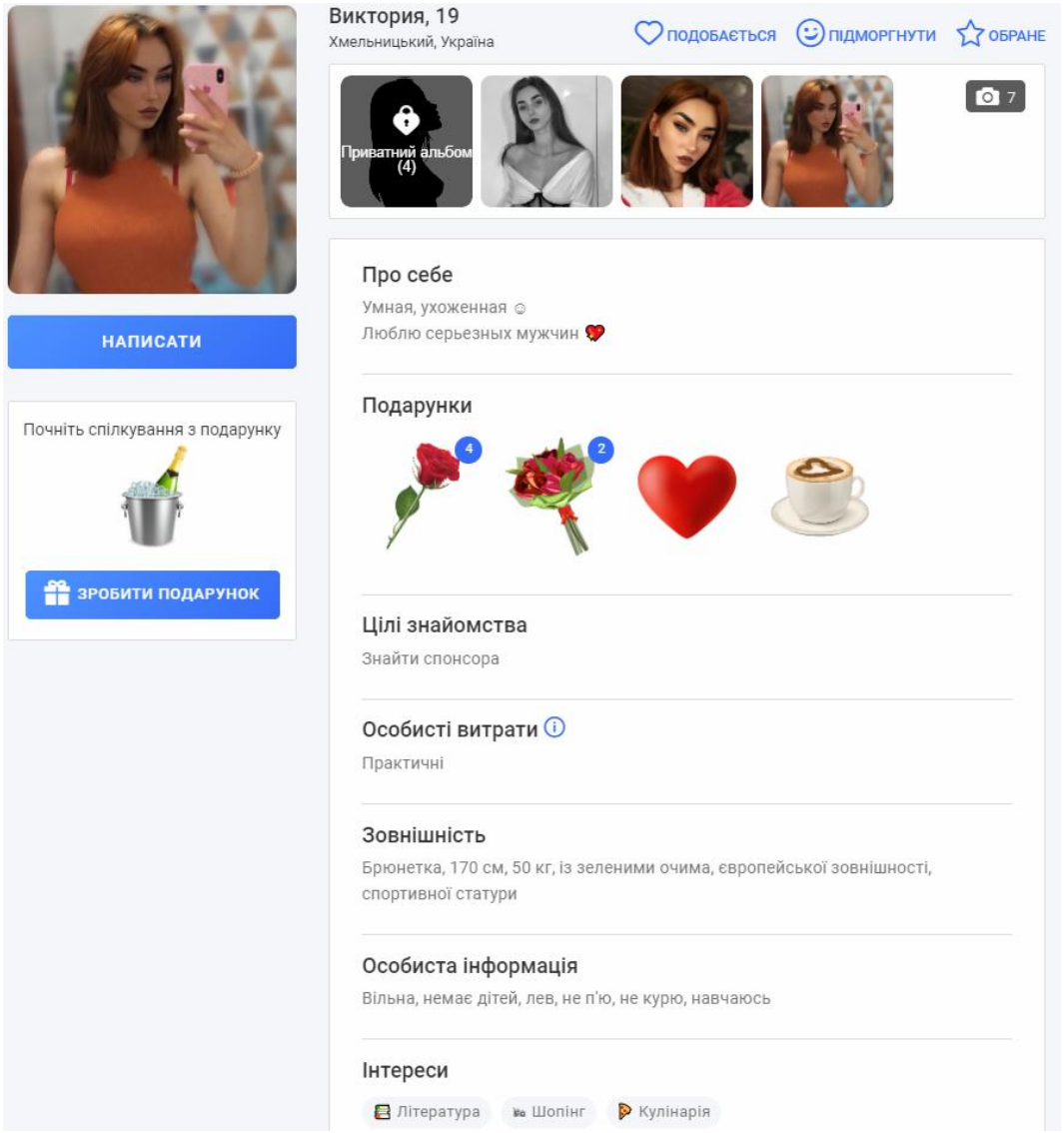


Рисунок 1.1 – Анкета користувачки сайту «JuliaDates» [5]

У XXI столітті більшість людей проводять значну кількість часу зі своїми смартфонами. Вони спілкуються з родичами та друзями у месенджерах, переглядають фотографії в «Instagram» [7] або читають публікації на «Reddit» [8]. Тому, знайомства в Інтернеті набувають все більшої популярності і стають актуальною темою.

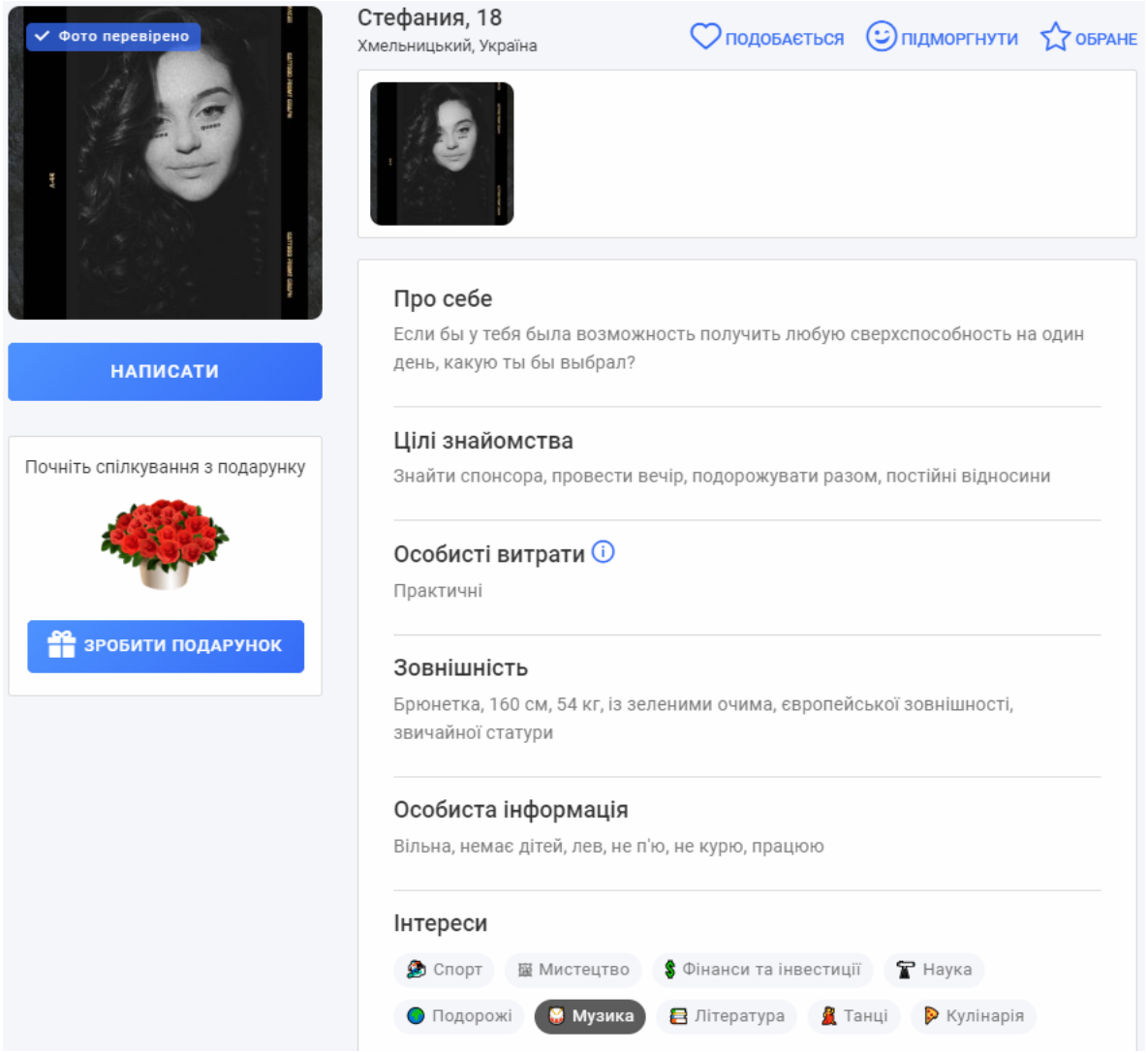


Рисунок 1.2 – Анкета користувачки сайту «JuliaDates» [6]

Багато людей мріють знайти собі життєвого партнера, з яким можна пройти усе життя. Тому, дослідження та розробка методів оцінювання сумісності профілів користувачів за допомогою фреймового представлення знань для соціальної інтелектуальної системи на прикладі сайтів знайомств є актуальною задачею.

1.2 Особливості застосування фреймового подання знань для задач класифікації та пошуку відповідності

Відповідно до теорії пізнання, знання можуть бути набуті шляхом досвіду, спостережень та роздумів [9]. Знання означають розуміння чогось,

представленого у вигляді фактів, досліджень або конкретної інформації. Важливо вміти ефективно вилучати та зберігати ці знання.

Вилучення знань – передбачає створення знань з різних джерел, які можуть бути структурованими або неструктурованими [9]. Знання, які отримані в результаті вилучення, повинні бути придатними для автоматичного відтворення та інтерпретації, що полегшує логічні висновки.

Структуровані знання зберігаються у вигляді реляційних баз даних або XML-файлів. Неструктуровані знання зберігаються у форматі зображень, текстів або документів.

Перед вибором певного типу обробки даних важливо проаналізувати всі можливі варіанти моделей представлення знань. Є чотири моделі представлення знань:

- продукційна модель;
- формальна логічна моделі;
- семантична мережа;
- фрейм.

Продукційна модель – є базовою моделлю, яка ґрунтується на правилах. Ця модель дозволяє представити знання у форматі «Якщо... то», відомому також як «Умова... дія». У продукційній моделі кожне правило складається з умови або набору умов, які потрібно перевірити (якщо), і з відповідної дії, яку потрібно виконати, якщо умова виконується (то). Це дозволяє системам знань виробляти логічні висновки та приймати рішення, базуючись на наборі правил і фактів.

У продукційній моделі, умова, також відома як антецедент, представляє собою пропозицію або зразок, за яким здійснюється пошук у базі даних або базі знань. Умова містить інформацію, яка потрібна для визначення відповідності або невідповідності до певного критерію.

Дія, відома як консеквент, є набором дій, які виконуються, якщо умова антецедента задовольняється або відповідає критерію. Консеквент містить інструкції або дії, які повинні бути виконані або здійснені у випадку, коли виконується умова.

Наприклад, можна змоделювати ситуацію поїздки у певне місце в якій потрібно підібрати авто.

Умова 1:

Якщо місце відпочинку – це гори, то дорога вибоїста та повна болота.

Умова 2:

Якщо намір поїздки – відпочинок та дорога вибоїста то слід використовувати позашляховик.

Формальна логічна модель розглядає всю необхідну інформацію як набір фактів або тверджень, що можуть бути використані для побудови логічних ланцюжків.

Семантична мережа, також відома як смислова мережа, є графом, у якому об'єкти або абстрактні концепції представлені вершинами, а відношення між цими об'єктами відображаються за допомогою дуг.

Фреймова модель є структурою даних, яка слугує для опису певного об'єкта або абстрактного концепту. Ця модель надає абстрактний шаблон для представлення певної концепції. Кожен фрейм включає слоти, які визначаються фасетами або характеристиками [12]. Щоб візуально представити структуру фрейму у вигляді таблиці, потрібно додати два стовбці, які відображатимуть характеристики або значення кожного слоту.

У таблиці 1.1 додаткові колонки використовуються для опису та отримання знань про конкретний слот [13]. Також, ім'я слоту може бути пов'язане з назвою іншого фрейму, що дає можливість побудувати мережі фреймів.

Таблиця 1.1 – Структура фрейму

Ім’я слоту	**Значення слоту**	**Спосіб отримання значення**

Фрейми мають декілька основних переваг. Перша з них – вкладеність, що забезпечує структурованість, спадковість та зв'язок між знаннями. Друга перевага – внутрішня інтерпретованість, що означає, що значення в фреймах

мають характер посилань. Проте, фрейми також мають певні обмеження. Наприклад, вони не є підходящими для представлення процедурних знань, що є основним недоліком [14].

Процедурні знання є формою знань, які зберігаються в пам'яті інтелектуальної системи у вигляді опису конкретних процедур [15]. Вони містять інформацію про методи та процедури вирішення завдань у конкретній предметній області. Процедурні знання відрізняються від декларативних знань, оскільки останні представляють факти та інформацію про стан речей. Приклад фреймової моделі зображено на рисунку 1.3

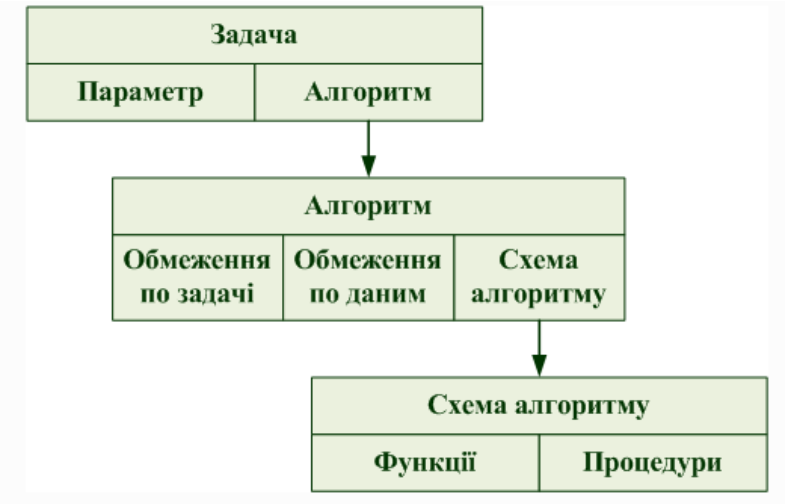


Рисунок 1.3 – Приклад фреймової моделі ієрархічного типу [16]

Фреймове представлення знань для задач класифікації та визначення відповідності можна розглядати як спробу систематизувати та організувати знання, щоб полегшити їх застосування у практичних ситуаціях. Використання фреймів дозволяє представляти різноманітні види знань. Наприклад, в дослідженні, згаданому у статті [17], досліджується використання фреймових знань для отримання інформації про знання та навички школярів. Кожен літературний твір розглядається як окремий фрейм, а автор описує, як за допомогою фреймів можна структурувати шкільну програму і надати їй форму сюжету або певної хронології. Це дозволяє забезпечити школярів цілісним та структурованим способом отримання знань.

У статті [18] досліджується ефективність алгоритмів, їх порівняння та передбачення подальшого розвитку. Використанням алгоритму подвійної згоди, який базується на взаємній зацікавленості сторін у спілкуванні, «Tinder» досягає успішних результатів. Крім того, на платформі «Tinder» присутня система «Elo», яка взята зі світу шахів і призначена для присвоєння рейтингу користувачам на основі їхніх результатів та перемог. Ця система допомагає підібрати співрозмовників з однаковим рівнем інтересів.

Фреймове подання знань забезпечує структурованість, що у свою чергу сприяє подальшому пошуку кандидатів за певними критеріями. Оскільки анкети на сайтах знайомств мають структурований формат згідно з певними принципами, використання фреймів для представлення знань про анкети користувачів є доцільним підходом.

1.3 Аналіз існуючих рішень для подібних задач

У теперішній час є багато сайтів для знайомств. Згідно зі статистикою, найпопулярнішими в українців є наступні сайти [19]:

– Tinder [20];

– Badoo [21];

– Edarling [22];

– Juliadates [4];

– Jolly [23].

Для ілюстрації алгоритму підбору анкет було обрано веб-сайт «juliadates». Щоб ознайомитись з роботою алгоритму, необхідно спочатку пройти реєстрацію і відповісти на перелік питань:

– розкажіть про свої цілі на сайті? (Стати спонсором, постійні стосунки, подорожувати разом, провести вечір);

– сума на місяць Вам необхідна для підтримки вашого рівня життя;

– розкажіть про свої інтереси;

– рід зайнятості, відносини, ставлення до алкоголю та паління;

– вказати дані зовнішності: зріст, вага, статура, колір волосся, колір очей;

– розповісти декілька слів про себе.

На рисунку 1.4 показано результат реєстрації нової анкети користувача на сайті знайомств «juliadates».

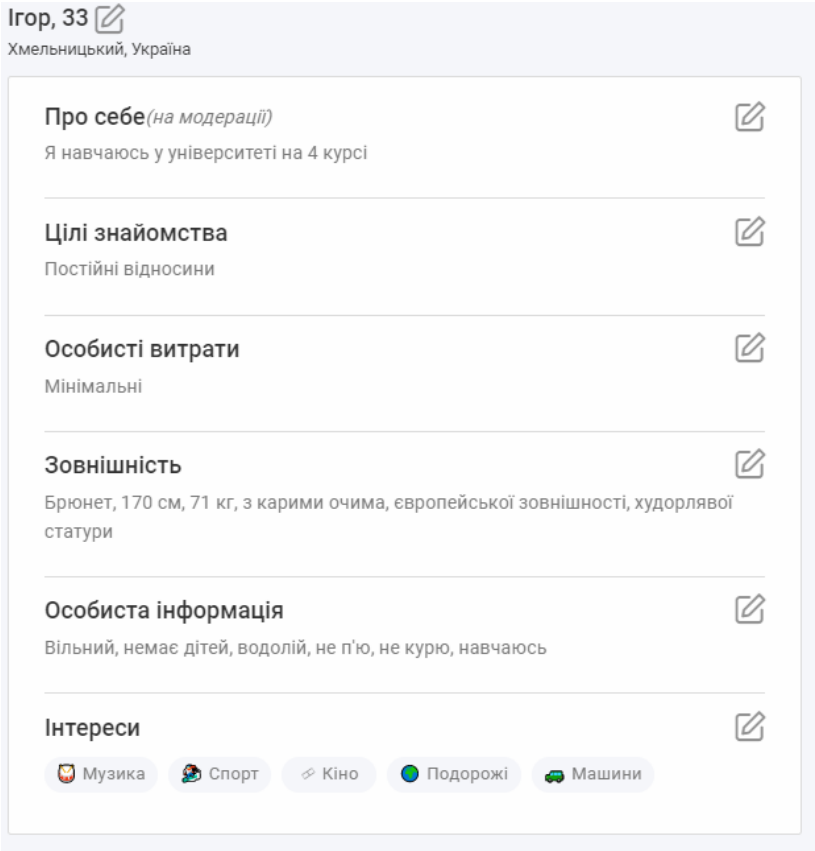


Рисунок 1.4 – Зареєстрований обліковий запис

Після переходу у розділ «Знайомства» було обрано дві випадкові анкети. Результат підбору анкет сайтом зображено на рисунку 1.5, та рисунку 1.6

У першій анкеті можна помітити співпадіння інтересів та місця проживання.

У цьому випадку, також, спостерігається врахування співпадіння інтересів і місця проживання. Це дозволяє зробити висновок, що на сайті пропонуються анкети людей з подібними інтересами, враховуючи якомога ближче місце проживання, відповідно до місця проживання зазначеного у профілі користувача.

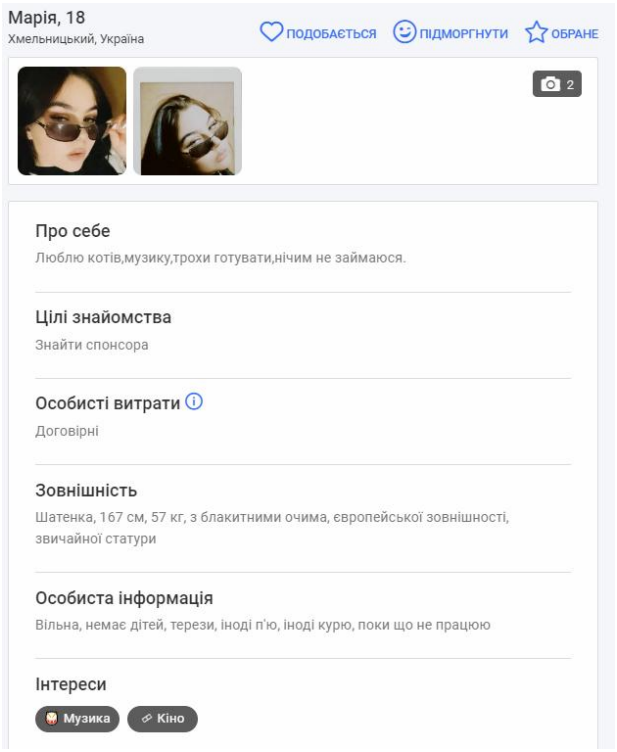


Рисунок 1.5 – Анкета користувачки Марії [24]

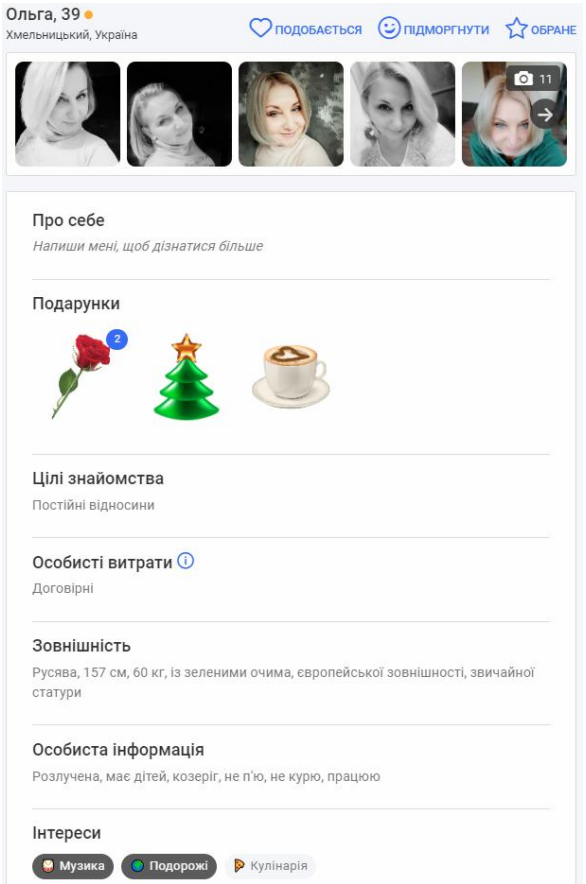


Рисунок 1.6 – Анкета користувачки Ольги [25]

Другим сайтом, що був використаний для аналізу, є сайт знайомств «Jolly.me». Процес реєстрації профілю на цьому сайті аналогічний до «juliadates», проте процес підбору анкет є випадковим.

Третім прикладом популярного сайту знайомств є «Badoo». Зареєстрований обліковий запис на цьому сайті має вигляд, як показано на рисунку 1.7.

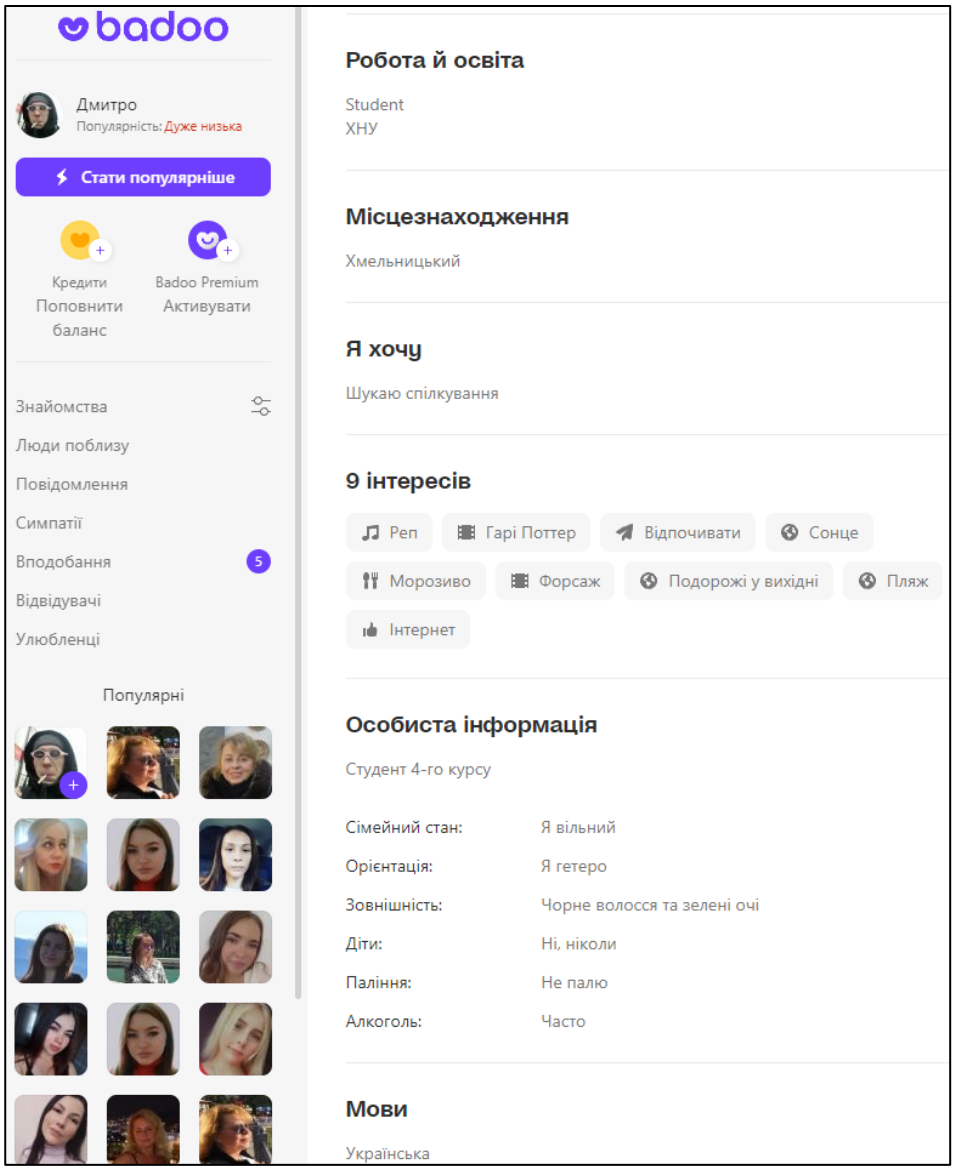


Рисунок 1.7 – Анкета користувача Дмитра [26]

Підбір анкет на сайті здійснюється за допомогою методу взаємних вподобань, але для того, щоб побачити, кому сподобався ваш профіль, потрібно придбати преміум-пакет. Це є основним недоліком цього сайту.

Отже, після аналізу наявних рішень стає очевидним, що більшість збігів анкет на сайтах знайомств відбувається за певними критеріями, такими як місто, інтереси або хобі. Проте жоден із згаданих сайтів не надає повної інформації про оцінку сумісності. Вони не надають експертну оцінку, висновки або пояснення щодо того, чому саме ці анкети рекомендовані.

1.4 Мета, задачі та вимоги до реалізації програмної системи

Метою роботи є розробка та програмна реалізація способу оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств, для чого потрібно вирішити наступні задачі:

– розробити спосіб оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств;

– розробити модель фрейму для узагальненого подання анкет користувачів та критеріїв пошуку користувачів для порівняння;

– спроєктувати структуру інформаційної системи у вигляді соціальної інтелектуальної вебсистеми для знайомств, що використовує спосіб оцінювання сумісності анкет користувачів за фреймовим поданням знань;

– спроєктувати структуру бази даних для соціальної інтелектуальної вебсистеми для знайомств;

– виконати вибір засобів розробки соціальної інтелектуальної вебсистеми для знайомств;

– виконати програмну реалізацію інформаційної системи у вигляді соціальної інтелектуальної вебсистеми для знайомств;

– провести тестування розробленої соціальної інтелектуальної вебсистеми для знайомств.

1.5 Висновки до розділу 1

Загалом, дослідження в обраному напрямку оцінювання сумісності анкет користувачів стає все більш актуальним у сучасний час. Це пов'язано з розвитком засобів знайомств і комунікації. У XXI столітті люди проводять багато часу онлайн, використовуючи смартфони, планшети, ноутбуки та персональні комп'ютери.

Для оцінювання сумісності анкет користувачів було вирішено використовувати фреймове подання знань. Фреймове подання забезпечує структурованість та простоту використання даних як для аналізу, так і для візуалізації.

У результаті аналізу наявних рішень виявилося, що в більшості випадків все обертається навколо придбання преміум-підписки. Ця підписка надає переваги профілю користувача, такі як вищий рейтинг або можливість спілкуватись на сайті. Вищий рейтинг підвищує видимість анкети у пошуку, тоді як анкети без підписки знаходяться на більш віддалених сторінках.

Розділ 2 Спосіб оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств

2.1 Опис способу оцінювання сумісності анкет користувачів за фреймовим поданням знань на прикладі сайту знайомств

Для оцінювання сумісності анкет користувачів за допомогою фреймового подання знань можна використовувати алгоритм порівняння фреймів. Цей алгоритм порівнює вимоги та очікування користувачів, які були введені в анкеті, з даними інших користувачів на сайті знайомств.

У алгоритмі порівняння фреймів можна використовувати критерії для оцінювання збігів. В даному випадку ці критерії включають два опитування для формування відповідних фреймів. Перший фрейм містить інформацію про користувача, а другий фрейм містить інформацію про бажаного партнера. Основні критерії для порівняння фреймів:

- спільні інтереси;
- сумісність освіти;
- сумісність сімейного стану;
- сумісність фізичних характеристик.

Для початку аналізу анкети системою, користувачу потрібно встановити характеристики та критерії, за якими система буде перевіряти сумісність анкет. Перевірка цих анкет буде здійснюватися шляхом виконання певного тесту, в якому за кожен критерій співпадіння будуть надаватись бали.

Для визначення ступеня сумісності між користувачами необхідно встановити правила відповідності. Наприклад, якщо користувач А шукає партнера у віковому діапазоні від 25 до 30 років, то користувачі, чий вік не відповідає цьому діапазону, не будуть отримувати бали за сумісність.

Після порівняння анкет за цими критеріями, система відображає відсоток сумісності анкет. Тобто, система порівнює два фрейми: фрейм конкретного користувача та фрейм очікувань користувача. Результатом сумісності є число,

яке обчислюється на основі співпадінь заданих критеріїв. Чим більше критеріїв співпадає, тим вище буде відсоток сумісності.

Математична модель способу сумісності буде мати наступний вигляд:

$$\sum_{n=1}^{k} \frac{1}{k}, \tag{2.1}$$

де k – загальна кількість критеріїв оцінювання.

Якщо у критеріїв є підкритерії, то максимальний відсоток за критерій буде сумою співпадінь підкритеріїв. Це потрібно, наприклад, для хобі. Якщо загальна кількість критеріїв 10, то за співпадіння одного критерію додається 10%. Якщо, наприклад, цей критерій це хобі користувача, то їх може бути багато. Для прикладу, якщо в користувача є чотири хобі, то за співпадіння одного буде додаватись по 2.5%. Схему етапів порівняння зображено на рисунку 2.1.

Вхідними даними для системи є дані про бажаного користувача. У цьому наборі даних вказані: стать, вік, зріст, рівень доходу, освіта, професія, місто, хобі, та сімейний стан.

Етап 1. На цьому етапі система формує із вхідних даних фрейм для подальшого порівняння.

Етап 2. Система порівнює фрейми користувачів та шуканий фрейм. У відповідності до співпадіння між даними фреймами кожен фрейм має власний відсоток співпадіння. Після порівняння усіх користувачів система автоматично відсіює усі анкети у яких відсоток співпадіння менший 50%.

Етап 3. Система вказує який із параметрів кандидата не співпав із очікуваним результатом. Сортує список анкет за відсотком співпадіння, та надсилає користувачеві 5 найкращих кандидатів.

Вихідні дані. Система формує експертний висновок для кожної анкети. Користувачеві пропонується список із 10 найкращих кандидатів.

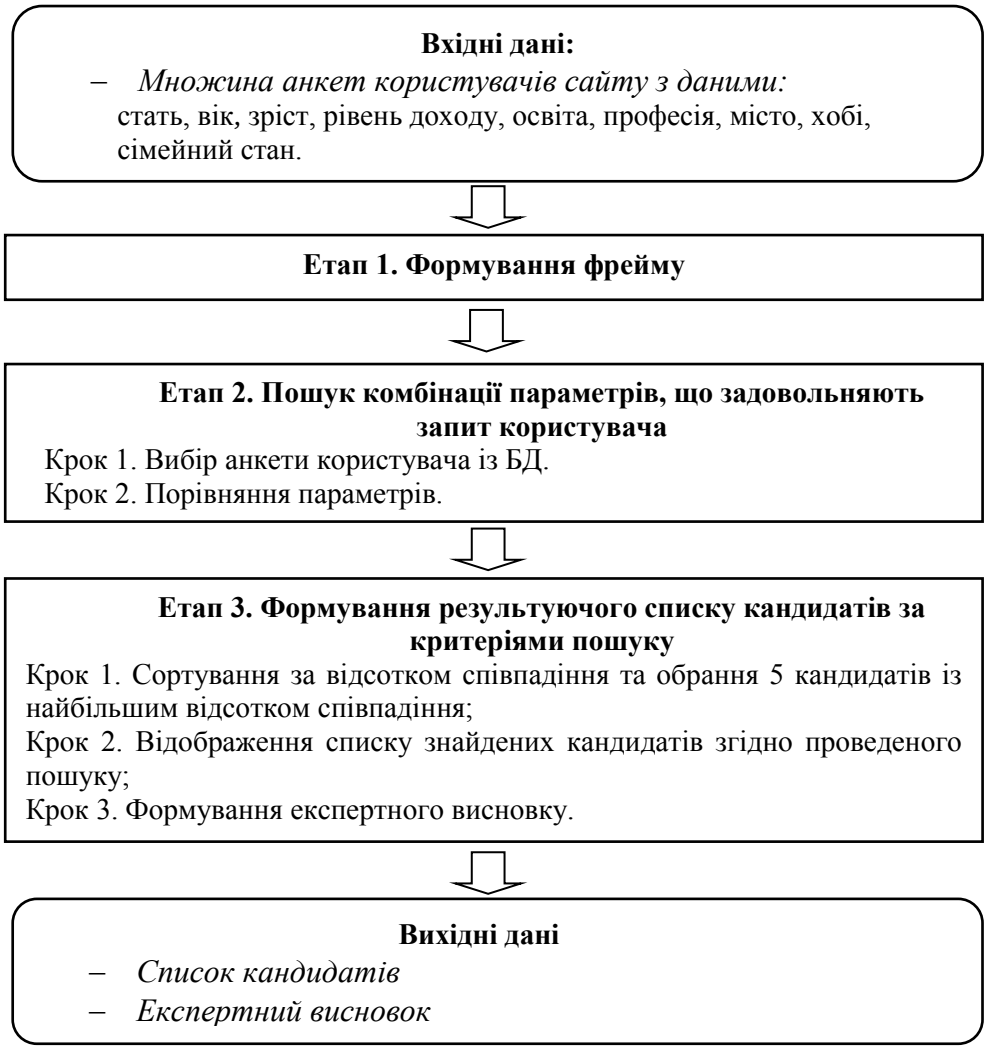


Рисунок 2.1 – Схема способу оцінювання сумісності анкет користувачів

Отже, для визначення сумісності між користувачами система використовує тест у якому за кожен критерій співпадіння будуть додаватись бали. Після порівняння анкет за цими критеріями, система відображає відсоток сумісності анкет, що обчислюється на основі співпадінь заданих критеріїв. Чим більше критеріїв співпадає, тим більший відсоток співпадіння.

2.2 Розробка моделі фрейму для узагальненого подання анкет користувачів та критеріїв пошуку користувачів

Для кожного користувача на сайті знайомств можна створити множину, яка включає значення кожного атрибута. Наприклад, для користувача з іменем «Іван», прізвищем «Петрів», віком 25 років, статтю «чоловік», місцем

проживання «Київ», статусом відносин «неодружений», ростом 180 см, без зайвої ваги, зовнішністю «спортивна», хобі та інтересами «футбол, музика», інформацією про роботу та освіту «студент», та фотографіями створено таку множину:

{Ім'я = «Іван», Прізвище = «Петрів», Вік = 25, Стать = «чоловік», Місце проживання = «Київ», Статус відносин = «неодружений», Висота = 180, Вага = 70, Зовнішність = «спортивна», Хобі та інтереси = {«футбол», «музика»}, Інформація про роботу та освіту = «студент», Фотографії = {...}}.

Така ж множина створюється для кожного користувача і порівнюється з множинами інших користувачів для пошуку подібних профілів.

Таблиця 2.1 – Структура фрейму користувача

Фрейм «Користувач»	
Ім’я	*Ім’я користувача*
Прізвище	*Прізвище користувача*
Електронна адреса	*Електронна адреса користувача*
Номер телефону	*Номер телефону користувача*
Рівень доходу	*Діапазони доходів*
Вік	*Вік користувача*
Освіта	*Освіта користувача*
Професія	*Назва фрейму Професія*
Хобі	*Назва фрейму хобі*
Місце проживання	*Місто*
Зріст	*Зріст у см*
Вага	*Число у кг*
Стать	*Чоловік або жінка*
Сімейне положення	*Не одружений/на, розлучений/на*
Тип особистості	*Інтроверт, екстраверт або амбіверт.*

Крім того, можна використовувати математичні моделі для порівняння різних атрибутів користувачів, наприклад, за допомогою матриць. Будуючи матрицю порівняння зовнішнього вигляду користувачів на основі їхніх

фотографій з різними ознаками зовнішності, можна здійснювати порівняння між користувачами на основі їхніх атрибутів та оцінювати їхню подібність. Для опису анкет був розроблений основний фрейм користувача (Таблиця 2.1).

Даний фрейм містить такі поля: Ім'я, Прізвище, Електронна адреса, Номер телефону, Рівень доходу, Вік, Освіта, Хобі, Професія, Місце проживання, Зріст, Вага, Стать, Сімейний стан та Тип особистості

Для опису сутності професії було створено фрейм «Професія», який містить у собі наступні поля: Назва, Посада, Компанія (Таблиця 2.2).

Для опису сутності хобі створено фрейм «Хобі», який містить у собі наступні поля: Назва, Опис (Таблиця 2.3).

Таблиця 2.2 – Структура фрейму «Професія»

Фрейм «Професія»	
Назва	Назва професії
Посада	Посада на роботі
Компанія	Назва компанії

Таблиця 2.3 – Структура фрейму «Хобі»

Фрейм Хобі	
Назва	Назва хобі
Опис	Опис хобі

Для того щоб користувачі могли шукати партнерів за певними критеріями було створено фрейм «Критерії пошуку», який містить наступні поля: Дохід, Вік, Хобі. Освіта, Стать, Сімейний стан, Зріст, Вага, Професія та Місце проживання (Таблиця 2.4).

Отже, було створено основний фрейм користувача, який включає особисті дані, такі як ім'я, прізвище, електронна адреса, номер телефону, рівень доходу, вік, освіта, професія, хобі, місце проживання, статура, зріст, вага, стать, сімейний стан та тип особистості.

Таблиця 2.4 – Структура фрейму «Критерії пошуку»

Фрейм «Критерії пошуку»	
Дохід	Діапазон очікуваних доходів
Вік	Діапазон віку
Хобі	Перелік хобі
Освіта	Ступінь освіти
Стать	Чоловік або жінка
Сімейний стан	Не одружений/на, розлучений/на
Зріст	Зріст у см
Вага	Число у кг
Професія	Професія
Місце проживання	Місто

Крім того, були створені фрейми «Професія» та «Хобі», що містять додаткову інформацію про ці дані. Також, був розроблений фрейм «Критерії пошуку», який дозволяє користувачам здійснювати пошук партнерів за обраними критеріями. В цілому, створені фрейми дають повну картину користувача та його вимог до потенційного партнера, що є важливим кроком у подальшому розвитку системи знайомств.

2.3 Підготовка робочих вхідних даних для системи

Для підготовки даних та подальшого тестування системи потрібен великий датасет. Для цього потрібно підготувати перелік наступних даних.

- чоловічі імена;
- жіночі імена;
- прізвища;
- професії;
- хобі;
- міста;
- електронні адреси;

– номера телефонів.

Для досягнення унікальності датасету потрібно створити значну кількість анкет. Для вирішення цієї задачі можна ефективно використати «Фабрику анкет». В рамках цієї фабрики всі співпадіння будуть генеруватися випадковим чином.

Для генератора підготовлено списки чоловічих [27] та жіночих [28] імен, прізвищ [29], хобі [30] та міст. Щоб забезпечити унікальність електронної адреси, назва скриньки також буде генеруватися випадковим чином. Закінчення адреси будуть вибиратися зі списку наступних варіантів:

– example@tutanoda.com;
– example@mail.com;
– example@outlook.com;
– example@zoho.com;
– example@yahoo.com;
– expample@gmail.com.

Номер телефону, також, створюватимуться випадковим чином.

Для реалізації методу оцінювання сумісності анкет користувачів на основі фреймового подання знань на сайті знайомств була створена множина, що включає значення кожного атрибуту користувача. Це необхідно для порівняння та ефективного пошуку потенційних партнерів серед цих анкет. Для цього використовується великий датасет, який можна створити за допомогою фабрики анкет.

2.4 Проєктування структури соціальної інтелектуальної вебсистеми для знайомств

Щоб розробити структуру соціальної інтелектуальної вебсистеми потрібно визначити типи користувачів та скласти список функцій для користувачів. У цій вебсистемі є два типи користувачів.

Перший тип – це адміністратор, який буде керувати даними системи, тобто додавати та редагувати. Необхідність у створені цього типу користувача дійсно дуже велика, адже потрібна людина яка буде постійно слідкувати за допоміжними даними та в певній мірі їх модерувати. У адміністратора вебсистеми повинні бути наступні функції:

– додавання нових записів;

– редагування існуючих записів;

– видалення існуючих записів.

Другий тип користувачів – це користувач веб системи. У користувача є наступні функції:

– реєстрація та авторизація у системі;

– створення власної анкети;

– редагування власної анкети;

– можливість зробити запис до системи для знаходження найкращого кандидату відносно свого ідеалу.

Для подальшої розробки структури системи потрібно її поділити на підсистеми:

– підсистема реєстрації та авторизації – це підсистема яка містить усі необхідні функції для реєстрації та авторизації користувачів, функції перевірки на валідність та функції первірки на існування ключових даних облікових записів (логіни та електронні скриньки);

– підсистема формування фрейму анкети користувача – це підсистема яка містить функції для формування, та встановлення зв’язків між фреймами;

– підсистема оцінювання сумісності анкет користувачів – це підсистема яка містить функції для перевірки співпадінь критеріїв, та функцію визначення відсотку за один критерій, та у випадку хобі за одне хобі;

– підсистема формування експертного висновку – це підсистема яка потрібна для формування експертного висновку системи після порівняння критеріїв;

– підсистема роботи із базою даних – це підсистема яка містить функції для додавання, редагування та створення нових записів про сутності системи.

На рисунку 2.2 зображена структура інформаційної системи.

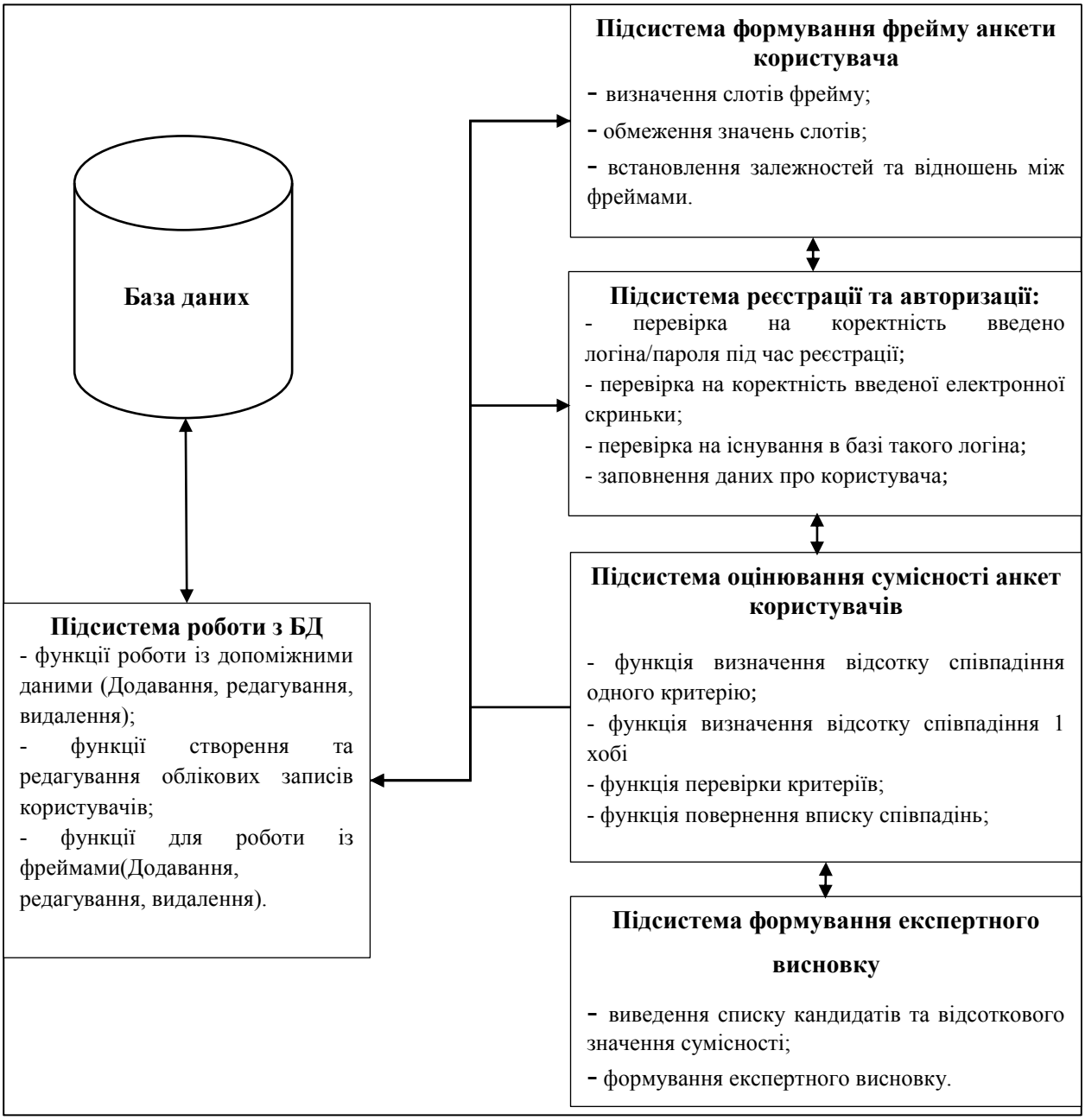


Рисунок 2.2 – Структура інформаційної системи

Також ця схема демонструє зв’язки між підсистемами. Наприклад, якщо користувач заповнив форму для формування анкети ідеалу. Підсистема формування фреймів створює фрейм, та передає його у підсистему для оцінювання сумісності анкет користувачів, після перевірки критеріїв анкети

підсистема формування висновків формує висновок порівняння анкети ідеалу та анкети користувача.

Для розробки соціальної інтелектуальної вебсистеми необхідно створити моделі взаємодії між користувачами та самою системою, а також визначити різноманітні можливі сценарії дій. У цьому можна скористатися UML-діаграмами.

UML є мовою моделювання, яка використовується для створення і опису складних систем, що включають різноманітні компоненти, такі як програмне забезпечення, апаратне забезпечення, бізнес-процеси та інші складні об'єкти [31]. Використання UML-діаграм дозволяє візуалізувати та документувати різноманітні характеристики системи. Ці діаграми детально описують аспекти системи, такі як її структура, поведінка, взаємодія між компонентами та динаміка процесів. Вони є потужним засобом для аналізу, проектування та комунікації між розробниками, архітекторами та іншими учасниками процесу розробки системи.

Для правильного проєктування системи потрібно розробити наступні діаграми:

– діаграма активності користувача (Activity Diagram);

– діаграма розгортання (Deployment Diagram);

– діаграма потоків даних (Data Flow Diagram).

Діаграма активностей (Activity Diagram) є інструментом для візуалізації динамічної поведінки системи (рисунок 2.3). Ця діаграма включає графічне представлення блок-схеми, яка ілюструє послідовність виконання бізнес-процесів і процедур, а також потоки роботи, тобто переходи між різними діями системи [32].

Діаграма розгортання – це візуальне зображення, яке відображає обчислювальні вузли, на яких працює програма, а також компоненти та об'єкти, які функціонують на цих вузлах. Компоненти представляють робочі екземпляри кодових блоків, тому компоненти, які не використовуються під час роботи

програми, не зображуються на цій діаграмі, але можуть бути відображені на діаграмі компонентів.

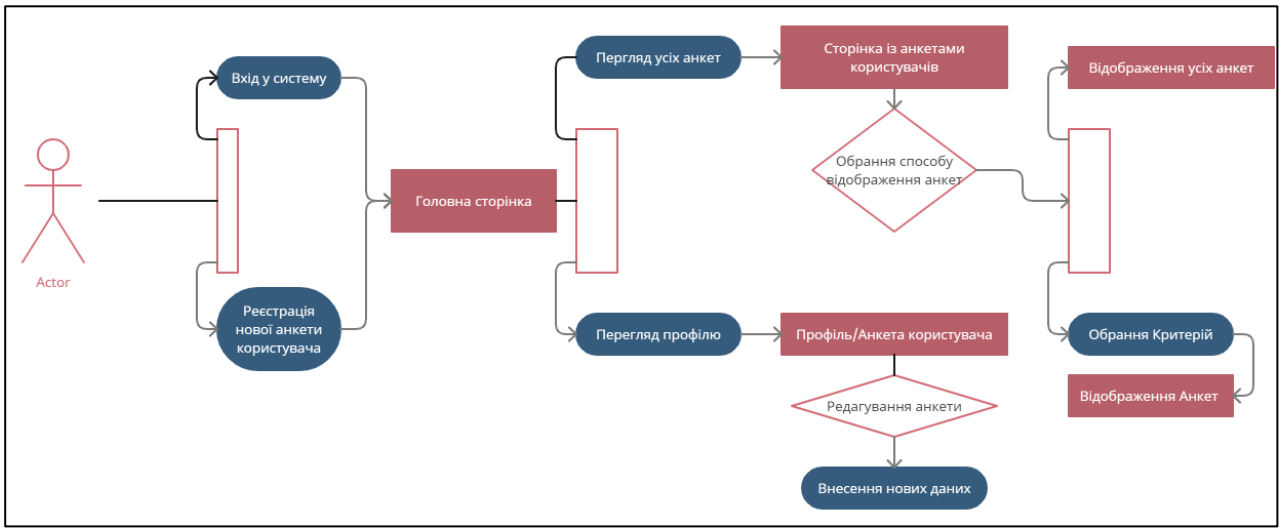


Рисунок 2.3 – UML діаграма активності користувача

Діаграма розгортання (рисунок 2.4) надає інформацію про робочі екземпляри компонентів, в той час як діаграма компонентів показує зв'язки між різними типами компонентів [33].

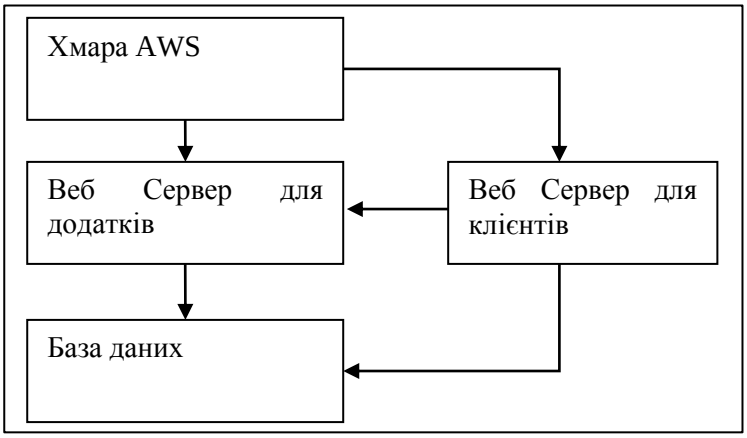


Рисунок 2.4 – Діаграма розгортання системи на сервісі AWS

На цій діаграмі соціальна інтелектуальна вебсистема для знайомств розгортається на хмарному середовищі AWS, що забезпечує високий рівень масштабованості та надійності. Веб-сервер для клієнтів та веб-сервер для додатків розташовані на окремих серверах, це дозволяє забезпечити більшу

гнучкість та можливість розширення системи. База даних розташована на окремому сервері. Клієнти та додатки отримують доступ до веб-серверів через Інтернет з будь-якого місця з використанням стандартних протоколів.

Також, у загальному система повинна мати дві підсистеми. Перша підсистема це власне сам веб сайт знайомств, а друга це підсистема для порівняння анкет за фреймовим поданням знань. Відокремлення сайту від системи порівняння зменшить навантаження на сайт.

Діаграма потоків даних (DFD або Data Flow Diagram) є моделлю проектування, що використовує графічне зображення для відображення "потоків" даних всередині інформаційної системи (рисунок 2.5). Ця діаграма не тільки допомагає візуалізувати потоки даних, але також може бути використана для представлення процесів обробки даних, сприяючи структурному проектуванню [34].

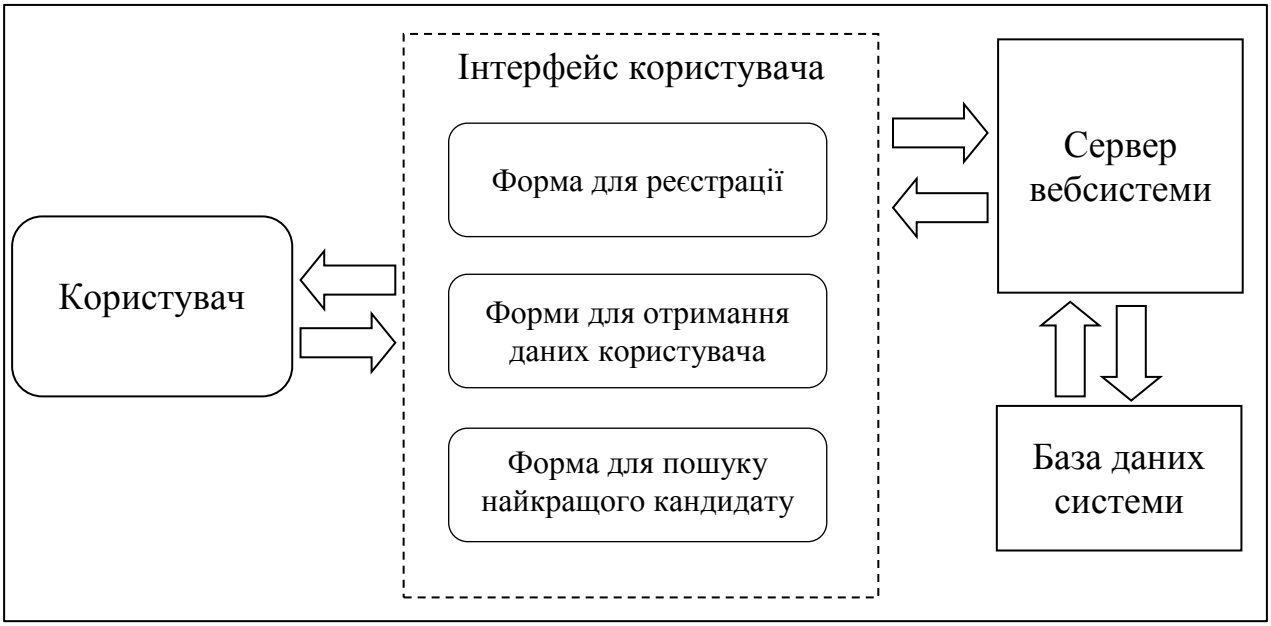


Рисунок 2.5 – Діаграма потоків даних системи

Отже, UML-діаграми є значним інструментом для візуалізації та документування різних аспектів системи, включаючи її структуру, поведінку, взаємодію між компонентами системи, динаміку процесів та інше. Для досягнення цієї мети, необхідно створити кілька типів діаграм, зокрема діаграму

активності користувача, діаграму розгортання та діаграму потоків даних. Діаграма активності користувача відображає послідовність виконання бізнес-процесів та процедур, а також потоки роботи, тобто переходи між різними діями системи. Діаграма розгортання надає зображення обчислювальних вузлів, на яких працює програма, а також компонентів та об'єктів, що функціонують на цих вузлах.

2.5 Даталогічна модель бази даних соціальної інтелектуальної вебсистеми для знайомств

Дана інформаційна система передбачена для використання багатьма користувачами. Отже, потрібно десь зберігати дані про цих користувачів. Для цього потрібно спроєктувати та створити базу даних. Для створення бази даних потрібно сформувати таблиці та відношення між цими таблицями. Потрібно була створена даталогічну модель бази даних, яка зображена на рисунку 2.5.

У цій схемі є наступні сутності:

- фрейми професій;
- фрейми користувачів(UserFrames);
- фрейми хобі;
- міста;
- країни;
- ступені освіти;
- освіти;
- статі.

Для опису сутності професії користувача потрібно створити таблицю із наступними полями: Id, Name, Description (Таблиця 2.6).

Для того щоб за акаунтом користувача можна було закріпити його фрейм потрібно створити таблицю «UserFrames» із наступними полями: Id, Name, Surname, Education, Income, Age, Sex, PersonalityType, Height, Weight, Stature, ProfessionId, CityId (Таблиця 2.7).

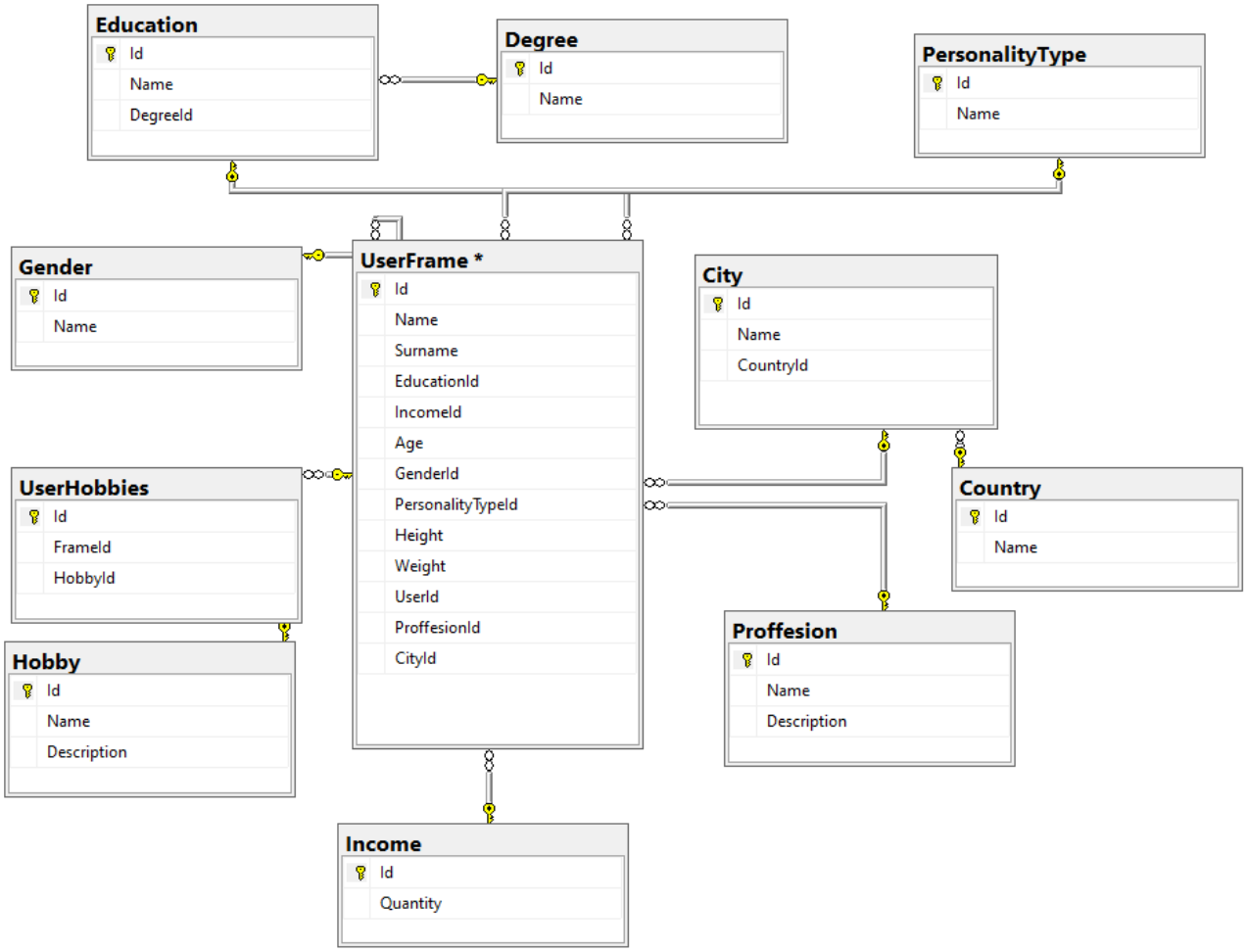


Рисунок 2.6 – Схема даталогічної моделі бази даних

Таблиця 2.6 – Структура таблиці «Professions»

№ поля	Назва	Тип даних	Опис
1	Id	Int	Первинний ключ професії
2	Name	Varchar(70)	Назва
3	Description	Varchar(255)	Опис

Для опису сутності міста проживання користувача потрібно створити таблицю із наступними полями: Id, Name, CountryId (Таблиця 2.8).

Таблиця 2.7 – Структура таблиці «UserFrames»

№ поля	Назва	Тип даних	Опис
1	Id	Int	Первинний ключ фрейму користувача
2	Name	Varchar(70)	Ім’я
3	Surname	Varchar(70)	Прізвище
4	EducationId	Int	Вторинний ключ для таблиці «Education»
5	IncomeId	Int	Вторинний ключ для таблиці «Income»
6	Age	Int	Вік
7	GenderId	Int	Вторинний ключ для таблиці «Gender»
8	PersonalityType	Int	Вторинний ключ для таблиці «PersonalityType»
10	Height	Int	Зріст
11	Weight	Int	Вага
13	ProfessionId	Int	Вторинний ключ для таблиці «Professions»
14	CityId	Int	Вторинний ключ для таблиці «Cities»

Таблиця 2.8 – Структура таблиці «Cities»

№ поля	Назва	Тип даних	Опис
1	Id	Int	Первинний ключ міста
2	Name	Varchar(70)	Назва
3	CountryId	Int	Вторинний ключ для таблиці «Countries»

Для опису сутності країни потрібно створити таблицю із наступними полями: Id, Name (Таблиця 2.9).

Таблиця 2.9 – Структура таблиці «Countries»

№ поля	Назва	Тип даних	Опис
1	Id	Int	Первинний ключ країни
2	Name	Varchar(70)	Назва

Для опису сутності хобі потрібно створити таблицю «Hobbies» із наступними полями: Id, Name, Description (Таблиця 2.10).

Таблиця 2.10 – Структура таблиці «Hobbies»

№ поля	Назва	Тип даних	Опис
1	Id	Int	Первинний ключ хобі
2	Name	Varchar(70)	Назва
3	Description	Varchar(255)	Опис

У користувача може бути багато хобі, і для цього потрібно зробити додаткову таблицю із переліком хобі користувача. У цій таблиці повинні бути наступні поля: Id, UserId, HobbyId (Таблиця 2.11).

Таблиця 2.11 – Структура таблиці «UserHobbies»

№ поля	Назва	Тип даних	Опис
1	Id	Int	Первинний ключ таблиці
2	UserId	Int	Вторинний ключ «UserFrames»
3	HobbyId	Int	Вторинний ключ «Hobbies»

Для опису сутності статі потрібно створити таблицю «Gender» із наступними полями: Id, Name (Таблиця 2.12).

Таблиця 2.12 – Структура таблиці «Gender»

№ поля	Назва	Тип даних	Опис
1	Id	Int	Первинний ключ таблиці
2	Name	Varchar(70)	Назва

Для опису сутності статі потрібно створити таблицю «Gender» із наступними полями: Id, Name (Таблиця 2.13).

Таблиця 2.13 – Структура таблиці «Education»

№ поля	Назва	Тип даних	Опис
1	Id	Int	Первинний ключ таблиці
2	Name	Varchar(70)	Назва
3	DegreeId	Int	Вторинний ключ «Degree»

Таблиця «Degree» створена для запису ступенів освіти та має наступну структуру (Таблиця 2.14).

Таблиця 2.14 – Структура таблиці «Degree»

№ поля	Назва	Тип даних	Опис
1	Id	Int	Первинний ключ таблиці
2	Name	Varchar(70)	Назва

Для опису сутності типу особистості потрібно створити таблицю «PersonalityType» із наступними полями: Id, Name (Таблиця 2.15).

Таблиця 2.15 – Структура таблиці «PersonalityType»

№ поля	Назва	Тип даних	Опис
1	Id	Int	Первинний ключ таблиці
2	Name	Varchar(70)	Назва

Для опису сутності доходу потрібно створити таблицю «Income» із наступними полями: Id, Quantity (Таблиця 2.16).

Рисунок 2.7 – Схема даталогічної моделі бази даних веб сайту

Таблиця 2.16 – Структура таблиці «PersonalityType»

№ поля	Назва	Тип даних	Опис
1	Id	Int	Первинний ключ таблиці
2	Quantity	Varchar(70)	Кількість

Також для збереження даних користувачів веб сайту знайомств було згенеровано наступну даталогічну модель даних (Рисунок 2.6). Ця даталогічна модель БД потрібна для керування акаунтами користувачів, та зв’язування фреймів із акаунтами.

У даному розділі було детально описано необхідність створення бази даних для інформаційної системи, яка передбачена для багатьох користувачів. Наведено структури таблиць та описані поля для зберігання даних про фрейми, професії, хобі, міста та країни, статі, типи особистостей, доходи, освіти, та ступені оствіти. Також було зазначено необхідність створення додаткової таблиці для зберігання інформації про хобі користувачів, оскільки у них може бути багато різних хобі. Використання бази даних дозволить ефективно та зручно зберігати та обробляти дані про користувачів та їхні інтереси.

2.6 Функції для роботи із базою даних

Для правильного збереження даних способу оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств потрібно їх описати у вигляді функцій. У системі повинні бути насупні групи функцій:

– функції для роботи із таблицею Accounts (Додавання нового акаунта, редагування обраного акаунта);

– функції для роботи із таблицею UserFrames (Додавання нового фрейму знань, редагування даних обраного фрейму);

– функції для роботи із таблицею Professions (Додавання нової професії, редагування даних обраної професії, видалення обраної професії);

– функції для роботи із таблицею Cities (Додавання нового міста, редагування даних обраного міста, видалення обраного міста);

– функції для роботи із таблицею Countries (Додавання нової країни, редагування даних обраної країни);

– функції для роботи із таблицею Hobbies (Додавання нового хоббі, редагування даних обраного хоббі);

– функції для роботи із таблицею UserHobbies (Додавання нового хоббі для обраного фрейму, видалення обраного хобі із фрейму).

Отже, для збереження даних та оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств розроблено шість груп функцій для роботи з базою даних.

2.7 Вибір засобів для розробки інформаційної системи

Основною мовою програмування C#(Сі шарп), а основною платформою ASP.NET. C# – це об'єктно-орієнтована мова програмування, розроблена компанією Microsoft[35]. Ця мова програмування є типізованою, що означає, що кожна змінна має певний тип даних, який визначає, яку інформацію вона може зберігати та як її можна використовувати. Також у цій мові програмування є багато додаткових бібліотек які полегшать розробу проєктів будь-яких проєктів.

ASP.NET (Active Server Pages .NET) – це веб-фреймворк, розроблений компанією Microsoft для створення динамічних веб-сайтів, веб-додатків та веб-сервісів[36]. У цього фреймворку є такі переваги як безпека, висока продуктивність так кросплатформенність.

Для розробки буде використано архітектурний шаблон MVC(Model-view-controller). Перага цього шаблону у тому що він дозволяє розділити логіку від представлень. Моделі містять у собі сутності які потрібні для коректної роботи із БД та представленнями. Контролер опрацьовує запити які надає клієнт, та

надсилає дані у представлення. У якості СКБД обрано SQLEXPRESS, який розроблений компанією Microsoft.

2.8 Висновки до розділу 2

Отже, у цьому розділі був розроблений спосіб оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств. Для представлення знань про користувача був створений фрейм «Користувач», який містить відповідні параметри для подальшого порівняння.

Була спроектована структура веб-системи за допомогою UML-діаграм. Була створена діаграма активності (Activity Diagram), яка дозволяє розібрати всі можливі сценарії та дії користувача під час використання програми. Для відображення потоків даних була розроблена діаграма потоків даних системи, відома як «Діаграма потоків даних». Також була створена діаграма розгортання системи з використанням AWS, щоб знизити навантаження на систему та мати можливість масштабування системи у майбутньому.

Для отримання великої кількості профілів був розроблений генератор анкет, для якого були підготовані дані про імена, прізвища, хобі та професії. Для збереження профілів та додаткових даних системи були описані основні сутності: обліковий запис користувача, хобі, місто, країна, стать, освіта, ступінь освіти. Для взаємодії з цими сутностями та збереження даних про них були створені функції бази даних, які включають доступні операції, такі як додавання, редагування та видалення записів про ці сутності з бази даних.

Розділ 3 Програмна реалізація соціальної інтелектуальної вебсистеми для знайомств

3.1 Структура модулів інформаційної системи і їх взаємозв'язок

Для розробки модулів соціальної інтелектуальної вебсистеми для знайомств з застосуванням способу оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств було створено два основних системи: підсистема роботи із базою знань та підсистема роботи із веб сайтом (рисунок 3.1).

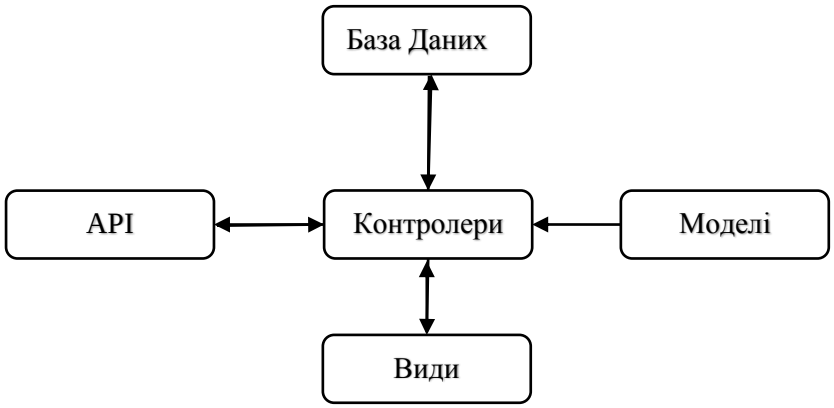


Рисунок 3.1 – Схема залежності систем

Підсистема роботи із базою знань представляє собою API (Application Programming Interface) для того щоб порівнювати фрейми. Та надавати розгорнуту відповідь для Веб сайту. Це додає цьому методу кросплатформленості, адже запити до API можна виконувати за наявності інтернету. У подальшому цей метод модна імплементувати у додатки на Android чи IOS.

Підсистема роботи із веб сайтом – це підсистема, яка містить сайт знайомств та усі дані для нього. Технологія, яка застосовувалась при розробці веб сайту це MVC (Model-View-Controller). Основна логіка веб сайту це відображення результатів які надіслав модуль знань, та редагування допоміжних даних для бази знань.

У модулі Веб сайту є наступні моделі:

– City, модель яка представляє собою сутність міста;

– Country, модель яка представляє собою сутність країни;

– Degree, модель яка представляє собою сутність ступеню освіти;

– Education, модель яка представляє собою місце здобування освіти;

– FrameForCompare, модель яка потрібна для формування фрейму-ідеалу, який у подальшому буде порівнюватись із фреймами у БД;

– FrameOutput, модель для виведення фреймів які повернула база знань;

– Gender, модель яка представляє сутність статі;

– Hobby, модель представлення сутності хобі;

– Income, модель представлення доходу;

– Personalitytype, модель представлення сутності типу особистості;

– Proffesion, модель представлення сутності професії;

– Userframe, модель представлення сутності анкети користувача;

– UserHobbies, модель представлення одного хобі користувача.

Для взаємодії із цими моделями даних були створені наступні контроллери:

– CitiesController;

– CountriesController;

– DegreesController;

– EducationsController;

– GendersController;

– HobbiesController;

– IncomesController;

– PersonalityTypesController;

– ProffesionsController;

– UserFramesController;

– UserHobbiesController.

У цих контролерах містяться функції для взаємодії із базою знань. URL адреси відповідають за різні дії над сутностями. Наприклад URL «https://localhost:7108/Cities/Create» виконує POST запит який створює новий запис у БД відносно введених значень.

Для кожної сутності було створено представлення для зручного керування інформацією із бази даних, тобро редагувати дані системи. Ці сутності мають назви:

– Create, представлення яке містить поля для створення нового запису у базі даних;

– Delete, представлення що дозволяє видалити запис у БД;

– Details, детальний перегляд обраної сутності;

– Edit, редагування даних обраної сутності;

– Index, представлення усіх записів про певну сутність.

Для реєстрації та авторизації створено представлення Login та Register. Ці два представлення створюють та авторизують користувачів.

Для підсистеми роботи із соціальною інтелектуальною вебсистемою для знайомств, було створено API яке має у собі один контролер який містить функції для формування експертного висновку та порівняння анкет. Цей модуль взаємодіє сутностями які описані для веб сайту.

3.2 Особливості реалізації соціальної інтелектуальної вебсистеми для знайомств

Для реалізації підсистеми для роботи із базою знань використовувались CRUD та REST технології.

CRUD [37] – це абревіатура, яка означає «створення» (Create), «читання» (Read), «оновлення» (Update) та «видалення» (Delete). Це основні операції, які використовуються для управління даними в базі даних.

– створення (Create) – дозволяє створювати нові записи або об'єкти в базі даних.

– читання (Read) – дозволяє отримувати інформацію з бази даних або системи управління контентом.

– оновлення (Update) – дозволяє змінювати існуючі записи або об'єкти в базі даних.

– видалення (Delete) – дозволяє видаляти записи або об'єкти з бази даних.

Тобто CRUD надає можливість користувачам взаємодіяти із даними шляхом створення, редагування, додавання та видалення інформації.

REST (Representational State Transfer) – це архітектурний стиль для розробки мережевих додатків, який використовується для взаємодії між клієнтами та серверами в розподіленій системі [38]. Основна ідея REST полягає в тому, щоб використовувати стандартні протоколи Інтернету, такі як HTTP, для забезпечення комунікації між клієнтом та сервером.

Для взаємодії із самою базою даних у додатку MVC потрібно створити моделі сутностей.

Entity Framework (EF) – це технологія доступу до даних в платформі .NET, розроблена компанією Microsoft [39]. Вона надає розробникам можливість взаємодіяти з базами даних, використовуючи об'єктно-орієнтовану модель даних. Entity Framework є потужним інструментом для роботи з базами даних в .NET-прикладах. Він дозволяє розробникам швидко і ефективно створювати, запитувати та оновлювати дані у базі даних, спрощуючи рутинну роботу з доступом до даних.

Для реалізації способу оцінювання сумісності анкет було розроблено наступний алгоритм. Система приймає модель фрейму ідеалу, після чого робить запит до БД на вибірку користувачів. Результат запиту записує у список UserFrames.Далі система ітерується по цьому списку та співставляє фрейми. Якщо критерій співпав, то система додає відсоток за один критерій у загальний відсоток співпадіння конкретної анкети із фреймом-ідеалом. Після перевірки усіх критеріїв система перевіряє відсоток співпадіння. Якщо цей відсоток більний за 50, то система додає цей фрейм у результуючий список, та формує експертний висновок для фрейму. У якому вказано критерії які співпали, та

відсоток співпадіння. У випадку якщо відсоток менший ніж 50 наступає наступна ітерація циклу. Після завершення циклу, система сортує анкети за відсотком співпадіння та повертає 5 найкращих кандидатів. Блок схема алгоритму співставлення фреймів зображено на рисунку 3.2.

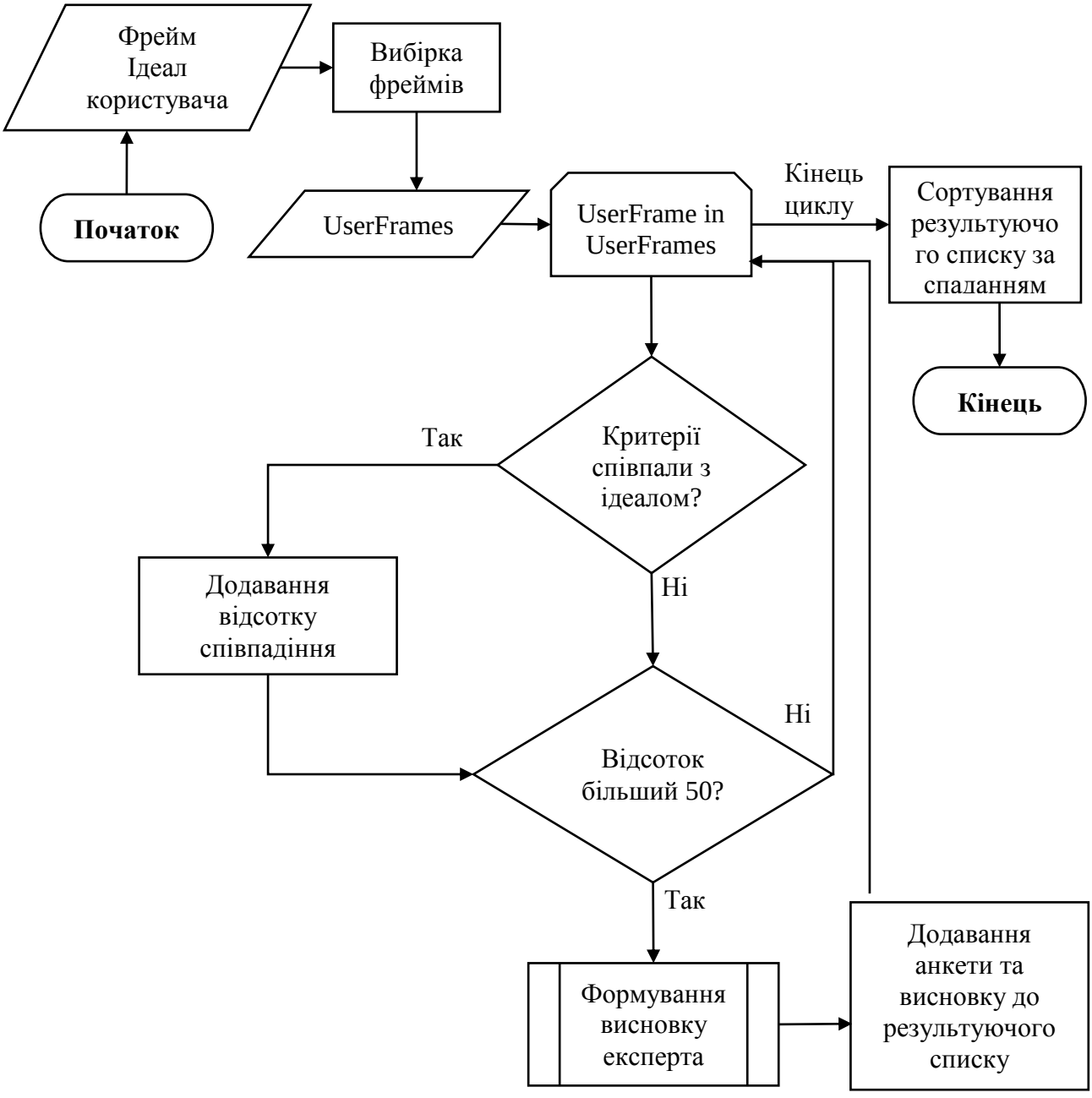


Рисунок 3.2 – Реалізація алгоритму способу сумісності анкет

Для перегляду запису про конкретне місто потрібно у контролері створити метод який буде приймати Id запису. При початку виконання методу система перевіряє чи є записи про міста у БД. Якщо записів у БД немає то

система повертає помилку 404 Не знайдено. При наявності записів про міста у БД система перевіряє чи існує запис із відповідним Id. Якщо запису не існує то система повертає помилку 404 не знайдено. У іншому випадку Система записує результат у змінну та повертає вигляд, вхідними даними якого і є запис про місто. Блок схема розробленого алгоритму виглядає наступним чином (рисунок 3.3).

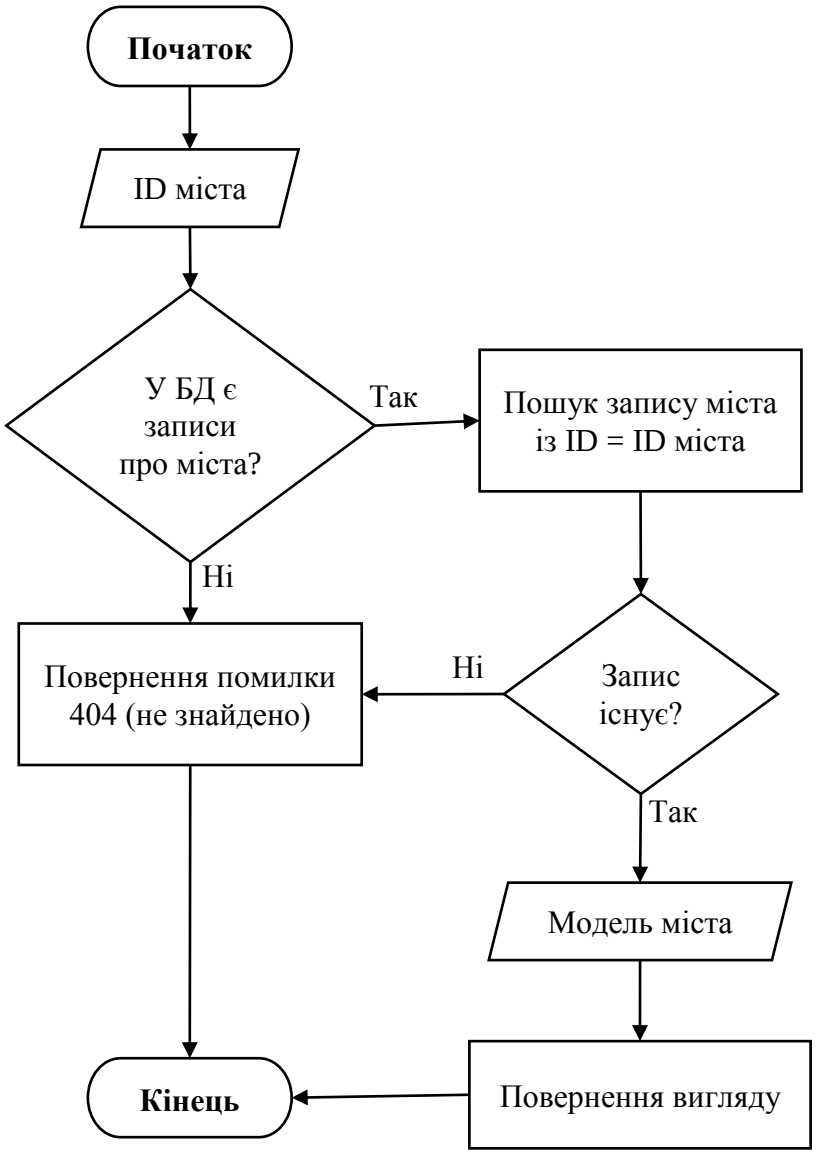


Рисунок 3.3 – Реалізація повернення запису конкретного міста

Для створення запису про сутність міста потрібно у відповідному контролері створити метод. Цей метод повинен приймати об'єкт моделі. Якщо мадель пройшла валідацію, то додати запис про нове місто у таблицю міст та

повернення вигляду із доданим записом. У іншому випадку, повернення помилки валідації. Блок схема алгоритму створення запису про сутність міста виглядає наступним чином (Рисунок 3.4).

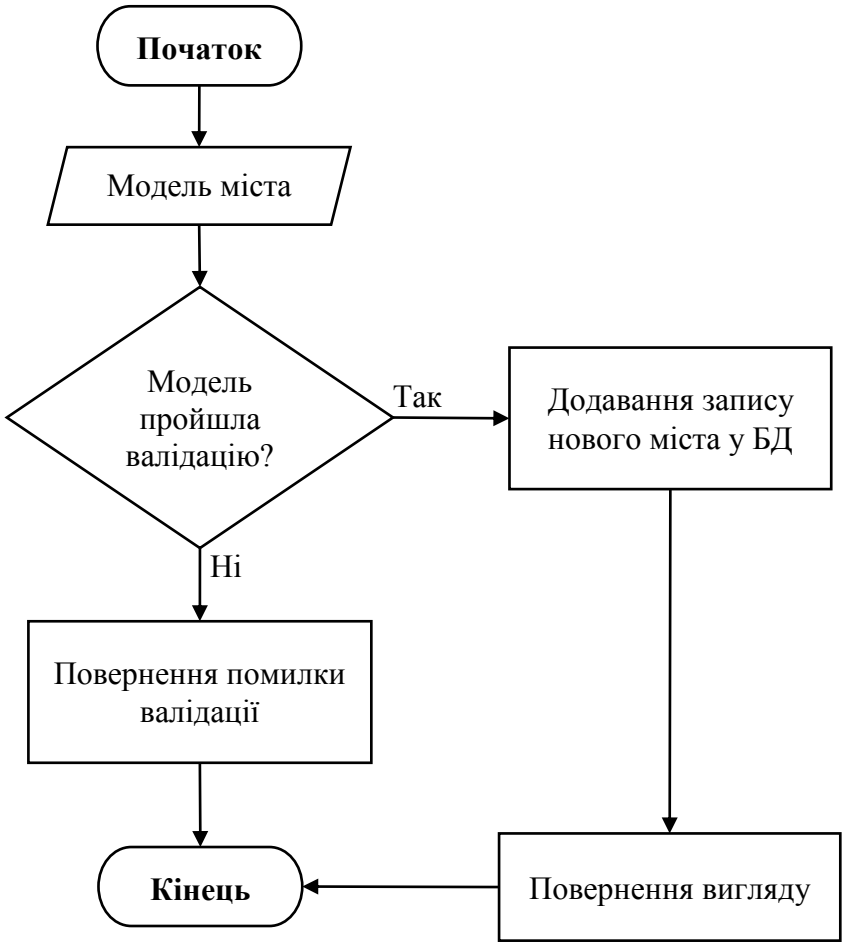


Рисунок 3.4 – Реалізація додавання запису нового міста

Для представлення усіх записів про міста потрібно у контролері створити відповідний метод, при виконанні цього методу контролер повинен зробити запис до БД, записати результат у список, та повернути вигляд вхідними даними якого буде список міст із БД. Блок схема розробленого алгоритму виглядає наступним чином (Рисунок 3.5).

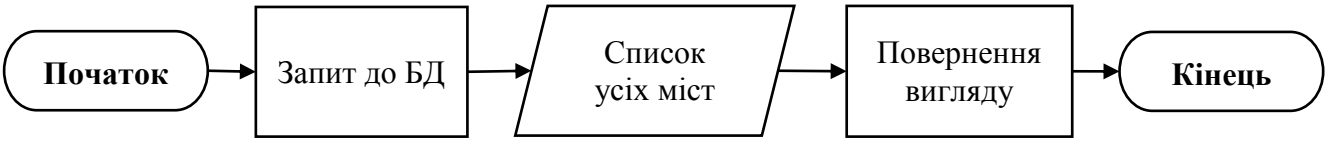


Рисунок 3.5 – Реалізація повернення списку записів усіх міст

Для видалення запису про конкретного міста потрібно у контролері створити метод який буде приймати Id міста. Для початку система перевіряє чи є у БД записи про міста, та чи існує запис із відповідним Id міста. Якщо у базі немає запису про місто або у БД немає записів про міста, повертається помилка 404 не знайдено. У іншому випадку система видаляє запис про міста із БД. Блок схема розробленого алгоритму виглядає наступним чином (Рисунок 3.6).

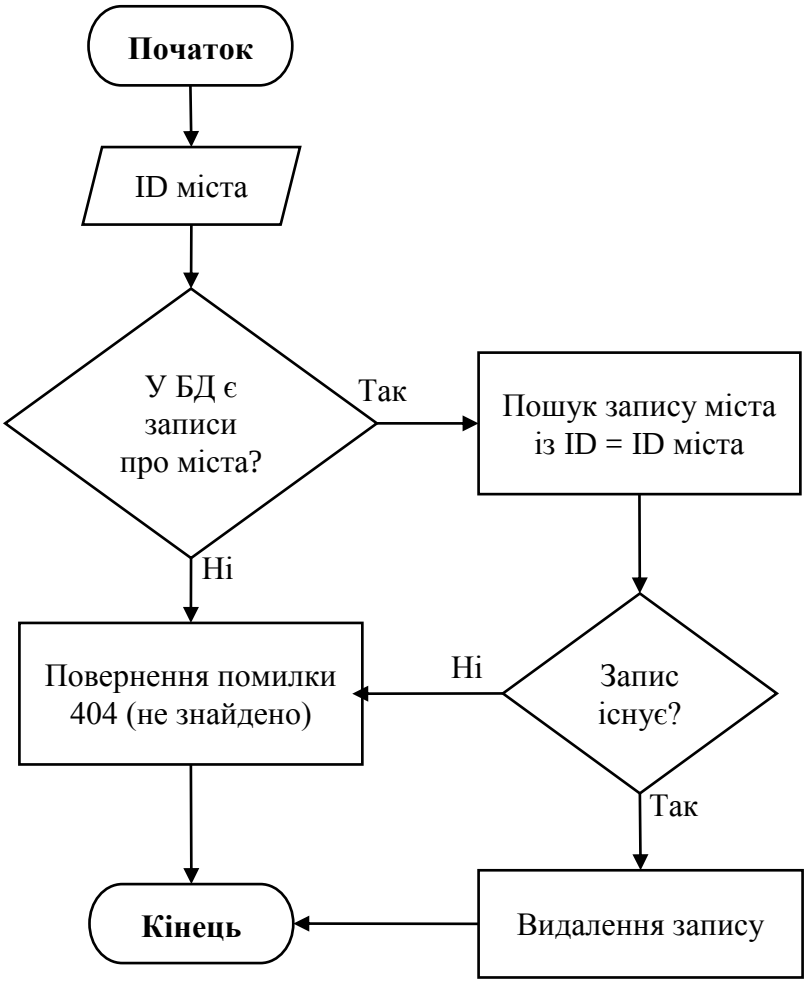


Рисунок 3.6 – Реалізація видалення запису про конкретне місто

Для редагування запису про конкретне місто у контролері потрібно створити метод, який буде приймати два параметри. Перший параметр це Id міста, другий параметр модель міста. Система перевіряє чи є записи про муста у БД, та чи є у ній запис із Id міста. Якщо запис існує то заміна стану моделі на вхідну . У іншому випадку повернення помилки 404 не знайдено. Блок схема

алгоритму редагування даних про конкретний запис міста виглядає наступним чином. (Рисунок 3.7).

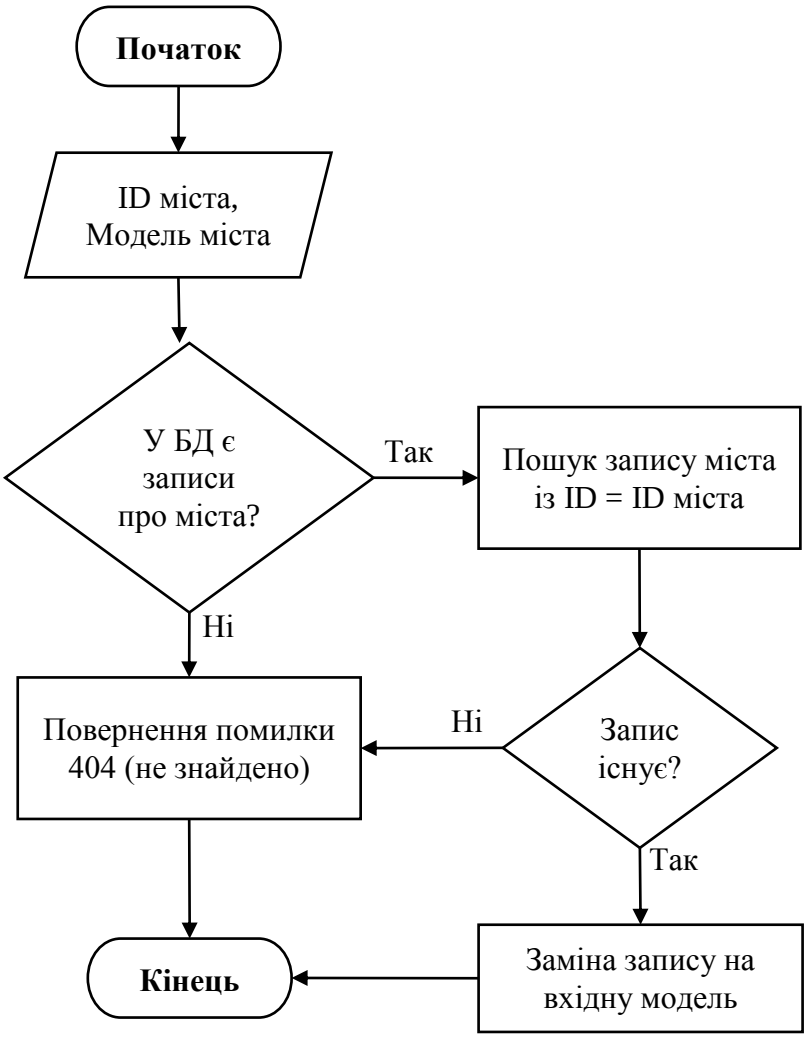


Рисунок 3.7 – Реалізація редагування запису про конкретне місто

Отже, було розроблено різні блок схеми алгоритмів модулів системи. Та описані основні технології які були використані при реалізації соціальної інтелектуальної системи.

3.3 Опис функціональних можливостей інформаційної системи

3.3.1 Опис загальної функціональності системи

У інформаційній системі є два типи користувачів, Адміністратор та Користувач.

У користувача є можливість користуватись наступним функціоналом.

– реєстрація, авторизація у системі;

– створення та редагування власного фрейму;

– перегляд списку фреймів;

– формування фрейму ідеалу, та перегляд списку результату порівняння;

– перегляд певного фрейму.

У адміністратора інформаційної системи функціонал більш розширений. У адміністраторів веб сайту знайомств доступний наступні групи функцій.

– додавання, редагування, видалення та перегляд даних про країни;

– додавання, редагування, видалення та перегляд даних про міста;

– додавання, редагування, видалення та перегляд даних про хобі;

– додавання, редагування, видалення та перегляд даних про типи особистості;

– додавання, редагування, видалення та перегляд даних про статі;

– додавання, редагування, видалення та перегляд даних про доходи;

– редагування, видалення та перегляд фреймів користувачів.

У цьому розділі описано основні функції інтелектуальної системи сайту знайомств, також описані типи користувачів, та поділ функціоналу між цими користвачами.

3.3.2 Опис функціональних можливостей користувача сайту знайомств

Для того щоб зареєструватись як користувач потрібно натиснути «Реєстрація», яка розташована у верхньому правому кутку сторінки у навігаційній панелі користувачів (Рисунок 3.8).

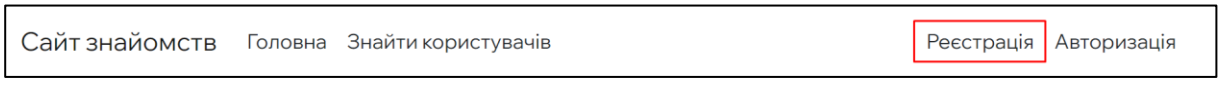


Рисунок 3.8 – Вигляд кнопки «Реєстрація»

Після натиснення кнопки реєстрації користувача перенаправить на сторінку із формою реєстрації, яка виглядає як зображено на рисуноку 3.9.

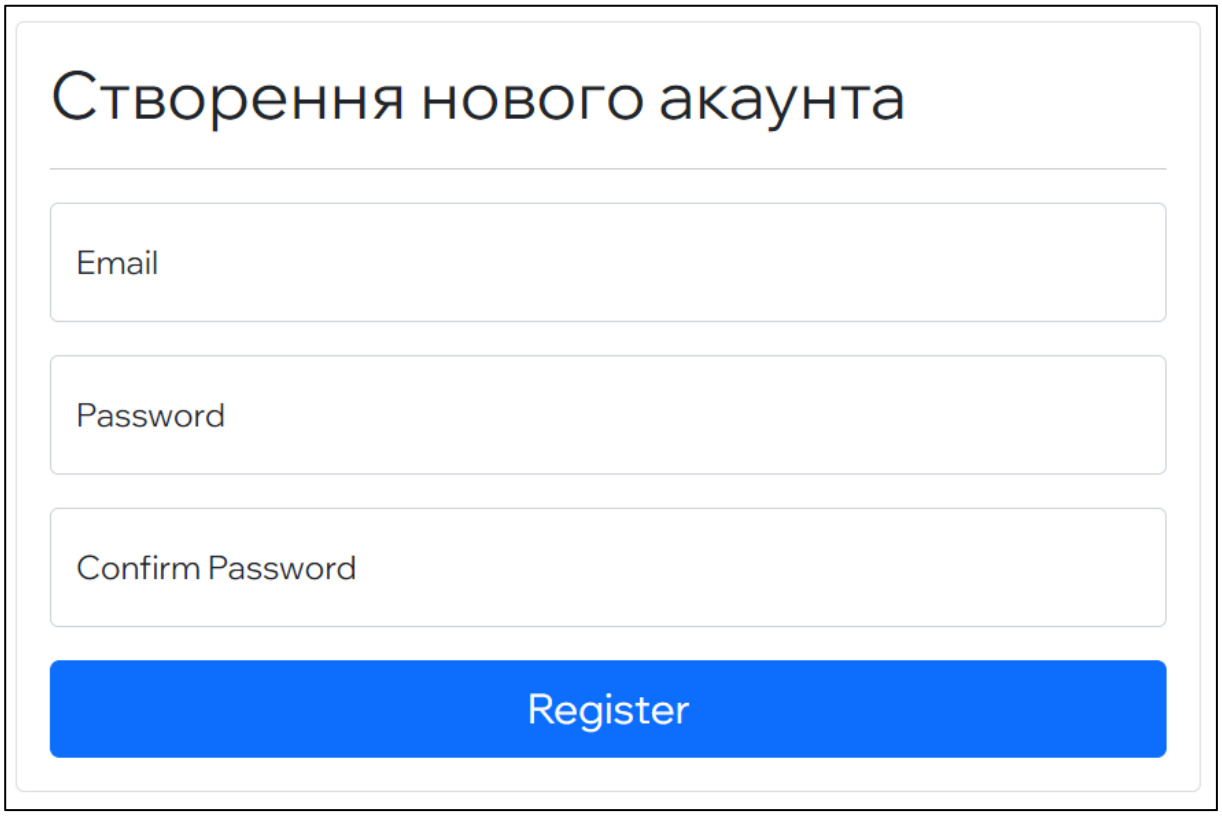


Рисунок 3.9 – Вигляд форми для реєстрації

Після заповнення відповідних полів форми користувача автоматично авторизує у системі. Та перенаправить на головну сторінку веб сайту.

Для того щоб створити власну анкету користувача, потрібно зайти у профіль акаунту. Для цього на панелі навігації натиснути на кнопку із написом Елетронної пошти яка була вказана при реєстрації, вигляд кнопки реєстрації зображено на рисунку 3.10

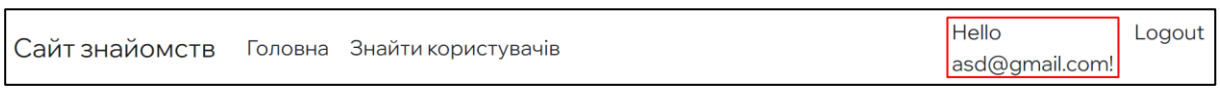


Рисунок 3.10 – Вигляд кнопки для реєстрації

Після натиснення на цю кнопку користувачеві поверне сторінку із редагуванням даних власного профілю (Рисунок 3.11).

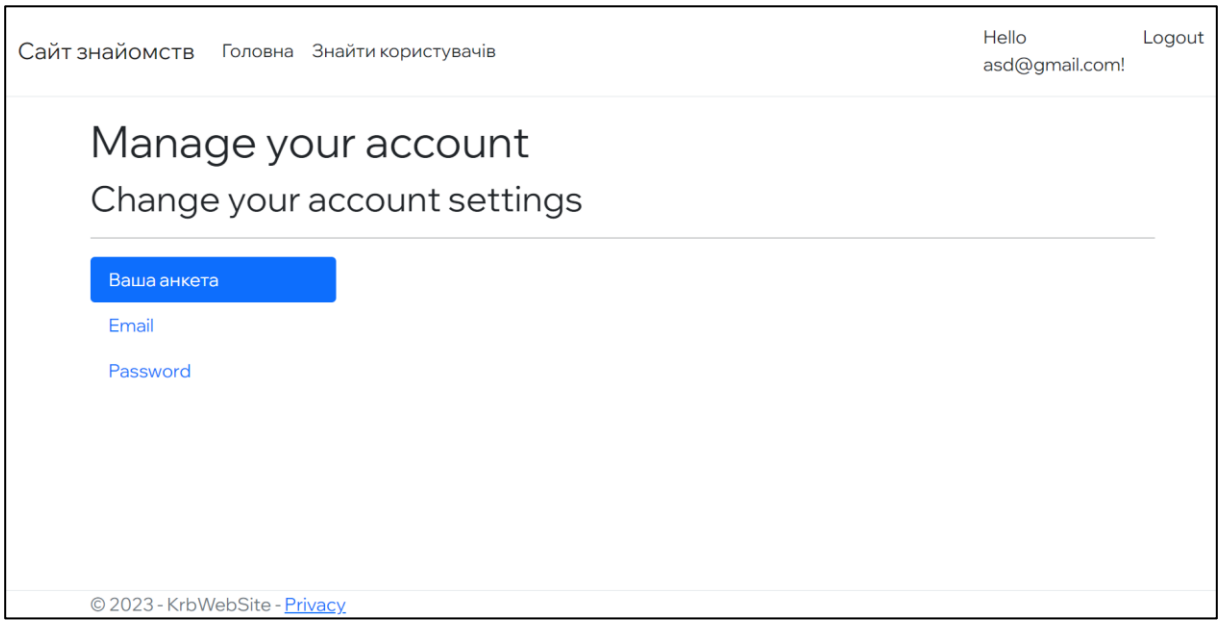


Рисунок 3.11 – Вигляд сторінки власного профілю

Для того щоб потрапити на сторінку із власною анкетою потрібно натиснути на кнопку «Ваша анкета», і заповнити відповідну форму яка зображена на рисунку 3.12.

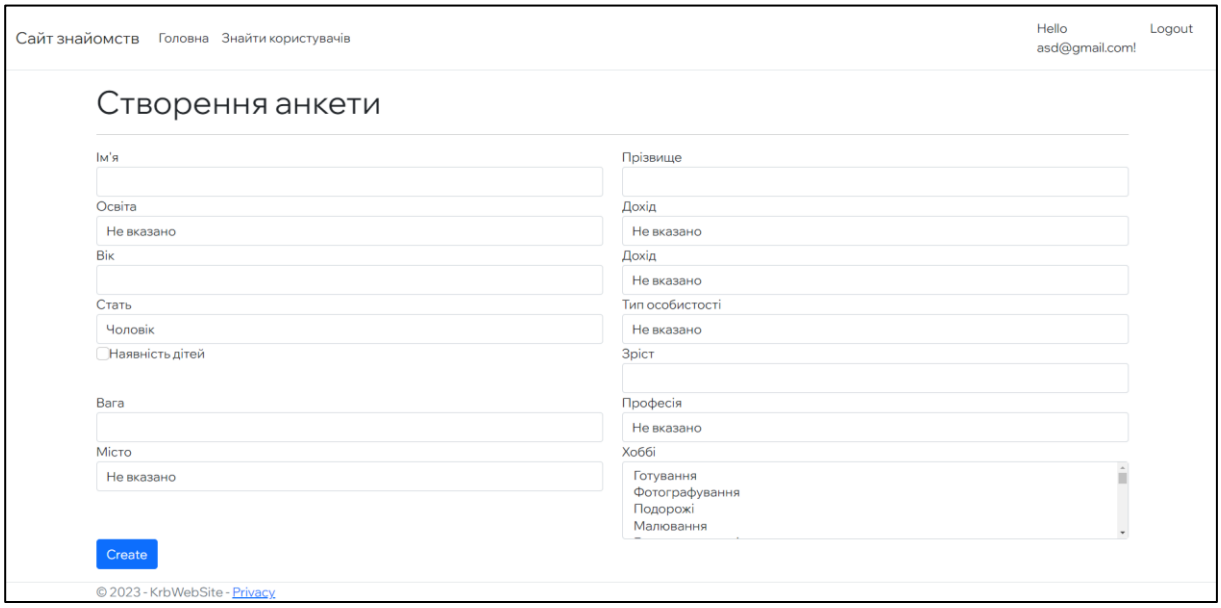


Рисунок 3.12 – Вигляд сторінки із формою для створення анкети

Для перегляду усіх можливих анкет потрібно на навігаційній панелі натиснути кнопку «Знайти користувачів», система поверне список анкет (Рисунок 3.13).

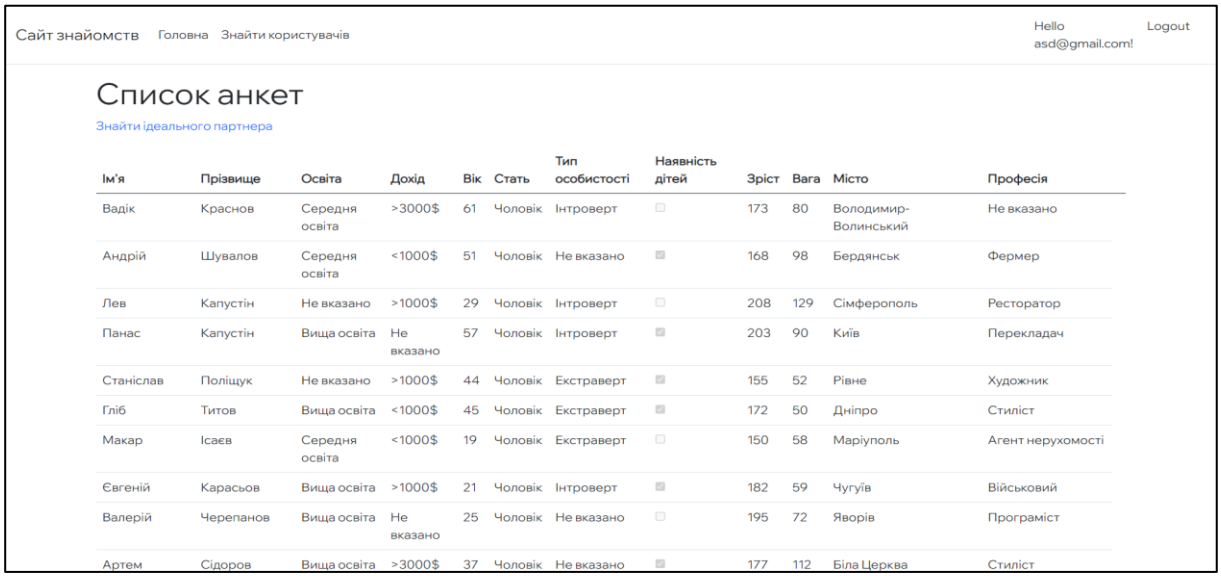


Рисунок 3.13 – Вигляд списку анкет

Для того щоб система порівняла користувацький «Ідеал», потрібно натиснути «знайти ідеального партнера», та заповнити відповідну форму (Рисунок 3.14)

Рисунок 3.14 – Форма для пошуку ідеального кандидату

У цьому розділі детально описано інструкцію із користування інтелектуальною системою сайту знайомств для ролі користувача. Описано функціональну можливість користувача, та покрокові дії для використання користувацького функціоналу.

3.3.3 Опис функціональних можливостей адміністратора сайту знайомств.

У адміністратора веб сайту знайомств, є багато однотипних дій, які можна описати одним алгоритмом.

Для редагування допоміжних даних потрібно на панелі навігації відповідну відповідну знопку із назвою категорії. Вигляд навігаційної панелі адміністратора веб сайту зображено на рисунку 3.15.

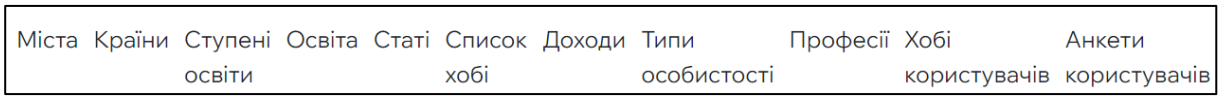


Рисунок 3.15 – Вигляд навігаційної панелі адміністратора веб сайту

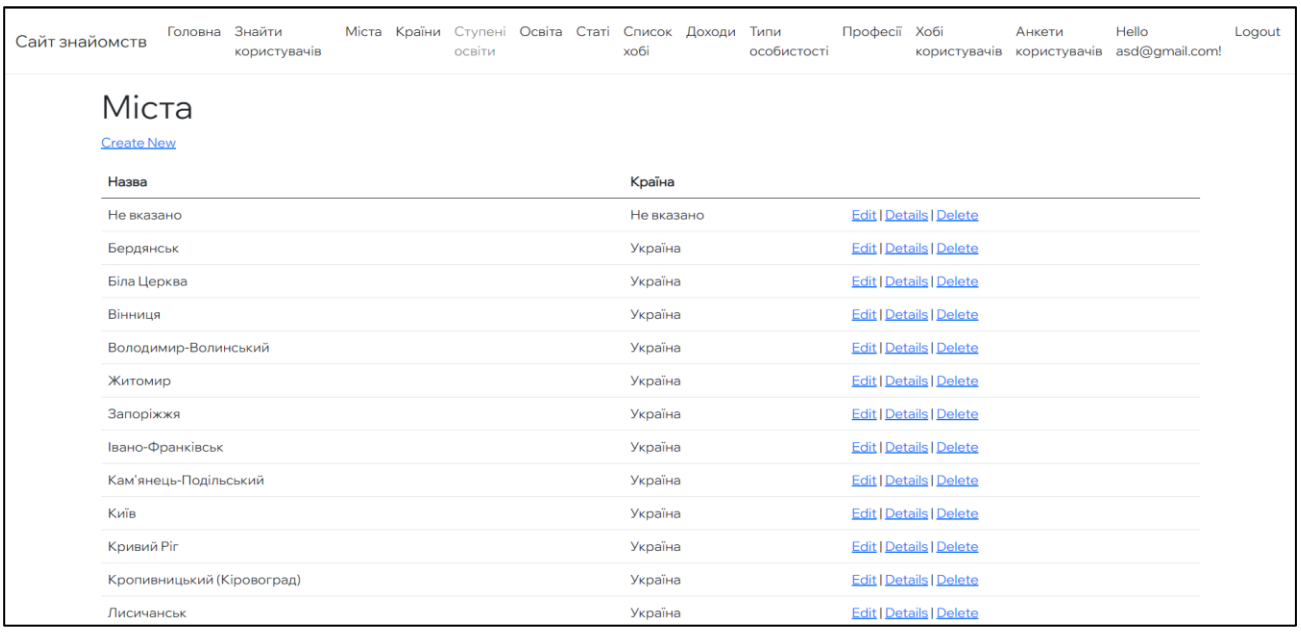


Рисунок 3.16 – Вигляд списку усіх міст

Для прикладу загального алгоритму дій буде описано додавання, редагування, та видалення записів про допоміжні дані на прикладі міста. Для цього потрібно натиснути кнопку «Міста», система перенаправить користувача на список усіх міст (Рисунок 3.16).

Для додавання нового міста потрібно натиснути кнопку «Create New» (Рисунок 3.17).

Рисунок 3.17 – Вигляд кнопки Create New

Після натиснення на кнопку, користувача перенаправить на стронку із формою для додавання нового міста (Рисунок 3.18).

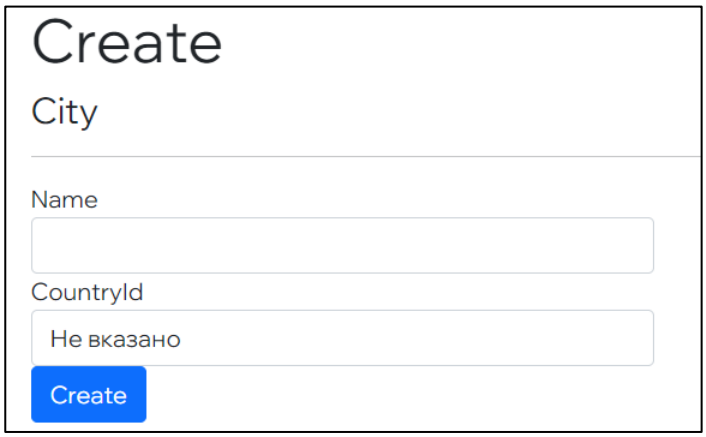


Рисунок 3.18 – Вигляд форми для створення нового міста.

Для того щоб редагувати дані про певний запис про місто потрібно у відповідному рядку таблиці натиснути кнопку «Edit» приклад рядка зображено на рисунку 3.19

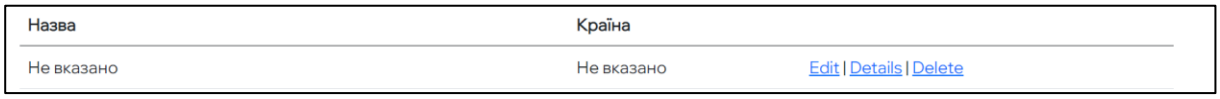


Рисунок 3.19 – Приклад табличного представлення запису міста

Далі потрібно заповнити форму вигляд якої показано на рисунку 3.20.

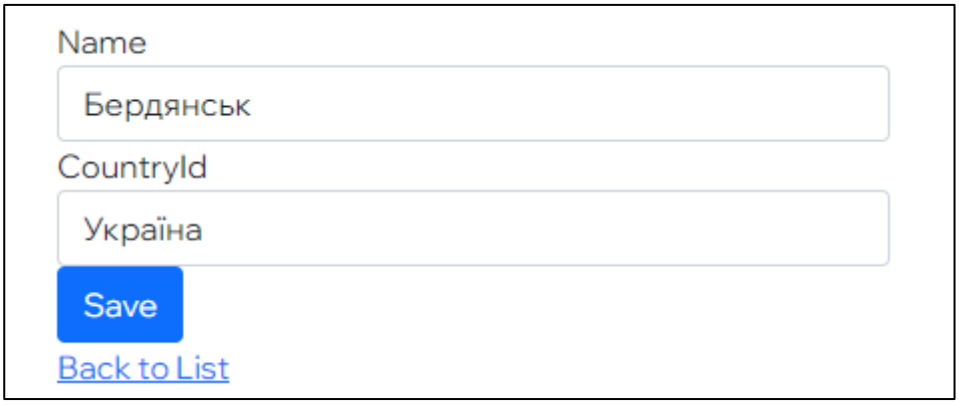


Рисунок 3.20 – Вигляд форми редагування обраного запису міста

Для видалення даних про певний запис потрібно у табличному представленні натиснути кнопку «Delete», після чого підтвердити видалення запису на сторінці на яку адміністратору поверне система (Рисунок 3.21).

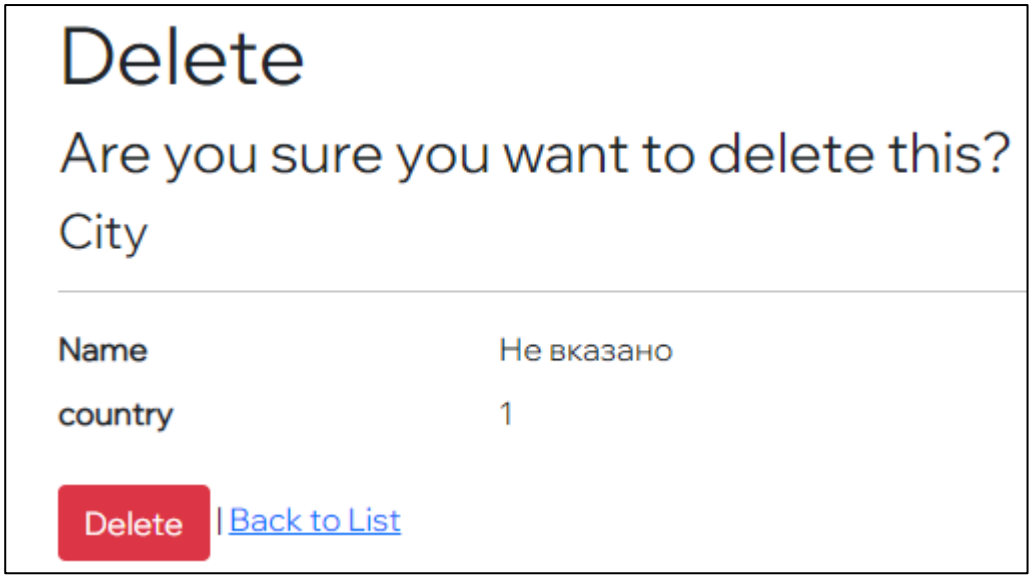


Рисунок 3.21 – Вигляд форми підтвердження видалення обраного запису міста

У цьому розділі було детально розписано інструкцію із користування інтелектуальною системою сайту знайомств для ролі адміністратора. Описано дії для вмкористання функціональних можливостей адміністратора.

3.3.4 Тестування соціальної інтелектуальної системи на прикладі сайту знайомств

Для тестування функціональності соціальної інтелектуальної системи на прикладі сайту знайомств розроблено перелік тест-кейсі для визначення потенційних вразливих місці системи.

Таблиця 3.1 – Шапка тест кейсу

Author: Жук Д.І	**Spec id** **1**	**Priority** **1**	**Producer** Жук Д.І	**Developer** Жук Д.І
OVERVIEW: Даний тест-комплект перевіряє виконання функцій web сайту знайомств.				
GLOBAL SETUP and ADDITIONAL INFO: Сайт «https://localhost:7108/»				

Для початку потрібно протестувати реєстрацію на веб сайті знайомств, для цього створено тест кейс №1 (Таблиця 3.2), результат якого зображено на рисунку 3.22.

Таблиця 3.2 – Тест-кейс №1

TC ID/Priority	**Реєстрація у системі у якості користувача.**	№1
Additional info: Сайт «https://localhost:7108/Identity/Account/Register» Логін: test1@gmail.com Пароль: 123Qwe!		
Revision History		
Created on: 22/05/2023 by Д. І. Жук	Тест-кейс №1	
Execution part		
PROCEDURE	**EXPECTED RESULT**	

Продовження таблиці 3.2

1. У текстове поле із написом «Email» вводимо логін «test1@gmail.com». 2. У текстове поле із написом «Password» вводимо пароль «123Qwe!». 3. У текстове поле із написом «Confirm Password» вводимо пароль «123Qwe!». 4. Натискаємо кнопку «Register»	Реєстрація повинна пройти успішно та у панелі навігації повинні з'явитись кнопки Logout та кнопка профілю із електронною скринькою «Hello test1@gmail.com».

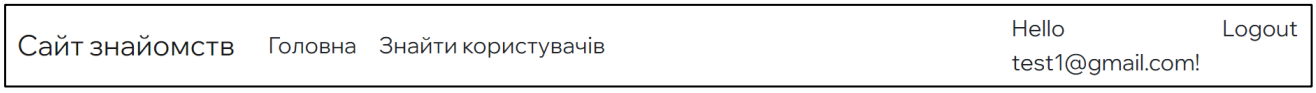


Рисунок 3.22 – Результат виконання тест кейсу №1

Отже, реєстрація у системі працює коректно. Тепер можна перевірити функцію формування фрейму користувача. Для цього розроблений тест кейс №2 (Таблиця 3.3), результат якого зображено на рисунку 3.23.

Таблиця 3.3 – Тест-кейс №2

TC ID/Priority	**Додавання фрейму користувача**	№2
Additional info: Сайт «https://localhost:7108/Identity/Account/Register» **Дані для форми:** Ім'я: Степан, Прізвище: Кожум'яка, Освіта: Середня, Дохід:<1000$, вік:25, місто: Київ, Стать: Чоловік, Тип особистості: Екстраверт, Зріст: 178, Вага 80, Професія: Вчитель, Хобі: Готування, фотографування, подорожування.		
Revision History		
Created on: 22/05/2023 by Д. І. Жук	Тест-кейс №2	
Execution part		
PROCEDURE	**EXPECTED RESULT**	

Продовження таблиці 3.3

1. На панелі навігації натиснути кнопку із написом «Hello test1@gmail.com». 2. Вибрати пункт «Ваша анкета». 3. Заповнити форму створення анкети відповідно до вхідних даних для форми. Натиснути кнопку «Create»	Створення анкети користувача із вказаними даними.

Рисунок 3.23 – Результат виконання тест кейсу №2

Отже, функція формування власних анкет користувачів процює коректно.

Далі потрібно перевірити права доступів до ресурсів. Користувач не має доступу до будь яких дій із допоміжними даними, навіть якшо він знає URL посилання на представлення цих даних. Для тестування прав доступу розроблено тест кейс №3 (Таблиця 3.4), результат якого зображено на рисунку 3.24.

Таблиця 3.4 – Тест кейс №3

TC ID/Priority	**Функція додавання нового запису**	№3
Idea: Тестування прав доступу **Additional info:** Сайт «https://localhost:7108/» Дані для входу систему у ролі користувача. Логін: test1@gmail.com Пароль: 123Qwe! **Ресурс доступ до якого у користувача відсутній** **URL:** «https://localhost:7108/Cities»		
Revision History		
Created on: 22/05/2023 by Д. І. Жук	Тест-кейс №3	
Execution part		
PROCEDURE	**EXPECTED RESULT**	

Продовження таблиці 3.4

1. Натиснути на кнопку «Авторизація» 2. У відповідні поля форми ввести дані для входу у систему. 3. Натиснути кнопку «Log In» 4. У адресний рядок браузера потрібно ввести URL посилання ресурсу доступу до якого у простого користувача немає.	Веб сайт повинен сповістити користувача написом «Access denied You do not have access to this resource.»

Рисунок 3.24 – Результат виконання тест кейсу №3

Отже, допоміжні дані системи не зможуть редагувати користувачі веб сайту знайомств, якшо у них немає ролі адміністратору сайту.

Для тестування редагування допоміжних даних системи розроблено наступний тест кейс (Таблиця 3.5), результат якого зображено на рисунку 3.25.

Таблиця 3.5 – Тест кейс №4

TC ID/Priority	**Тестування прав доступу до ресурсу**	№4
Additional info: Сайт «https://localhost:7108/» Дані для входу систему у ролі адміністратора. Логін: asd@gmail.com Пароль: 123Qwe! **URL:** «https://localhost:7108/Cities»		
Revision History		
Created on: 22/05/2023 by Д. І. Жук	Тест-кейс №1	
Execution part		
PROCEDURE	**EXPECTED RESULT**	

Продовження таблиці 3.5

1. Натиснути на кнопку «Авторизація» 2. У відповідні поля форми ввести дані для входу у систему. 3. Натиснути кнопку «Log In» 4. У адресний рядок браузера потрібно ввести URL. 5. У рядку із Назвою міста «Вінниця», натиснути кнопку «Edit». 6. У поле форми «Назва» ввести значення «Ялта» 7. Натиснути кнопку «Save».	У таблиці назва «Вінниця» повинна змінитись на значення «Ялта»

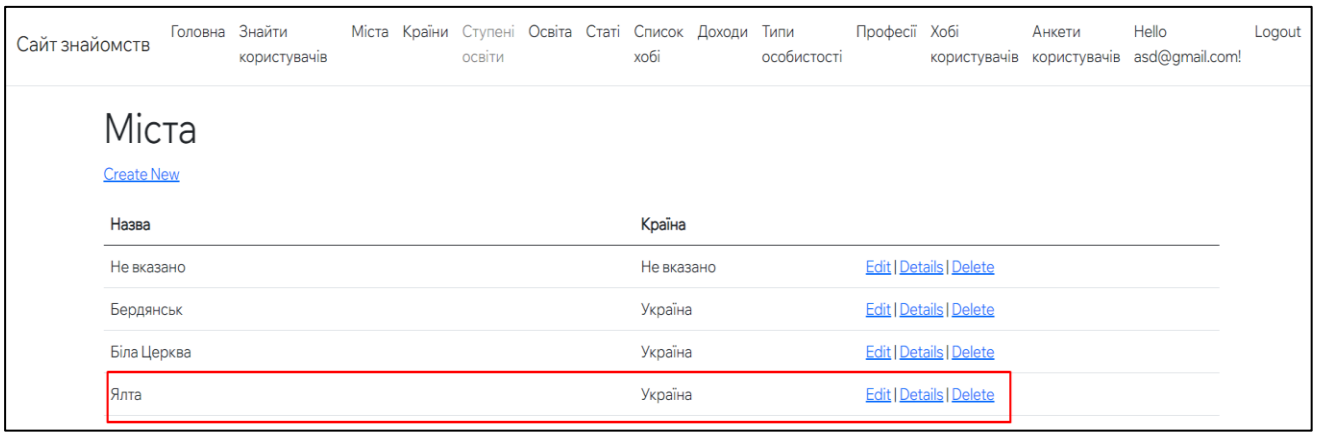


Рисунок 3.25 – Результат виконання тест кейсу №4

Отже, функція редагування даних про обране місто працює коректно. Таким чином, було розроблено тест комплект для тестування функціональності інтелектуальної веб системи. Система пройшла тестування успішно. Усі функції працюєть коректно.

3.4 Дослідження ефективності застосування способу оцінювання сумісності анкет користувачів за фреймовим поданням знань

Для дослідження ефективності способу порівняння сумісності анкет користувачів було створено 2 тестових фрейми-запити (Таблиці 3.6, 3.8).

Таблиця 3.6 – Запит №1

Фрейм №1	
Вік	20-40
Зріст	140-170
Освіта	2
Дохід	<1000$
Стать	Чоловік
Тип особистості	Екстраверт
Професія	Інвестиційний аналітик
Місто	Калуш
Хобі	В’язання

У відповідь додаток надіслав наступний перелік фреймів (Таблиця 3.7).

Таблиця 3.7 – 5 найкращих співпадінь із фреймом-ідеалом

Ім’я	**Прізвище**	**Вік**	**Зріст**	**Стать**	**Тип особистості**	**Дохід**	**Освіта**	**Місто**	**Хобі**	**Професія**	**%**
Євген	Єрмак	25	156	Чоловік	Екстраверт	<1000$	Середня освіта	Калуш	В’язання	Інвестиційний аналітик	100%
Вадім	Ігнат	26	158	Чоловік	Інтроверт	<1000$	Середня освіта	Бориспіль		Інвестиційний аналітик	66%
Лук'ян	Гаврилів	24	151	Чоловік	Екстраверт	Не вказано	Середня освіта	Кам'янка-Дніпровська		Інвестиційний аналітик	66%
Василь	Ільїн	38	161	Чоловік	Не вказано	<1000$	Середня освіта	Калуш		Менеджер	66%
Іван	Сідорів	40	152	Чоловік	Екстраверт	>3000$	Середня освіта	Коломия	В’язання	Фармацевт	66%

Аналізуючи дані таблиці, та початкові дані ціна одного критерія у відсотках 11%, а відсоток за співпадіння 1 хоббі 11%.

У першому випадку система дала співпадіння у 100%, усі критерії співпали. Другий фрейм «Вадім Ігнат», має деякі розбіжності: Тип особистості, Місто та Хобі. Третій фрейм «Лук'ян Гаврилів», не співпав по наступним

критеріям: Дохід, місто та хобі. Четвертий фрейм «Василь Ільїн», не співпав по наступним критеріям: Хобі, професія, тип особистості. Останній фрейм «Іван Сідорів», не співпав по наступним критеріям: Професія, дохід та місто.

Спосіб оцінювання сумісності анкет користувачів правильно зробив вибірку даних із бази знань. Для того щоб переконатись у дієвості цього методу було розроблено наступний фрейм-запит (Таблиця 3.8).

Таблиця 3.8 – Запит №2

Фрейм №2	
Вік	18-20
Зріст	140-150
Освіта	Вища освіта
Дохід	>3000$
Стать	Жінка
Тип особистості	Не вказано
Професія	Актор
Місто	Бердянськ
Хобі	Гра на музичних інструментах, В'язання, Комп'ютерні ігри, Гра у настільні ігри, Вивчення мов.

У відповідь додаток надіслав наступний перелік фреймів (Таблиця 3.9).

Таблиця 3.9 – Результат запиту №2

Ім’я	**Прізвище**	**Вік**	**Зріст**	**Стать**	**Тип особистості**	**Дохід**	**Освіта**	**Місто**	**Хобі**	**Професія**	**%**
Влада	Гребенюк	19	142	Жінка	Інтроверт	>3000$	Вища освіта	Луганськ	Вивчення мов	Рятувальник	57.41 %
Корнелія	Кравчук	57	142	Жінка	Інтроверт	>3000$	Вища освіта	Бердянськ	Немає співпадінь	Військова	55.56 %

Аналізуючи дані таблиці, та початкові дані ціна одного критерія 12.5%, тому що у вхідному фреймі не вказано тип особистості. Системо не буде брати до уваги цей критерій. Ціна одного хобі у відсотках буде рівна 2.5%.

Перша анкета користувачки , співпала по наступним критеріям: стать, вік, зріст, дохід, освіта. Також співпало одне хобі, це вивчення мов. Не співпали наступні критерії: професія та місто.

Друга анкета користувачки «Корнелія Кравчук», співпала по наступним критаріям: стать, зріст, дохід, місто, освіта. Не співпали наступні критерії: жодне із хобі, вік, професія.

Отже, для дослідження ефективності способу порівняння сумісності анкет користувачів було створено два тестові фрейми-запити. Перший фрейм-запит (Таблиця 3.5) містить характеристики користувача, а наступні фрейми-відповіді (Таблиці 3.6 і 3.8) містять потенційних користувачів, які найбільше відповідають заданим характеристикам.

У випадку першого фрейму-запиту (Таблиця 3.5), система знайшла одне повне співпадіння з фреймом ідеалом, де всі критерії співпали на 100%. У наступних відповідях (Таблиця 3.6), були знайдені фрейми, які мають деякі розбіжності зі стандартним фреймом, але все ще відповідають деяким критеріям.

У випадку другого фрейму-запиту (Таблиця 3.7), також були знайдені фрейми-відповіді (Таблиця 3.8), які відповідають заданим характеристикам користувачки. Проте, ні в одному з цих відповідних фреймів не повного співпадіння усіх критеріїв. Також у другому випадку система повернула тільки два фрейми, це свідчить про те що у згенерованих даних це виявилось фреймів із повним співпадінням. Це демонструє, що спосіб порівняння сумісності анкет користувачів виявився ефективним.

3.5 Висновки до розділу 3

Отже, у розділі було створено та описано структуру соціальної інтелектуальної вебсистеми для знайомств з застосуванням способу оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств. Розглянуті два типи користувачів системи: адміністратор та звичайний користувач. Для звичайного користувача були надані такі можливості, як реєстрація та авторизація, створення та редагування власного фрейму, перегляд списку фреймів, формування фрейму ідеалу та перегляд результатів порівняння, а також перегляд конкретного фрейму. Адміністратор системи, з свого боку, мав доступ до розширеного функціоналу, який включав додавання, редагування, видалення та перегляд даних про країни, міста, хобі, типи особистості, статі та доходи.

Також було розроблено дві інструкції для користування соціальною інтелектуальною вебсистемою для знайомств з застосуванням способу оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств.

Користувацька інструкція містить докладні вказівки для користувачів щодо використання функцій інформаційної системи. Інструкція надає крок за кроком пояснення щодо реєстрації, авторизації, створення фрейму, перегляду фреймів і формування фрейму ідеалу.

У Інструкції Адміністратора вебсайту надано докладний посібник для адміністраторів системи. Інструкція охоплює додавання, редагування, видалення та перегляд даних про країни, міста, хобі, типи особистості, статі та доходи, а також редагування, видалення та перегляд фреймів користувачів.

Для тестування системи було розроблено тест комплект, та проведено тестування, надано звіти про результати цього тестування.

Досліджено ефективність способу порівняння сумісності анкет за допомогою фреймового подання знань. Створено дві тестових анкети-ідеали для

порівняння анкет. У першому запиті було знайдено повне співпадіння з фреймом ідеалом. Інші анкети-відповіді які були повернуті програмою мають деякі розбіжності, але все ще відповідають критеріям. У другому запиті не знайдено жодної анкети із повним співпадінням. Це зв’язано їз тим що у тестовій вибірці не виявилось достатньої кількості анкет для того, щоб знайти хоча б одне сівпадіння з даним фреймом. Можна зробити висновок про те, що розроблений спосіб порівняння сумісності анкет за допомогою фреймового подання знань виявився ефективним.

Висновки

Кваліфікаційна робота бакалавра присвячена розробці способу оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств та відповідної інформаційної системи у вигляді соціальної інтелектуальної вебсистеми для знайомств, що використовує розроблений спосіб оцінювання сумісності анкет користувачів за фреймовим поданням знань

Під час виконання кваліфікаційної роботи бакалавра проаналізовано предметну область, проведено огляд та дослідження її особливостей. Також розглянуто застосування фреймового подання знань для задач класифікації та пошуку відповідності. Для подібних задач були проаналізовані існуючі рішення.

У результаті виконання кваліфікаційної роботи бакалавра досягнуто наступних цілей:

– розроблено спосіб оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств;

– розроблено модель фрейму для узагальненого подання анкет користувачів та критеріїв пошуку користувачів для порівняння;

– спроєктувано структуру інформаційної системи у вигляді соціальної інтелектуальної вебсистеми для знайомств, що використовує спосіб оцінювання сумісності анкет користувачів за фреймовим поданням знань;

– спроєктувано структуру бази даних для соціальної інтелектуальної вебсистеми для знайомств;

– виконано програмну реалізацію інформаційної системи у вигляді соціальної інтелектуальної вебсистеми для знайомств;

– проведено тестування розробленого способу оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств.

У подальшому є можливість розробки та покращення цієї системи: розробка версії мобільного додатку для знайомств та покращення методу порівняння анкет. Розроблений метод можна покращити наступним способом, додаванням подвійного порівняння анкет. Тобто при запиті «користувача 1» система також порівнювала і його анкету із фреймом-ідеалом кандидатів із високим відсотком співпадіння.

Перелік посилань

1. Голос України. Ретро-огляд: як знаходили одне одного наші батьки. URL:http://www.golos.com.ua/article/75203

2. Покупон. Сайти знайомств: всі «за» і «проти» відносин через інтернет URL: https://blog.pokupon.ua/sajty-znakomstv-vse-za-i-protiv-otnoshenij-cherez-internet/

3. Вікіпедія. Обліковий запис. URL: https://uk.wikipedia.org/wiki/Обліковий_запис

4. JuliaDates. URL:https://juliadates.com

5. JuliaDates. Профіль Вікторії URL: https://juliadates.com/profile/16820737

6. JuliaDates. Профіль Стефанії URL: https://juliadates.com/profile/18111680

7. Instagram. URL: https://www.instagram.com/

8. Reddit. URL: https://www.reddit.com/

9. Вікіпедія. Видобування знань. URL: https://uk.wikipedia.org/wiki/Видобування_знань

10. Вікіпедія. Антецедент. URL: https://uk.wikipedia.org/wiki/ Антецедент

11. Вікіпедія. Консеквент. URL: https://uk.wikipedia.org/wiki/Консеквент

12. Вікіпедія. Фасетна класифікація. URL:https://uk.wikipedia.org/wiki/Фасетна_класифікація

13. StudFiles. Теорія фреймів. Структура фрейму. Слоти та приєднані процедури. URL: https://studfile.net/preview/2582507/page:13/

14. StudFiles. Моделі подання знань. URL: https://studfile.net/preview/14532009/page:2

15. Всі на урок. Процедурні та декларативні знання: приклади і опис. URL: https://yrok.pp.ua/serednya-osvta/12031-procedurn-ta-deklarativn-znannya-prikladi-opis.html

16. StudFiles Моделі подання знань. URL: https://studfile.net/preview/14532009/page:2/

17. Бондаренко Ю. І. «Формування літературних знань та умінь школярів із допомогою фреймів». Психолого-педагогічні, 2022. С 48–60.

18. HDSR. Finding Love on a First Data: Matching Algorithms in Online Dating. URL: https://hdsr.mitpress.mit.edu/pub/i4eb4e8b/release/2

19. Obozrevatel. 7 найпопулярніших сайтів для романтичних знайомств в Україні. URL: https://news.obozrevatel.com/lady/sex/top-7-samyih-populyarnyih-portalov-dlya-romanticheskih-znakomstv-v-ukraine.htm

20. Tinder. URL: https://tinder.com/uk

21. Badoo. URL: https://badoo.com/uk/

22. Google Play. eDarling – For people looking. URL: https://play.google.com/store/apps/details?id=net.edarling.mobile&hl=uk&gl=US&pli=1

23. Jolly.me. URL: https://jolly.me/

24. JuliaDates. Профіль Марії URL: https://juliadates.com/profile/25127269.

25. JuliaDates. Профіль Ольги URL: https://juliadates.com/profile/23823475

26. Badoo. Профіль Дмитра URL: https://gew3.badoo.com/profile/0zAhMACjE4NzM0ODg2MjMAIB4hk4w2NtD_IQU6daKU4xYLe3fVbiLcUfo-d5sx6F03

27. Вікіпедія. Список українських чоловічих імен. URL: https://uk.wikipedia.org/wiki/Список_українських_чоловічих_імен

28. Вікіпедія. Список українських жіночих імен. URL: https://uk.wikipedia.org/wiki/Список_українських_жіночих_імен

29. Вікіпедія. Список найпоширеніших прізвищ в Україні URL: https://uk.wikipedia.org/wiki/Список_найпоширеніших_прізвищ_в_Україні

30. HobiTera. Список хобі. URL: https://hobitera.com/

31. Вікіпедія. Unified Modeling Language URL: https://uk.wikipedia.org/wiki/Unified_Modeling_Language

32. Evergreen. UML для бізнес-моделювання: для чого потрібні діаграми процесів. URL: https://evergreens.com.ua/ua/articles/uml-diagrams.html

33. StudFiles. Діаграма розгортання URL: https://studfile.net/preview/5010027/page:6/

34. Вікіпедія. Діаграма потоків даних. URL: https://uk.wikipedia.org/wiki/Діаграма_потоків_даних

35. Вікіпедія. C Sharp URL: https://uk.wikipedia.org/wiki/C_Sharp

36. Вікіпедія. ASP.NET URL: https://uk.wikipedia.org/wiki/ASP.NET

37. Вікіпедія. CRUD URL: https://uk.wikipedia.org/wiki/CRUD

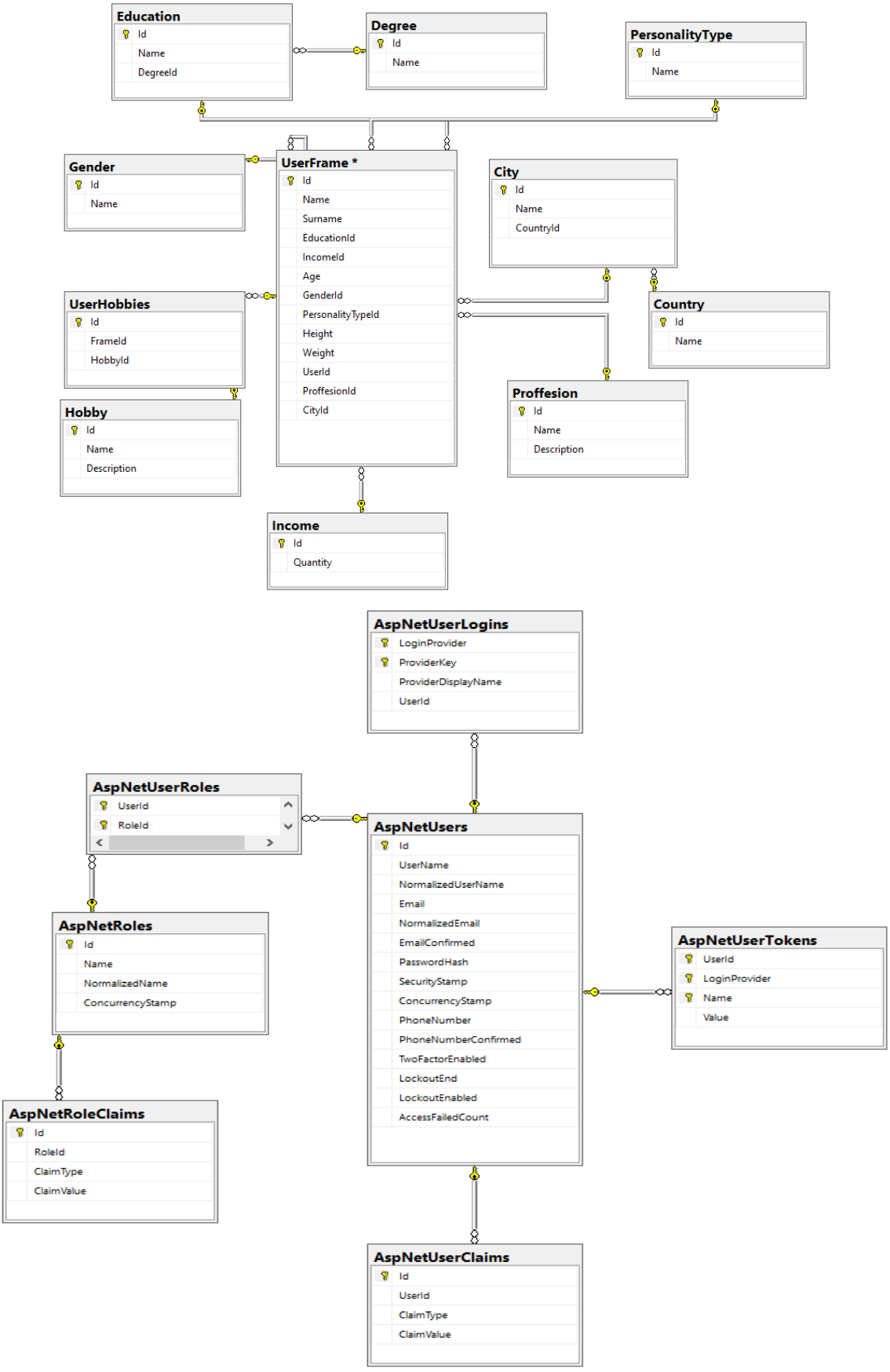
38. Codeguida. ІЩо таке RESTful API. URL: https://codeguida.com/post/601

39. Вікіпедія. Entity Framework URL: https://uk.wikipedia.org/wiki/Entity_Framework

ДОДАТКИ

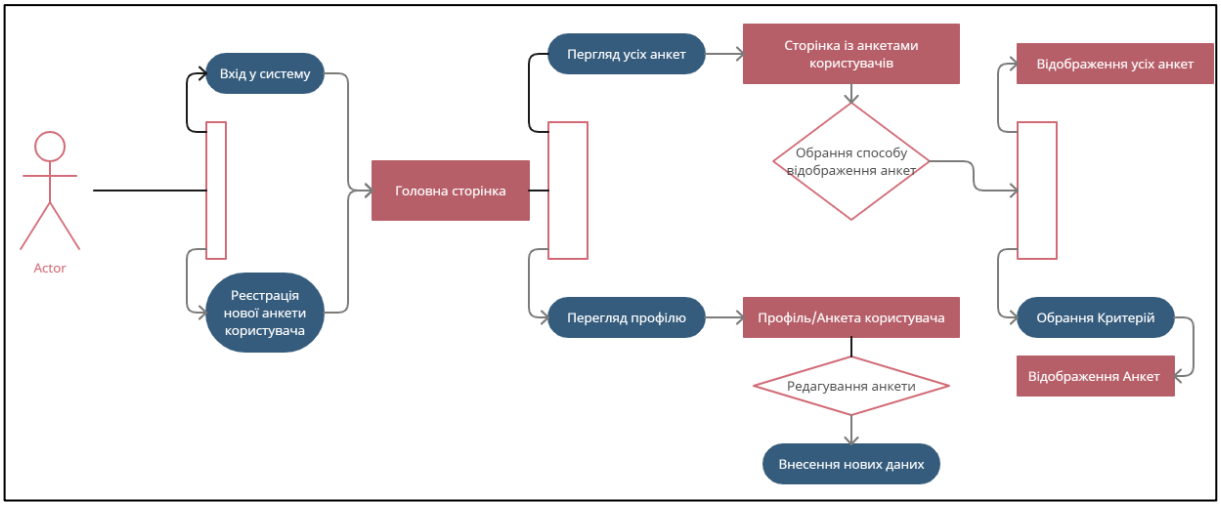
Додаток А

Даталогічні моделі БД



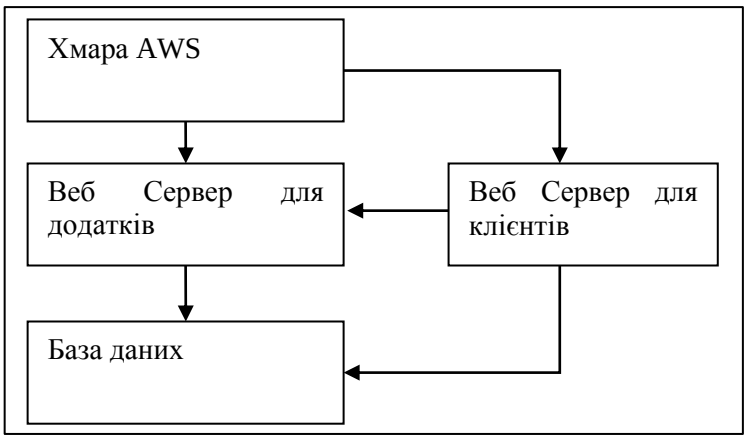
Додаток Б

UML Діаграма діаграма активності користувача



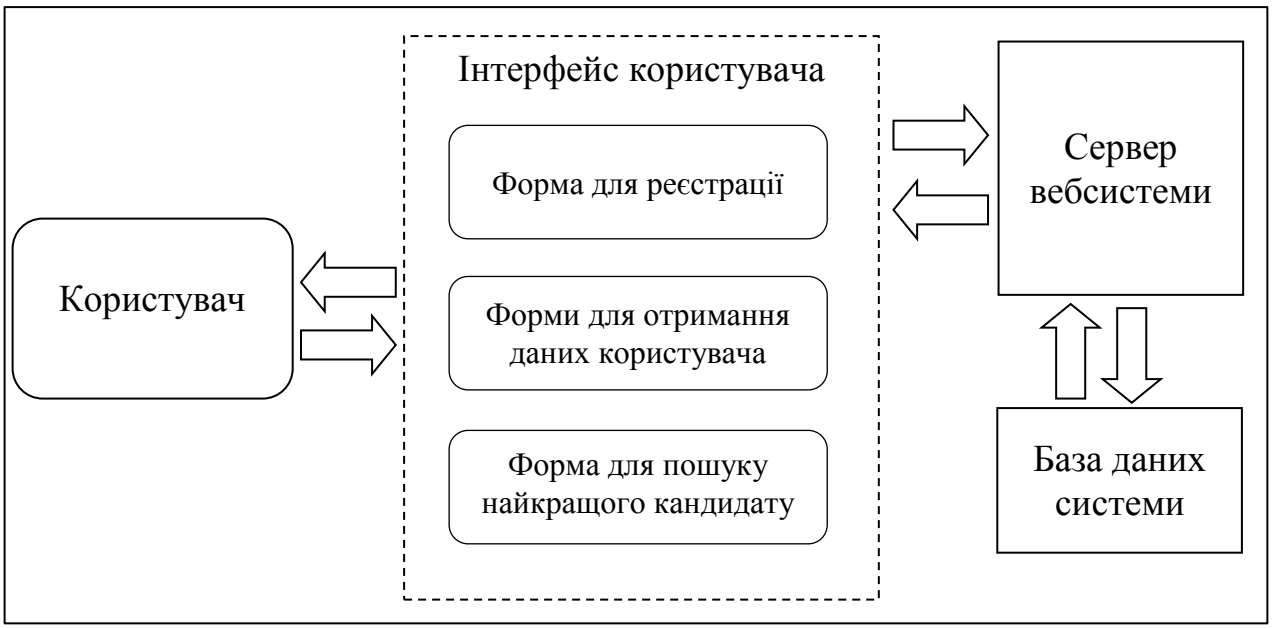
Додаток В

UML Діаграма розгортання системи на сервісі AWS



Додаток Г

UML Діаграма потоків даних системи

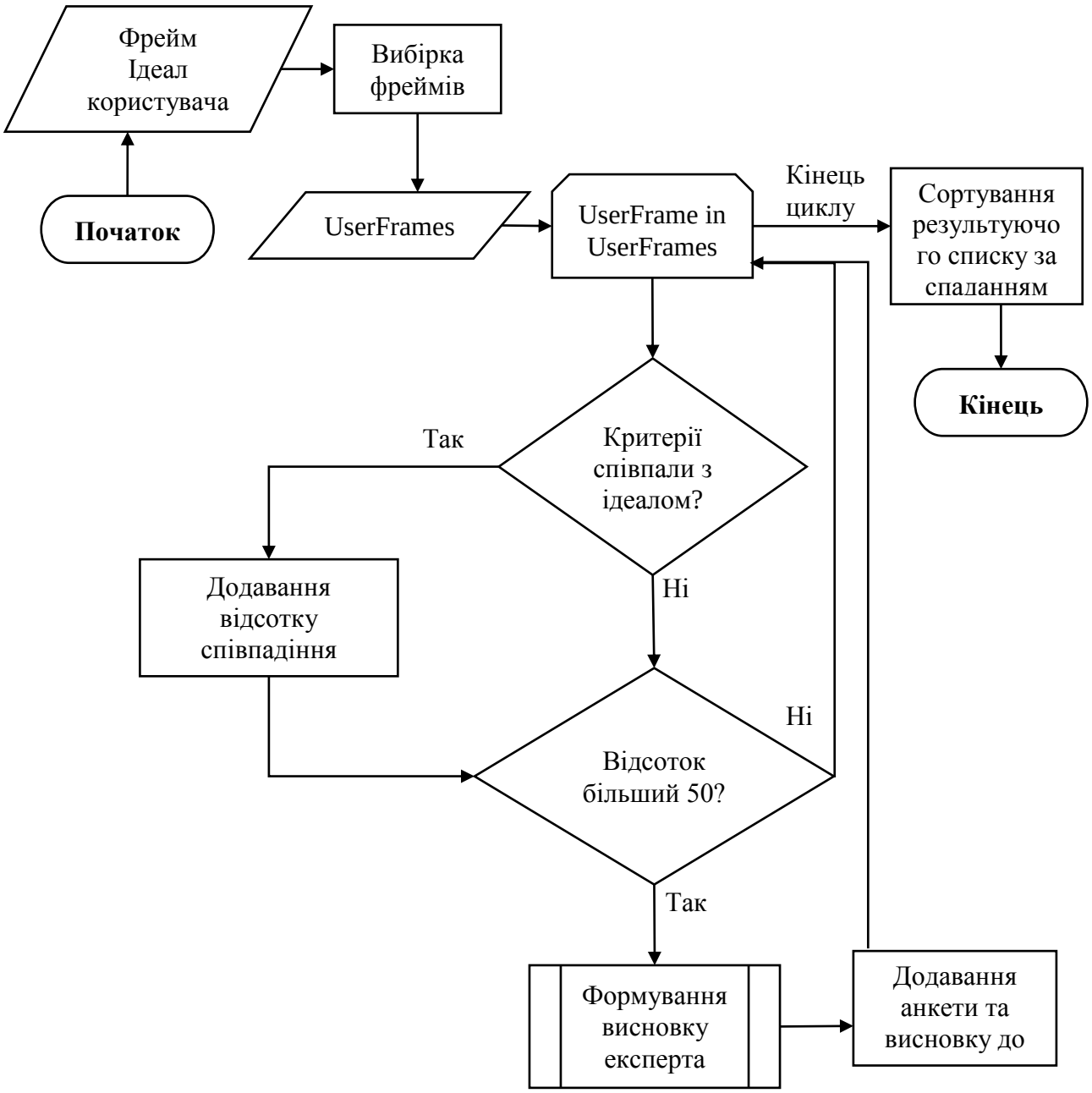


Додаток Д

Структура інформаційної системи

Додаток Е

Блок схема алгоритму способу сумісності анкет

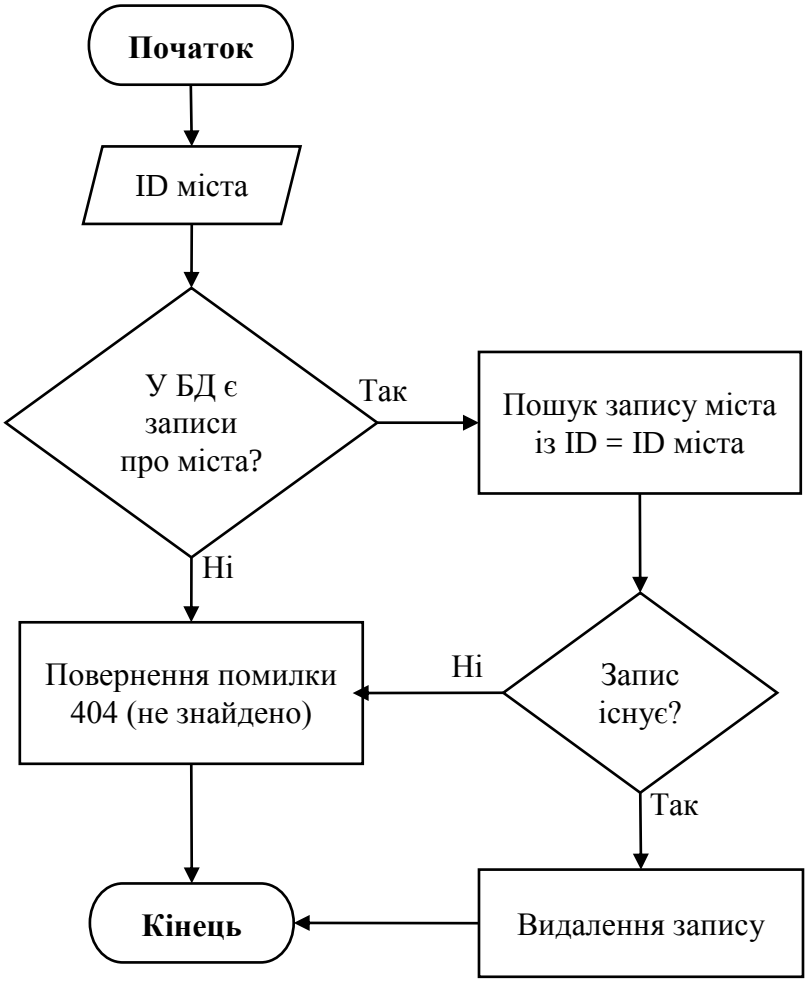


Додаток Ж

Блок схема алгоритму редагування запису про певне місто

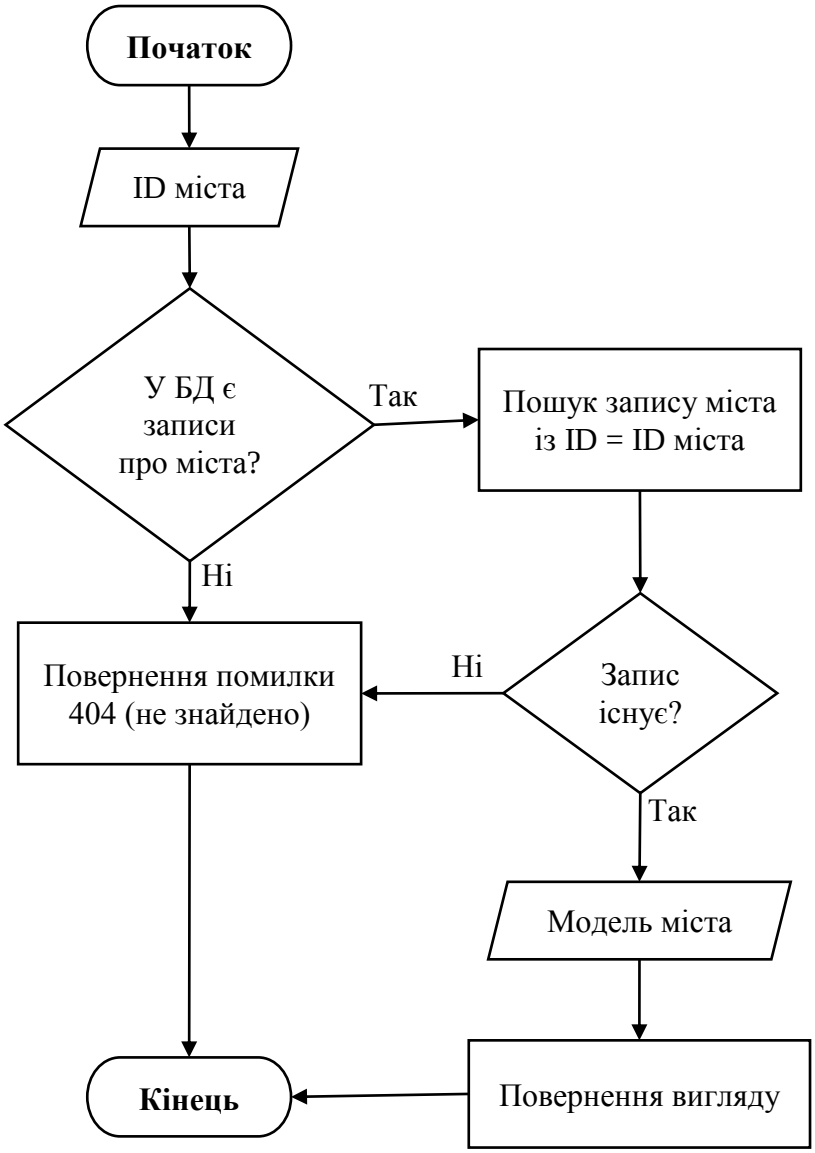
Додаток З

Блок схема алгоритму видалення запису про певне місто



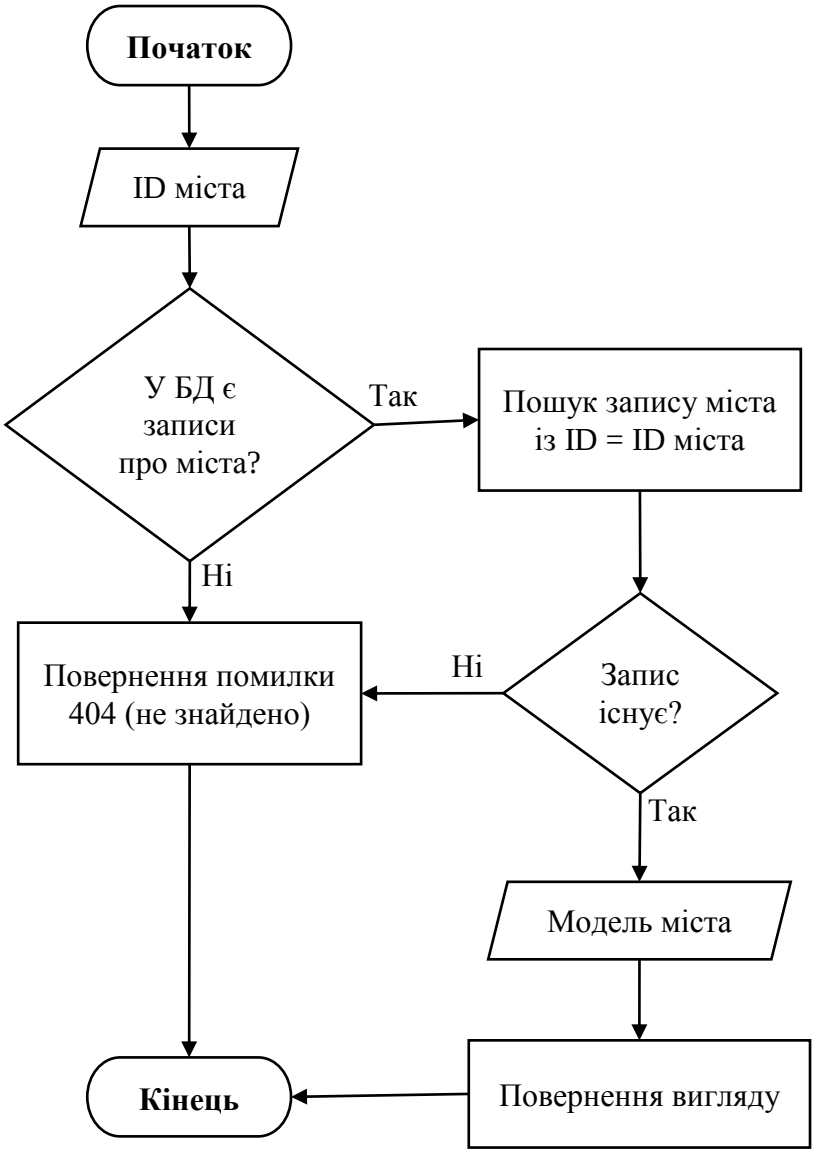
Додаток И

Блок схема алгоритму повернення запису про певне місто



Додаток К

Блок схема алгоритму повернення списку міст



Додаток Л

Блок схема алгоритму додавання запису нового міста

Додаток М

Лістинг програмного коду

Лістинг CompareFramesController.cs

```
using API.Data;
using krb.Models;
using krbApi.Models;
using Microsoft.AspNetCore.Mvc;
using Swashbuckle.AspNetCore.SwaggerGen;
using System.Collections.Generic;
using System.Linq;

namespace API.Controllers
{
   [Route("api/[controller]")]
   [ApiController]
   public class CompareFramesController : ControllerBase
   {
      private readonly APIdbContext _context;
      public CompareFramesController(APIdbContext context)
      {
         _context = context;
      }


      [HttpPost]
      public List<FrameOutput> Post([FromBody] FrameForCompare InputFrame)
      {
         List<FrameOutput> Rel = new List<FrameOutput>();
         var UserFrames = _context.UserFrame.Where(x => x.GenderId == InputFrame.GenderId ).AsEnumerable();
         double percentForOneCriteria = InputFrame.GetMatchingPercentForOneCriteria(),
              percentForOneHobby = (InputFrame.UserHobbies==null)?0: InputFrame.GetMatchingPercentForOneHobby();

         foreach (UserFrame userFrame in UserFrames) {
            FrameOutput frameOutput = new FrameOutput() {
               Name = userFrame.Name,
               Surname = userFrame.Surname,
               FrameId = userFrame.Id,
            };
            string conclusion = "Анкета користувача " + frameOutput.Name + " " + frameOutput.Surname;
            conclusion += " співпадає із шуканим фреймом, та має такі спільні критерії: ";
            List<int> Hobbies = _context.UserHobbies.Where(x => x.FrameId == userFrame.Id).Select(x=>x.HobbyId).ToList();

            frameOutput.SetPercents(percentForOneCriteria,percentForOneHobby);

            AddConclusionToDictionary(frameOutput.Sex, "Id", userFrame.GenderId, true);
            if (frameOutput.Sex["IsMatched"].ToLower() == "true")
            {
               conclusion += "стать, ";
               frameOutput.AddPercentForCriteria();
            }
            AddConclusionToDictionary(frameOutput.Age, "Value", userFrame.Age, (userFrame.Age >= InputFrame.ageLowerBound &&
userFrame.Age <= InputFrame.ageUpperBound) ? true : false);
            if (frameOutput.Age["IsMatched"].ToLower() == "true")
            {
               conclusion += "вік, ";
               frameOutput.AddPercentForCriteria();
            }
            AddConclusionToDictionary(frameOutput.Height, "Value", userFrame.Height, (userFrame.Height >= InputFrame.heightLowerBound &&
userFrame.Height <= InputFrame.heightUpperBound) ? true : false);
            if (frameOutput.Height["IsMatched"].ToLower() == "true")
            {
               conclusion += "зріст, ";
               frameOutput.AddPercentForCriteria();
            }
            AddConclusionToDictionary(frameOutput.Income, "Id", userFrame.IncomeId, (InputFrame.IncomeId > 1 && userFrame.IncomeId ==
InputFrame.IncomeId) ? true : false);
```

```
        if (frameOutput.Income["IsMatched"].ToLower() == "true")
        {
            conclusion += "дохід, ";
            frameOutput.AddPercentForCriteria();
        }
        AddConclusionToDictionary(frameOutput.PersonalityType, "Id", userFrame.PersonalityTypeId, (InputFrame.PersonalityTypeId > 1 &&
userFrame.PersonalityTypeId == InputFrame.PersonalityTypeId) ? true : false);
        if (frameOutput.PersonalityType["IsMatched"].ToLower() == "true")
        {
            conclusion += "Тип особистості, ";
            frameOutput.AddPercentForCriteria();
        }
        AddConclusionToDictionary(frameOutput.City, "Id", userFrame.CityId, (InputFrame.CityId > 1 && userFrame.CityId == InputFrame.CityId) ?
true : false);
        if (frameOutput.City["IsMatched"].ToLower() == "true")
        {
            conclusion += "Місто, ";
            frameOutput.AddPercentForCriteria();
        }
        AddConclusionToDictionary(frameOutput.Proffesion, "Id", userFrame.ProffesionId, (InputFrame.ProffesionId > 1 &&
userFrame.ProffesionId == InputFrame.ProffesionId) ? true : false);
        if (frameOutput.Proffesion["IsMatched"].ToLower() == "true") {
            conclusion += "Професія, ";
            frameOutput.AddPercentForCriteria();
        }

        AddConclusionToDictionary(frameOutput.Education, "Id", userFrame.EducationId, (InputFrame.EducationId > 1 && userFrame.EducationId
== InputFrame.EducationId) ? true : false);
        if (frameOutput.Education["IsMatched"].ToLower() == "true") {
            conclusion += "Освіта./n ";
            frameOutput.AddPercentForCriteria();
        }


        if (frameOutput.MatchingPercent >= 0.50) {
            if (InputFrame.UserHobbies != null && InputFrame.UserHobbies.Intersect(Hobbies).Any()) {
                conclusion += "Також співпали наступні хобі:\n ";
                if (InputFrame.UserHobbies.Count < Hobbies.Count)
                {
                    foreach (int hobbyId in Hobbies)
                    {
                        bool isMatched = InputFrame.UserHobbies.Contains(hobbyId);
                        if (!frameOutput.Hobbies.ContainsKey(hobbyId.ToString()))
                        {
                            AddHobbyConclusionToDictionary(frameOutput.Hobbies, hobbyId.ToString(), isMatched);
                            if (isMatched)
                            {
                                conclusion += $"{_context.Hobby.Find(hobbyId).Name}\n";
                                frameOutput.AddPercentForHobby();
                            }
                        }
                    }
                }
                else
                {
                    foreach (int hobbyId in InputFrame.UserHobbies)
                    {
                        bool isMatched = Hobbies.Contains(hobbyId);
                        if (!frameOutput.Hobbies.ContainsKey(hobbyId.ToString()))
                        {
                            AddHobbyConclusionToDictionary(frameOutput.Hobbies, hobbyId.ToString(), isMatched);
                            if (isMatched)
                            {
                                conclusion += $"{_context.Hobby.Find(hobbyId).Name}\n";

                                frameOutput.AddPercentForHobby();
                            }
                        }
                    }
                }
            }
            conclusion = conclusion.TrimEnd(',', ' ');
            conclusion += ". Загальний відсоток співпадіння: " + Math.Round(frameOutput.MatchingPercent * 100, 2).ToString() + ".";
```

```
                frameOutput.Conclusion = conclusion;
                Rel.Add(frameOutput);
            }
        }
        return Rel.OrderByDescending(x => x.MatchingPercent).Take(5).ToList();

        void AddConclusionToDictionary<T>(Dictionary<string, string> dictionary, string key, T value, bool isMatched = false)
        {
            dictionary.Add(key, value.ToString());
            dictionary.Add($"IsMatched", isMatched.ToString());
        }
        void AddHobbyConclusionToDictionary(Dictionary<string, Dictionary<string, string>> dictionary, string key,  bool isMatched = false)
        {
            Dictionary<string, string> subdictionary = new Dictionary<string, string>();
            subdictionary.Add("Id", key);
            subdictionary.Add($"IsMatched", isMatched.ToString());
            dictionary.Add(key, subdictionary);
        }
    }
  }
}
```

Лістинг UserFramesController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using KrbWebSite.Data;
using krb.Models;
using Microsoft.AspNetCore.Authorization;
using krbApi.Models;
using Newtonsoft.Json;
using System.Text;

namespace KrbWebSite.Controllers
{
  [Authorize]
  public class UserFramesController : Controller
  {
    private readonly KrbWebSiteDbContext _context;

    public UserFramesController(KrbWebSiteDbContext context)
    {
      _context = context;
    }

    // GET: UserFrames
    public async Task<IActionResult> Index()
    {
      var krbWebSiteDbContext = _context.UserFrame.Include(u => u.city).Include(u => u.education).Include(u => u.gender).Include(u =>
u.income).Include(u => u.personalityType).Include(u => u.proffesion);
      return View(await krbWebSiteDbContext.ToListAsync());
    }
    [AllowAnonymous]
    [HttpGet]
    public async Task<IActionResult> FrameCompare()
    {
      ViewData["CityId"] = new SelectList(_context.City, "Id", "Name");
      ViewData["EducationId"] = new SelectList(_context.Education, "Id", "Name");
      ViewData["GenderId"] = new SelectList(_context.Gender, "Id", "Name");
      ViewData["IncomeId"] = new SelectList(_context.Income, "Id", "Quantity");
      ViewData["PersonalityTypeId"] = new SelectList(_context.PersonalityType, "Id", "Name");
      ViewData["ProffesionId"] = new SelectList(_context.Proffesion, "Id", "Name");
      ViewData["Hobbies"] = new SelectList(_context.Hobby, "Id", "Name");

      return View();
    }
    [AllowAnonymous]
```

```
[HttpPost]
public async Task<IActionResult> Result(FrameForCompare frameForCompare, int userId = 0)
{

    using (HttpClient client = new HttpClient())
    {
        // Адреса API-методу
        string apiUrl = "https://localhost:7228/api/CompareFrames";

        // Підготовка даних для відправки
        var json = JsonConvert.SerializeObject(frameForCompare);
        Console.WriteLine(json);
        var content = new StringContent(json, Encoding.UTF8, "application/json");

        // Відправка POST-запиту до API
        HttpResponseMessage response = await client.PostAsync(apiUrl, content);

        // Обробка відповіді API
        if (response.IsSuccessStatusCode)
        {
            // Отримання результату з API
            string apiResponse = await response.Content.ReadAsStringAsync();
            Console.Write(apiResponse);
            // Обробка результату з API
            List<FrameOutput> frameOutputs = JsonConvert.DeserializeObject<List<FrameOutput>>(apiResponse);
            return View(frameOutputs);
        }
        else
        {
            // Обробка помилки від API
            string errorResponse = await response.Content.ReadAsStringAsync();

            // Обробка помилки

            return View();
        }
    }
}
// GET: UserFrames/Details/5
public async Task<IActionResult> Details(int? id)
{
    if (id == null || _context.UserFrame == null)
    {
        return NotFound();
    }

    var userFrame = await _context.UserFrame
        .Include(u => u.city)
        .Include(u => u.education)
        .Include(u => u.gender)
        .Include(u => u.income)
        .Include(u => u.personalityType)
        .Include(u => u.proffesion)
        .FirstOrDefaultAsync(m => m.Id == id);
    if (userFrame == null)
    {
        return NotFound();
    }

    return View(userFrame);
}

// GET: UserFrames/Create
public IActionResult Create()
{
    ViewData["CityId"] = new SelectList(_context.City, "Id", "Name");
    ViewData["EducationId"] = new SelectList(_context.Education, "Id", "Name");
    ViewData["GenderId"] = new SelectList(_context.Gender, "Id", "Name");
    ViewData["IncomeId"] = new SelectList(_context.Income, "Id", "Quantity");
    ViewData["PersonalityTypeId"] = new SelectList(_context.PersonalityType, "Id", "Name");
    ViewData["ProffesionId"] = new SelectList(_context.Proffesion, "Id", "Name");
    ViewData["Hobby"] = new SelectList(_context.Hobby, "Id", "Name");
    return View();
}
```

```
    // POST: UserFrames/Create
    // To protect from overposting attacks, enable the specific properties you want to bind to.
    // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult>
Create([Bind("Id,Name,Surname,EducationId,IncomeId,Age,GenderId,PersonalityTypeId,Childs,Height,Weight,UserId,ProffesionId,CityId")]
UserFrame userFrame)
    {
      if (_context.UserFrame.FirstOrDefault(x => x.UserId == userFrame.UserId) is not null) {
        return RedirectToAction(nameof(Index));
      }
      if (ModelState.IsValid )
      {
        _context.Add(userFrame);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
      }
      ViewData["CityId"] = new SelectList(_context.City, "Id", "Name", userFrame.CityId);
      ViewData["EducationId"] = new SelectList(_context.Education, "Id", "Name", userFrame.EducationId);
      ViewData["GenderId"] = new SelectList(_context.Gender, "Id", "Name", userFrame.GenderId);
      ViewData["IncomeId"] = new SelectList(_context.Income, "Id", "Quantity", userFrame.IncomeId);
      ViewData["PersonalityTypeId"] = new SelectList(_context.PersonalityType, "Id", "Name", userFrame.PersonalityTypeId);
      ViewData["ProffesionId"] = new SelectList(_context.Proffesion, "Id", "Name", userFrame.ProffesionId);
      return View(userFrame);
    }

    // GET: UserFrames/Edit/5
    public async Task<IActionResult> Edit(int? id)
    {
      if (id == null || _context.UserFrame == null)
      {
        return NotFound();
      }

      var userFrame = await _context.UserFrame.FindAsync(id);
      if (userFrame == null)
      {
        return NotFound();
      }
      ViewData["CityId"] = new SelectList(_context.City, "Id", "Id", userFrame.CityId);
      ViewData["EducationId"] = new SelectList(_context.Education, "Id", "Id", userFrame.EducationId);
      ViewData["GenderId"] = new SelectList(_context.Gender, "Id", "Id", userFrame.GenderId);
      ViewData["IncomeId"] = new SelectList(_context.Income, "Id", "Id", userFrame.IncomeId);
      ViewData["PersonalityTypeId"] = new SelectList(_context.PersonalityType, "Id", "Id", userFrame.PersonalityTypeId);
      ViewData["ProffesionId"] = new SelectList(_context.Proffesion, "Id", "Id", userFrame.ProffesionId);
      return View(userFrame);
    }

    // POST: UserFrames/Edit/5
    // To protect from overposting attacks, enable the specific properties you want to bind to.
    // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Edit(int id,
[Bind("Id,Name,Surname,EducationId,IncomeId,Age,GenderId,PersonalityTypeId,Childs,Height,Weight,UserId,ProffesionId,CityId")] UserFrame
userFrame)
    {
      if (id != userFrame.Id)
      {
        return NotFound();
      }

      if (ModelState.IsValid)
      {
        try
        {
          _context.Update(userFrame);
          await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
          if (!UserFrameExists(userFrame.Id))
```

```
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    ViewData["CityId"] = new SelectList(_context.City, "Id", "Id", userFrame.CityId);
    ViewData["EducationId"] = new SelectList(_context.Education, "Id", "Id", userFrame.EducationId);
    ViewData["GenderId"] = new SelectList(_context.Gender, "Id", "Id", userFrame.GenderId);
    ViewData["IncomeId"] = new SelectList(_context.Income, "Id", "Id", userFrame.IncomeId);
    ViewData["PersonalityTypeId"] = new SelectList(_context.PersonalityType, "Id", "Id", userFrame.PersonalityTypeId);
    ViewData["ProffesionId"] = new SelectList(_context.Proffesion, "Id", "Id", userFrame.ProffesionId);
    return View(userFrame);
}

// GET: UserFrames/Delete/5
public async Task<IActionResult> Delete(int? id)
{
    if (id == null || _context.UserFrame == null)
    {
        return NotFound();
    }

    var userFrame = await _context.UserFrame
        .Include(u => u.city)
        .Include(u => u.education)
        .Include(u => u.gender)
        .Include(u => u.income)
        .Include(u => u.personalityType)
        .Include(u => u.proffesion)
        .FirstOrDefaultAsync(m => m.Id == id);
    if (userFrame == null)
    {
        return NotFound();
    }

    return View(userFrame);
}

// POST: UserFrames/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(int id)
{
    if (_context.UserFrame == null)
    {
        return Problem("Entity set 'KrbWebSiteDbContext.UserFrame'  is null.");
    }
    var userFrame = await _context.UserFrame.FindAsync(id);
    if (userFrame != null)
    {
        _context.UserFrame.Remove(userFrame);
    }

    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}

private bool UserFrameExists(int id)
{
  return (_context.UserFrame?.Any(e => e.Id == id)).GetValueOrDefault();
}
  }
}
```

Лістинг UserHobbiesController.cs

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using KrbWebSite.Data;
using krb.Models;
using Microsoft.AspNetCore.Authorization;

namespace KrbWebSite.Controllers
{
    [Authorize]
    public class UserHobbiesController : Controller
    {
        private readonly KrbWebSiteDbContext _context;
        public UserHobbiesController(KrbWebSiteDbContext context)
        {
            _context = context;
        }

        // GET: UserHobbies
        public async Task<IActionResult> Index()
        {
            var krbWebSiteDbContext = _context.UserHobbies.Include(u =>
u.Frame).Include(u => u.Hobbies);
            return View(await krbWebSiteDbContext.ToListAsync());
        }

        // GET: UserHobbies/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null || _context.UserHobbies == null)
            {
                return NotFound();
            }

            var userHobbies = await _context.UserHobbies
                .Include(u => u.Frame)
                .Include(u => u.Hobbies)
                .FirstOrDefaultAsync(m => m.Id == id);
            if (userHobbies == null)
            {
                return NotFound();
            }

            return View(userHobbies);
        }

        // GET: UserHobbies/Create
        public IActionResult Create()
        {
            ViewData["FrameId"] = new SelectList(_context.UserFrame, "Id", "Id");
            ViewData["HobbyId"] = new SelectList(_context.Hobby, "Id", "Id");
            return View();
        }

        // POST: UserHobbies/Create
        // To protect from overposting attacks, enable the specific properties you
want to bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create([Bind("Id,FrameId,HobbyId")]
UserHobbies userHobbies)
        {
            if (ModelState.IsValid)
```

```
            {


                _context.Add(userHobbies);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            ViewData["FrameId"] = new SelectList(_context.UserFrame, "Id", "Id",
userHobbies.FrameId);
            ViewData["HobbyId"] = new SelectList(_context.Hobby, "Id", "Id",
userHobbies.HobbyId);
            return View(userHobbies);
        }

        // GET: UserHobbies/Edit/5
        public async Task<IActionResult> Edit(int? id)
        {
            if (id == null || _context.UserHobbies == null)
            {
                return NotFound();
            }

            var userHobbies = await _context.UserHobbies.FindAsync(id);
            if (userHobbies == null)
            {
                return NotFound();
            }
            ViewData["FrameId"] = new SelectList(_context.UserFrame, "Id", "Id",
userHobbies.FrameId);
            ViewData["HobbyId"] = new SelectList(_context.Hobby, "Id", "Id",
userHobbies.HobbyId);
            return View(userHobbies);
        }

        // POST: UserHobbies/Edit/5
        // To protect from overposting attacks, enable the specific properties you
want to bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id, [Bind("Id,FrameId,HobbyId")]
UserHobbies userHobbies)
        {
            if (id != userHobbies.Id)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(userHobbies);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!UserHobbiesExists(userHobbies.Id))
                    {
                        return NotFound();
                    }
                    else
                    {
                        throw;
                    }
                }
                return RedirectToAction(nameof(Index));
```

```
            }
            ViewData["FrameId"] = new SelectList(_context.UserFrame, "Id", "Id",
userHobbies.FrameId);
            ViewData["HobbyId"] = new SelectList(_context.Hobby, "Id", "Id",
userHobbies.HobbyId);
            return View(userHobbies);
        }

        // GET: UserHobbies/Delete/5
        public async Task<IActionResult> Delete(int? id)
        {
            if (id == null || _context.UserHobbies == null)
            {
                return NotFound();
            }

            var userHobbies = await _context.UserHobbies
                .Include(u => u.Frame)
                .Include(u => u.Hobbies)
                .FirstOrDefaultAsync(m => m.Id == id);
            if (userHobbies == null)
            {
                return NotFound();
            }

            return View(userHobbies);
        }

        // POST: UserHobbies/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            if (_context.UserHobbies == null)
            {
                return Problem("Entity set 'KrbWebSiteDbContext.UserHobbies'  is
null.");
            }
            var userHobbies = await _context.UserHobbies.FindAsync(id);
            if (userHobbies != null)
            {
                _context.UserHobbies.Remove(userHobbies);
            }

            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }

        private bool UserHobbiesExists(int id)
        {
          return (_context.UserHobbies?.Any(e => e.Id == id)).GetValueOrDefault();
        }
    }
}
```

Лістинг ProffesionsController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using KrbWebSite.Data;
using krb.Models;
using Microsoft.AspNetCore.Authorization;
```

```
namespace KrbWebSite.Controllers
{
    [Authorize(Roles = "Admin")]
    public class ProffesionsController : Controller
    {
        private readonly KrbWebSiteDbContext _context;

        public ProffesionsController(KrbWebSiteDbContext context)
        {
            _context = context;
        }

        // GET: Proffesions
        public async Task<IActionResult> Index()
        {
             return _context.Proffesion != null ?
                        View(await _context.Proffesion.ToListAsync()) :
                        Problem("Entity set 'KrbWebSiteDbContext.Proffesion'  is null.");
        }

        // GET: Proffesions/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null || _context.Proffesion == null)
            {
                return NotFound();
            }

            var proffesion = await _context.Proffesion
                .FirstOrDefaultAsync(m => m.Id == id);
            if (proffesion == null)
            {
                return NotFound();
            }

            return View(proffesion);
        }

        // GET: Proffesions/Create
        public IActionResult Create()
        {
            return View();
        }

        // POST: Proffesions/Create
        // To protect from overposting attacks, enable the specific properties you want to bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create([Bind("Id,Name,Description")] Proffesion proffesion)
        {
            if (ModelState.IsValid)
            {
                _context.Add(proffesion);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(proffesion);
        }

        // GET: Proffesions/Edit/5
        public async Task<IActionResult> Edit(int? id)
        {
            if (id == null || _context.Proffesion == null)
            {
                return NotFound();
            }

            var proffesion = await _context.Proffesion.FindAsync(id);
            if (proffesion == null)
            {
                return NotFound();
            }
```

```
    return View(proffesion);
}

// POST: Proffesions/Edit/5
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id, [Bind("Id,Name,Description")] Proffesion proffesion)
{
    if (id != proffesion.Id)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(proffesion);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!ProffesionExists(proffesion.Id))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    return View(proffesion);
}

// GET: Proffesions/Delete/5
public async Task<IActionResult> Delete(int? id)
{
    if (id == null || _context.Proffesion == null)
    {
        return NotFound();
    }

    var proffesion = await _context.Proffesion
        .FirstOrDefaultAsync(m => m.Id == id);
    if (proffesion == null)
    {
        return NotFound();
    }

    return View(proffesion);
}

// POST: Proffesions/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(int id)
{
    if (_context.Proffesion == null)
    {
        return Problem("Entity set 'KrbWebSiteDbContext.Proffesion'  is null.");
    }
    var proffesion = await _context.Proffesion.FindAsync(id);
    if (proffesion != null)
    {
        _context.Proffesion.Remove(proffesion);
    }

    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}
```

```
    private bool ProffesionExists(int id)
    {
      return (_context.Proffesion?.Any(e => e.Id == id)).GetValueOrDefault();
    }
  }
}
```

Лістинг PersonalityTypesController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using KrbWebSite.Data;
using krb.Models;
using Microsoft.AspNetCore.Authorization;

namespace KrbWebSite.Controllers
{
  [Authorize(Roles ="Admin")]
  public class PersonalityTypesController : Controller
  {
    private readonly KrbWebSiteDbContext _context;

    public PersonalityTypesController(KrbWebSiteDbContext context)
    {
      _context = context;
    }

    // GET: PersonalityTypes
    public async Task<IActionResult> Index()
    {
        return _context.PersonalityType != null ?
                    View(await _context.PersonalityType.ToListAsync()) :
                    Problem("Entity set 'KrbWebSiteDbContext.PersonalityType'  is null.");
    }

    // GET: PersonalityTypes/Details/5
    public async Task<IActionResult> Details(int? id)
    {
      if (id == null || _context.PersonalityType == null)
      {
        return NotFound();
      }

      var personalityType = await _context.PersonalityType
        .FirstOrDefaultAsync(m => m.Id == id);
      if (personalityType == null)
      {
        return NotFound();
      }

      return View(personalityType);
    }

    // GET: PersonalityTypes/Create
    public IActionResult Create()
    {
      return View();
    }

    // POST: PersonalityTypes/Create
    // To protect from overposting attacks, enable the specific properties you want to bind to.
    // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Create([Bind("Id,Name")] PersonalityType personalityType)
    {
      if (ModelState.IsValid)
```

```
    {
        _context.Add(personalityType);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    return View(personalityType);
}

// GET: PersonalityTypes/Edit/5
public async Task<IActionResult> Edit(int? id)
{
    if (id == null || _context.PersonalityType == null)
    {
        return NotFound();
    }

    var personalityType = await _context.PersonalityType.FindAsync(id);
    if (personalityType == null)
    {
        return NotFound();
    }
    return View(personalityType);
}

// POST: PersonalityTypes/Edit/5
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id, [Bind("Id,Name")] PersonalityType personalityType)
{
    if (id != personalityType.Id)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(personalityType);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!PersonalityTypeExists(personalityType.Id))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    return View(personalityType);
}

// GET: PersonalityTypes/Delete/5
public async Task<IActionResult> Delete(int? id)
{
    if (id == null || _context.PersonalityType == null)
    {
        return NotFound();
    }

    var personalityType = await _context.PersonalityType
        .FirstOrDefaultAsync(m => m.Id == id);
    if (personalityType == null)
    {
        return NotFound();
    }
```

```
            return View(personalityType);
        }

        // POST: PersonalityTypes/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            if (_context.PersonalityType == null)
            {
                return Problem("Entity set 'KrbWebSiteDbContext.PersonalityType'  is null.");
            }
            var personalityType = await _context.PersonalityType.FindAsync(id);
            if (personalityType != null)
            {
                _context.PersonalityType.Remove(personalityType);
            }

            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }

        private bool PersonalityTypeExists(int id)
        {
          return (_context.PersonalityType?.Any(e => e.Id == id)).GetValueOrDefault();
        }
    }
}
```

Лістинг IncomesController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using KrbWebSite.Data;
using krb.Models;
using Microsoft.AspNetCore.Authorization;

namespace KrbWebSite.Controllers
{
    [Authorize(Roles = "Admin")]
    public class IncomesController : Controller
    {
        private readonly KrbWebSiteDbContext _context;

        public IncomesController(KrbWebSiteDbContext context)
        {
            _context = context;
        }

        // GET: Incomes
        public async Task<IActionResult> Index()
        {
             return _context.Income != null ?
                        View(await _context.Income.ToListAsync()) :
                        Problem("Entity set 'KrbWebSiteDbContext.Income'  is null.");
        }

        // GET: Incomes/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null || _context.Income == null)
            {
                return NotFound();
            }

            var income = await _context.Income
                .FirstOrDefaultAsync(m => m.Id == id);
            if (income == null)
```

```
    {
        return NotFound();
    }

    return View(income);
}

// GET: Incomes/Create
public IActionResult Create()
{
    return View();
}

// POST: Incomes/Create
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Create([Bind("Id,Quantity")] Income income)
{
    if (ModelState.IsValid)
    {
        _context.Add(income);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    return View(income);
}

// GET: Incomes/Edit/5
public async Task<IActionResult> Edit(int? id)
{
    if (id == null || _context.Income == null)
    {
        return NotFound();
    }

    var income = await _context.Income.FindAsync(id);
    if (income == null)
    {
        return NotFound();
    }
    return View(income);
}

// POST: Incomes/Edit/5
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id, [Bind("Id,Quantity")] Income income)
{
    if (id != income.Id)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(income);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!IncomeExists(income.Id))
            {
                return NotFound();
            }
            else
            {
                throw;
```

```
                }
            }
            return RedirectToAction(nameof(Index));
        }
        return View(income);
    }

    // GET: Incomes/Delete/5
    public async Task<IActionResult> Delete(int? id)
    {
        if (id == null || _context.Income == null)
        {
            return NotFound();
        }

        var income = await _context.Income
            .FirstOrDefaultAsync(m => m.Id == id);
        if (income == null)
        {
            return NotFound();
        }

        return View(income);
    }

    // POST: Incomes/Delete/5
    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> DeleteConfirmed(int id)
    {
        if (_context.Income == null)
        {
            return Problem("Entity set 'KrbWebSiteDbContext.Income'  is null.");
        }
        var income = await _context.Income.FindAsync(id);
        if (income != null)
        {
            _context.Income.Remove(income);
        }

        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }

    private bool IncomeExists(int id)
    {
      return (_context.Income?.Any(e => e.Id == id)).GetValueOrDefault();
    }
  }
}
```

Лістинг HobbiesController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using KrbWebSite.Data;
using krb.Models;
using Microsoft.AspNetCore.Authorization;

namespace KrbWebSite.Controllers
{
  [Authorize(Roles = "Admin")]
  public class HobbiesController : Controller
  {
    private readonly KrbWebSiteDbContext _context;

    public HobbiesController(KrbWebSiteDbContext context)
    {
```

```
    _context = context;
}

// GET: Hobbies
public async Task<IActionResult> Index()
{
     return _context.Hobby != null ?
                View(await _context.Hobby.ToListAsync()) :
                Problem("Entity set 'KrbWebSiteDbContext.Hobby'  is null.");
}

// GET: Hobbies/Details/5
public async Task<IActionResult> Details(int? id)
{
    if (id == null || _context.Hobby == null)
    {
        return NotFound();
    }

    var hobby = await _context.Hobby
        .FirstOrDefaultAsync(m => m.Id == id);
    if (hobby == null)
    {
        return NotFound();
    }

    return View(hobby);
}

// GET: Hobbies/Create
public IActionResult Create()
{
    return View();
}

// POST: Hobbies/Create
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Create([Bind("Id,Name,Description")] Hobby hobby)
{
    if (ModelState.IsValid)
    {
        _context.Add(hobby);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    return View(hobby);
}

// GET: Hobbies/Edit/5
public async Task<IActionResult> Edit(int? id)
{
    if (id == null || _context.Hobby == null)
    {
        return NotFound();
    }

    var hobby = await _context.Hobby.FindAsync(id);
    if (hobby == null)
    {
        return NotFound();
    }
    return View(hobby);
}

// POST: Hobbies/Edit/5
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id, [Bind("Id,Name,Description")] Hobby hobby)
{
```

```
            if (id != hobby.Id)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(hobby);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!HobbyExists(hobby.Id))
                    {
                        return NotFound();
                    }
                    else
                    {
                        throw;
                    }
                }
                return RedirectToAction(nameof(Index));
            }
            return View(hobby);
        }

        // GET: Hobbies/Delete/5
        public async Task<IActionResult> Delete(int? id)
        {
            if (id == null || _context.Hobby == null)
            {
                return NotFound();
            }

            var hobby = await _context.Hobby
                .FirstOrDefaultAsync(m => m.Id == id);
            if (hobby == null)
            {
                return NotFound();
            }

            return View(hobby);
        }

        // POST: Hobbies/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            if (_context.Hobby == null)
            {
                return Problem("Entity set 'KrbWebSiteDbContext.Hobby'  is null.");
            }
            var hobby = await _context.Hobby.FindAsync(id);
            if (hobby != null)
            {
                _context.Hobby.Remove(hobby);
            }

            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }

        private bool HobbyExists(int id)
        {
          return (_context.Hobby?.Any(e => e.Id == id)).GetValueOrDefault();
        }
    }
}
```

Лістинг GendersController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using KrbWebSite.Data;
using krb.Models;
using Microsoft.AspNetCore.Authorization;

namespace KrbWebSite.Controllers
{
    [Authorize(Roles = "Admin")]
    public class GendersController : Controller
    {
        private readonly KrbWebSiteDbContext _context;

        public GendersController(KrbWebSiteDbContext context)
        {
            _context = context;
        }

        // GET: Genders
        public async Task<IActionResult> Index()
        {
             return _context.Gender != null ?
                        View(await _context.Gender.ToListAsync()) :
                        Problem("Entity set 'KrbWebSiteDbContext.Gender'  is null.");
        }

        // GET: Genders/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null || _context.Gender == null)
            {
                return NotFound();
            }

            var gender = await _context.Gender
                .FirstOrDefaultAsync(m => m.Id == id);
            if (gender == null)
            {
                return NotFound();
            }

            return View(gender);
        }

        // GET: Genders/Create
        public IActionResult Create()
        {
            return View();
        }

        // POST: Genders/Create
        // To protect from overposting attacks, enable the specific properties you want to bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create([Bind("Id,Name")] Gender gender)
        {
            if (ModelState.IsValid)
            {
                _context.Add(gender);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(gender);
        }

        // GET: Genders/Edit/5
        public async Task<IActionResult> Edit(int? id)
        {
```

```
    if (id == null || _context.Gender == null)
    {
        return NotFound();
    }

    var gender = await _context.Gender.FindAsync(id);
    if (gender == null)
    {
        return NotFound();
    }
    return View(gender);
}

// POST: Genders/Edit/5
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id, [Bind("Id,Name")] Gender gender)
{
    if (id != gender.Id)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(gender);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!GenderExists(gender.Id))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    return View(gender);
}

// GET: Genders/Delete/5
public async Task<IActionResult> Delete(int? id)
{
    if (id == null || _context.Gender == null)
    {
        return NotFound();
    }

    var gender = await _context.Gender
        .FirstOrDefaultAsync(m => m.Id == id);
    if (gender == null)
    {
        return NotFound();
    }

    return View(gender);
}

// POST: Genders/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(int id)
{
    if (_context.Gender == null)
    {
        return Problem("Entity set 'KrbWebSiteDbContext.Gender'  is null.");
```

```
        }
        var gender = await _context.Gender.FindAsync(id);
        if (gender != null)
        {
            _context.Gender.Remove(gender);
        }

        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }

    private bool GenderExists(int id)
    {
      return (_context.Gender?.Any(e => e.Id == id)).GetValueOrDefault();
    }
  }
}
```

Лістинг EducationsController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using KrbWebSite.Data;
using KrbWebSite.Models;
using Microsoft.AspNetCore.Authorization;

namespace KrbWebSite.Controllers
{
  [Authorize(Roles = "Admin")]
  public class EducationsController : Controller
  {
    private readonly KrbWebSiteDbContext _context;

    public EducationsController(KrbWebSiteDbContext context)
    {
      _context = context;
    }

    // GET: Educations
    public async Task<IActionResult> Index()
    {
      var krbWebSiteDbContext = _context.Education.Include(e => e.degree);
      return View(await krbWebSiteDbContext.ToListAsync());
    }

    // GET: Educations/Details/5
    public async Task<IActionResult> Details(int? id)
    {
      if (id == null || _context.Education == null)
      {
        return NotFound();
      }

      var education = await _context.Education
        .Include(e => e.degree)
        .FirstOrDefaultAsync(m => m.Id == id);
      if (education == null)
      {
        return NotFound();
      }

      return View(education);
    }

    // GET: Educations/Create
    public IActionResult Create()
    {
      ViewData["DegreeId"] = new SelectList(_context.Degree, "Id", "Id");
```

```
    return View();
}

// POST: Educations/Create
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Create([Bind("Id,Name,DegreeId")] Education education)
{
    if (ModelState.IsValid)
    {
        _context.Add(education);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    ViewData["DegreeId"] = new SelectList(_context.Degree, "Id", "Id", education.DegreeId);
    return View(education);
}

// GET: Educations/Edit/5
public async Task<IActionResult> Edit(int? id)
{
    if (id == null || _context.Education == null)
    {
        return NotFound();
    }

    var education = await _context.Education.FindAsync(id);
    if (education == null)
    {
        return NotFound();
    }
    ViewData["DegreeId"] = new SelectList(_context.Degree, "Id", "Id", education.DegreeId);
    return View(education);
}

// POST: Educations/Edit/5
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id, [Bind("Id,Name,DegreeId")] Education education)
{
    if (id != education.Id)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(education);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!EducationExists(education.Id))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    ViewData["DegreeId"] = new SelectList(_context.Degree, "Id", "Id", education.DegreeId);
    return View(education);
}

// GET: Educations/Delete/5
```

```
        public async Task<IActionResult> Delete(int? id)
        {
            if (id == null || _context.Education == null)
            {
                return NotFound();
            }

            var education = await _context.Education
                .Include(e => e.degree)
                .FirstOrDefaultAsync(m => m.Id == id);
            if (education == null)
            {
                return NotFound();
            }

            return View(education);
        }

        // POST: Educations/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            if (_context.Education == null)
            {
                return Problem("Entity set 'KrbWebSiteDbContext.Education'  is null.");
            }
            var education = await _context.Education.FindAsync(id);
            if (education != null)
            {
                _context.Education.Remove(education);
            }

            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }

        private bool EducationExists(int id)
        {
          return (_context.Education?.Any(e => e.Id == id)).GetValueOrDefault();
        }
    }
}
```

Лістинг DegreesController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using KrbWebSite.Data;
using KrbWebSite.Models;
using Microsoft.AspNetCore.Authorization;

namespace KrbWebSite.Controllers
{
    [Authorize(Roles = "Admin")]
    public class DegreesController : Controller
    {
        private readonly KrbWebSiteDbContext _context;

        public DegreesController(KrbWebSiteDbContext context)
        {
            _context = context;
        }

        // GET: Degrees
        public async Task<IActionResult> Index()
        {
            return _context.Degree != null ?
```

```
            View(await _context.Degree.ToListAsync()) :
            Problem("Entity set 'KrbWebSiteDbContext.Degree'  is null.");
}

// GET: Degrees/Details/5
public async Task<IActionResult> Details(int? id)
{
   if (id == null || _context.Degree == null)
   {
      return NotFound();
   }

   var degree = await _context.Degree
      .FirstOrDefaultAsync(m => m.Id == id);
   if (degree == null)
   {
      return NotFound();
   }

   return View(degree);
}

// GET: Degrees/Create
public IActionResult Create()
{
   return View();
}

// POST: Degrees/Create
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Create([Bind("Id,Name")] Degree degree)
{
   if (ModelState.IsValid)
   {
      _context.Add(degree);
      await _context.SaveChangesAsync();
      return RedirectToAction(nameof(Index));
   }
   return View(degree);
}

// GET: Degrees/Edit/5
public async Task<IActionResult> Edit(int? id)
{
   if (id == null || _context.Degree == null)
   {
      return NotFound();
   }

   var degree = await _context.Degree.FindAsync(id);
   if (degree == null)
   {
      return NotFound();
   }
   return View(degree);
}

// POST: Degrees/Edit/5
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id, [Bind("Id,Name")] Degree degree)
{
   if (id != degree.Id)
   {
      return NotFound();
   }

   if (ModelState.IsValid)
   {
```

```
            try
            {
                _context.Update(degree);
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!DegreeExists(degree.Id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }
            return RedirectToAction(nameof(Index));
        }
        return View(degree);
    }

    // GET: Degrees/Delete/5
    public async Task<IActionResult> Delete(int? id)
    {
        if (id == null || _context.Degree == null)
        {
            return NotFound();
        }

        var degree = await _context.Degree
            .FirstOrDefaultAsync(m => m.Id == id);
        if (degree == null)
        {
            return NotFound();
        }

        return View(degree);
    }

    // POST: Degrees/Delete/5
    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> DeleteConfirmed(int id)
    {
        if (_context.Degree == null)
        {
            return Problem("Entity set 'KrbWebSiteDbContext.Degree'  is null.");
        }
        var degree = await _context.Degree.FindAsync(id);
        if (degree != null)
        {
            _context.Degree.Remove(degree);
        }

        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }

    private bool DegreeExists(int id)
    {
      return (_context.Degree?.Any(e => e.Id == id)).GetValueOrDefault();
    }
  }
}
```

Лістинг CountriesController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
```

```
using Microsoft.EntityFrameworkCore;
using KrbWebSite.Data;
using krb.Models;
using Microsoft.AspNetCore.Authorization;

namespace KrbWebSite.Controllers
{
    [Authorize(Roles = "Admin")]
    public class CountriesController : Controller
    {
        private readonly KrbWebSiteDbContext _context;

        public CountriesController(KrbWebSiteDbContext context)
        {
            _context = context;
        }

        // GET: Countries
        public async Task<IActionResult> Index()
        {
            return _context.Country != null ?
                        View(await _context.Country.ToListAsync()) :
                        Problem("Entity set 'KrbWebSiteDbContext.Country'  is null.");
        }

        // GET: Countries/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null || _context.Country == null)
            {
                return NotFound();
            }

            var country = await _context.Country
                .FirstOrDefaultAsync(m => m.Id == id);
            if (country == null)
            {
                return NotFound();
            }

            return View(country);
        }

        // GET: Countries/Create
        public IActionResult Create()
        {
            return View();
        }

        // POST: Countries/Create
        // To protect from overposting attacks, enable the specific properties you want to bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create([Bind("Id,Name")] Country country)
        {
            if (ModelState.IsValid)
            {
                _context.Add(country);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(country);
        }

        // GET: Countries/Edit/5
        public async Task<IActionResult> Edit(int? id)
        {
            if (id == null || _context.Country == null)
            {
                return NotFound();
            }

            var country = await _context.Country.FindAsync(id);
```

```
    if (country == null)
    {
        return NotFound();
    }
    return View(country);
}

// POST: Countries/Edit/5
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id, [Bind("Id,Name")] Country country)
{
    if (id != country.Id)
    {
        return NotFound();
    }
    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(country);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!CountryExists(country.Id))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    return View(country);
}

// GET: Countries/Delete/5
public async Task<IActionResult> Delete(int? id)
{
    if (id == null || _context.Country == null)
    {
        return NotFound();
    }

    var country = await _context.Country
        .FirstOrDefaultAsync(m => m.Id == id);
    if (country == null)
    {
        return NotFound();
    }

    return View(country);
}

// POST: Countries/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(int id)
{
    if (_context.Country == null)
    {
        return Problem("Entity set 'KrbWebSiteDbContext.Country'  is null.");
    }
    var country = await _context.Country.FindAsync(id);
    if (country != null)
    {
        _context.Country.Remove(country);
    }
```

```
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
      }

      private bool CountryExists(int id)
      {
        return (_context.Country?.Any(e => e.Id == id)).GetValueOrDefault();
      }
    }
}
```

Лістинг CitiesController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using KrbWebSite.Data;
using krb.Models;
using Microsoft.AspNetCore.Authorization;
using System.Net.Http;

namespace KrbWebSite.Controllers
{
    [Authorize(Roles = "Admin")]
    public class CitiesController : Controller
    {
        private readonly KrbWebSiteDbContext _context;
        private readonly HttpClient httpClient;

        public CitiesController(KrbWebSiteDbContext context, HttpClient httpClient)
        {
            _context = context;
            this.httpClient = httpClient;
        }

        // GET: Cities
        public async Task<IActionResult> Index()
        {
            var krbWebSiteDbContext = _context.City.Include(c => c.country);
            return View(await krbWebSiteDbContext.ToListAsync());
        }

        // GET: Cities/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null || _context.City == null)
            {
                return NotFound();
            }

            var city = await _context.City
                .Include(c => c.country)
                .FirstOrDefaultAsync(m => m.Id == id);
            if (city == null)
            {
                return NotFound();
            }

            return View(city);
        }

        // GET: Cities/Create
        public IActionResult Create()
        {
            ViewData["CountryId"] = new SelectList(_context.Set<Country>(), "Id", "Name");
            return View();
        }

        // POST: Cities/Create
```

```
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Create([Bind("Id,Name,CountryId")] City city)
{
   if (ModelState.IsValid)
   {
      _context.Add(city);
      await _context.SaveChangesAsync();
      return RedirectToAction(nameof(Index));
   }
   ViewData["CountryId"] = new SelectList(_context.Set<Country>(), "Id", "Id", city.CountryId);
   return View(city);
}

// GET: Cities/Edit/5
public async Task<IActionResult> Edit(int? id)
{
   if (id == null || _context.City == null)
   {
      return NotFound();
   }

   var city = await _context.City.FindAsync(id);
   if (city == null)
   {
      return NotFound();
   }
   ViewData["CountryId"] = new SelectList(_context.Set<Country>(), "Id", "Name", city.CountryId);
   return View(city);
}

// POST: Cities/Edit/5
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id, [Bind("Id,Name,CountryId")] City city)
{
   if (id != city.Id)
   {
      return NotFound();
   }

   if (ModelState.IsValid)
   {
      try
      {
         _context.Update(city);
         await _context.SaveChangesAsync();
      }
      catch (DbUpdateConcurrencyException)
      {
         if (!CityExists(city.Id))
         {
            return NotFound();
         }
         else
         {
            throw;
         }
      }
      return RedirectToAction(nameof(Index));
   }
   ViewData["CountryId"] = new SelectList(_context.Set<Country>(), "Id", "Id", city.CountryId);
   return View(city);
}

// GET: Cities/Delete/5
public async Task<IActionResult> Delete(int? id)
{
   if (id == null || _context.City == null)
   {
```

```
            return NotFound();
        }

        var city = await _context.City
            .Include(c => c.country)
            .FirstOrDefaultAsync(m => m.Id == id);
        if (city == null)
        {
            return NotFound();
        }

        return View(city);
    }

    // POST: Cities/Delete/5
    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> DeleteConfirmed(int id)
    {
        if (_context.City == null)
        {
            return Problem("Entity set 'KrbWebSiteDbContext.City'  is null.");
        }
        var city = await _context.City.FindAsync(id);
        if (city != null)
        {
            _context.City.Remove(city);
        }

        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }

    private bool CityExists(int id)
    {
      return (_context.City?.Any(e => e.Id == id)).GetValueOrDefault();
    }
  }
}
```

Додаток Н

Презентаційний матеріал

Кваліфікаційна Робота Бакалавра

Спосіб оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств

ВИКОНАВ:
Студент 4 курсу, групи КН-19-1
Жук Дмитро Іванович

КЕРІВНИК:
викладач кафедри КН
Собко Олена Віталіївна

Актуальність

Раніше знайомились у громадських місцях, але у 21 столітті додався ще один спосіб знайомств. Це онлайн знайомства, усе що для цього потрібно це смартфон який підключений до мережі інтернет. Виникає проблема пошуку кандидатів. Людина хоче знайти свого ідеала. Отже, розробка способу оцінювання сумісності анкет є актуальною у наш час.

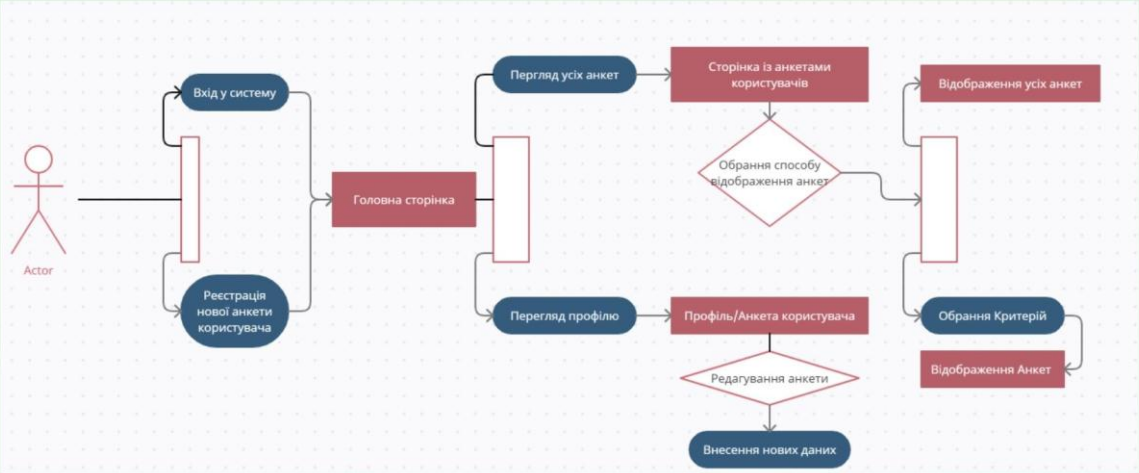
Мета та постановка задачі

- Розробити спосіб оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств;
- Розробити модель фрейму для узагальненого подання анкет користувачів та критеріїв пошуку користувачів для порівняння;
- Спроєктувати структуру інформаційної системи у вигляді соціальної інтелектуальної вебсистеми для знайомств, яка буде використовувати розроблений спосіб оцінювання анкет
- Спроєктувати структуру бази даних для соціальної інтелектуальної вебсистеми для знайомств;
- Виконати вибір засобів розробки соціальної інтелектуальної вебсистеми для знайомств;
- Провести тестування розробленої соціальної інтелектуальної вебсистеми для знайомств.

Структура інформаційної системи

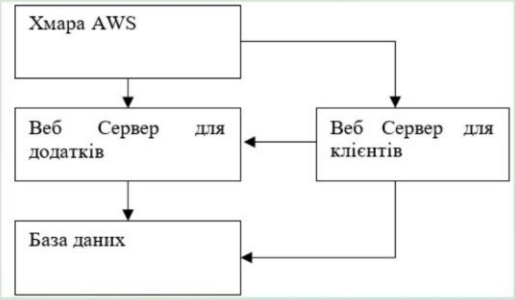
- База даних
- Підсистема реєстрації та авторизації
- Підсистема формування фрейму анкети користувача
- Підсистема оцінювання сумісності анкет користувачів
- Підсистема формування експертних висновків
- Підсистема для роботи із БД

UML - діаграма активностей користувача



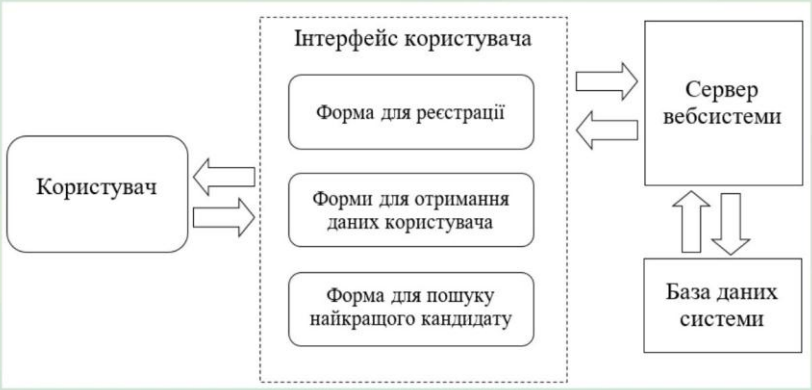
Ця діаграма потрібна для передбачення усіх можливих сценаріїв дій користувача. Тобто переходами між можливими діями системи.

UML - Діаграма розгортання системи на сервісі AWS



Основною метою цієї діаграми є відображення обчислювальних вузлів, на яких працює програма. Та зв'язків між цими вузлами.

UML - Діаграма потоків даних системи

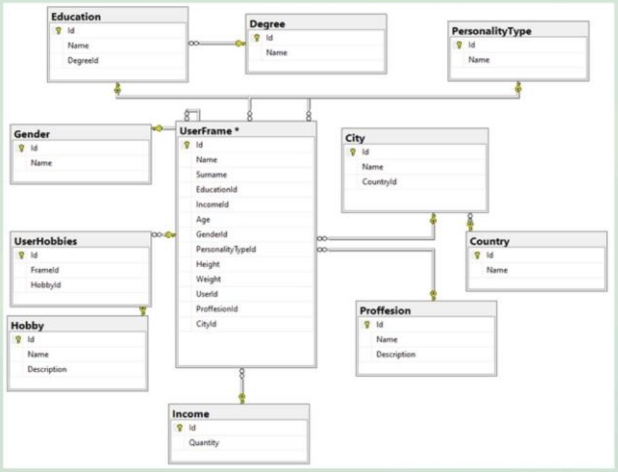


Призначення цієї діаграми – це відображення потоків даних всередині інформаційної системи

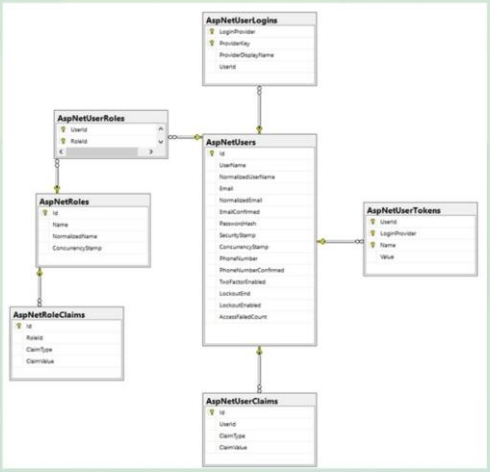
Засоби розробки

- Основною мовою програмування було використано C#
- Для створення Веб сайту було застосовано ASP.NET MVC
- Для створення додатку для оцінювання сумісності анкет було використано ASP.NET API
- Для зв'язку із базою даних було використано бібліотеку EntityFramework Core
- Для реєстрації та авторизації було використано ASP.NET Identity

Даталогічна модель бази даних для додатку способу оцінювання сумісності анкет



Даталогічна модель бази даних для веб сайту знайомств



Вхідні дані:

– *Множина анкет користувачів сайту з даними:* стать, вік, зріст, рівень доходу, освіта, професія, місто, хобі, сімейний стан.

Етап 1. Формування фрейму

Етап 2. Пошук комбінації параметрів, що задовольняють запит користувача

Крок 1. Вибір анкети користувача із БД.
Крок 2. Порівняння параметрів.

Етап 3. Формування результуючого списку кандидатів за критеріями пошуку

Крок 1. Сортування за відсотком співпадіння та обрання 5 кандидатів із найбільшим відсотком співпадіння;
Крок 2. Відображення списку знайдених кандидатів згідно проведеного пошуку;
Крок 3. Формування експертного висновку.

Вихідні дані

– *Список кандидатів*
– *Експертний висновок*

схема Способу оцінювання сумісності анкет.

- Система отримує фрейм користувача ідеалу.
- Робить вибірку усіх фреймів із БД.
- Перевіряє співпадіння критеріїв.
- Перевіряє відсоток співпадіння.
- Якщо відсоток співпадіння більший 50%, то система додає анкету у результуючий список. У іншому випадку розпочинається наступна ітерація циклу.
- Після опрацювання усіх анкет система сортує результуючий список за спаданням, та повертає 5 найкращих співпадінь, та експертні висновки про порівняння анкет.

Тестовий фрейм запит №1

Для тестування роботи розробленого фрейму створено два тестових фрейми запити.

Фрейм №1	
Вік	20-40
Зріст	140-170
Освіта	Середня освіта
Дохід	<1000$
Стать	Чоловік
Тип особистості	Екстраверт
Професія	Інвестиційний аналітик
Місто	Калуш
Хобі	В'язання

Дані фрейму :

1. Вік від 20 до 40 років
2. Зріст від 140 до 170
3. Середня освіта
4. <1000$
5. Чоловік
6. Екстраверт
7. Інвестиційний аналітик
8. Місто: Калуш
9. Хобі: В'язання

Результат порівняння по 1 тестовому фреймі

Ім'я	Прізвище	Вік	Зріст	Стать	Тип особистості	Дохід	Освіта	Місто	Хобі	Професія	%
Євген	Єрмаков	25	156	Чоловік	Екстраверт	<1000$	Середня освіта	Калуш	В'язання	Інвестиційний аналітик	100%
Вадім	Ігнатов	26	158	Чоловік	Інтроверт	<1000$	Середня освіта	Бориспіль		Інвестиційний аналітик	66%
Лук'ян	Гаврилов	24	151	Чоловік	Екстраверт	Не вказано	Середня освіта	Кам'янка-Дніпровська		Інвестиційний аналітик	66%
Василь	Ільїн	38	161	Чоловік	Не вказано	<1000$	Середня освіта	Калуш		Менеджер	66%
Іван	Сідоров	40	152	Чоловік	Екстраверт	>3000$	Середня освіта	Коломия	В'язання	Фармацевт	66%

Результатом запиту є наступні анкети користувачів:

1. Євген Єрмаков 100% співпадіння. Співпали усі критерії.
2. Вадім Ігнатов 66% співпадіння. Не співпали: Тип особистості, Місто та Хобі.
3. Лук'ян Гаврилов 66% співпадіння. Не співпали: Дохід, місто та хобі.
4. Василь Ільїн 66% співпадіння. Не співпали: Хобі, професія, тип особистості.
5. Іван Сідоров 66% співпадіння. Не співпали: Професія, дохід та місто.

Тестовий фрейм запит №2

Для тестування роботи розробленого фрейму створено два тестових фрейми запити.

Фрейм №2	
Вік	18-20
Зріст	140-150
Освіта	Вища освіта
Дохід	>3000$
Стать	Жінка
Тип особистості	Не вказано
Професія	Актор
Місто	Бердянськ
Хобі	Гра на музичних інструментах, В'язання, Комп'ютерні ігри, Гра у настільні ігри, Вивчення мов.

Дані фрейму :

1. Вік від 18 до 20 років
2. Зріст від 140 до 150
3. Вища освіта
4. >3000$
5. Жінка
6. Не вказано
7. Актор
8. Місто: Бердянськ
9. Хобі: Гра на музичних інструментах, В'язання, Комп'ютерні ігри, Гра у настільні ігри, Вивчення мов.

Результат порівняння по 2 тестовому фреймі

Ім'я	Прізвище	Вік	Зріст	Стать	Тип особистості	Дохід	Освіта	Місто	Хобі	Професія	%
Влада	Гребенюк	19	142	Жінка	Інтроверт	>3000$	Вища освіта	Луганськ	Вивчення мов	Рятувальник	57.41 %
Корнелія	Кравчук	57	142	Жінка	Інтроверт	>3000$	Вища освіта	Бердянськ	Немає співпадінь	Військова	55.56 %

Результатом запиту є наступні анкети користувачів:

1. Влада Гребенюк 57.41% співпадіння. Співпала по наступним критеріям: стать, вік, зріст, дохід, освіта. Також співпало одне хобі, це вивчення мов. Не співпали наступні критерії: професія та місто.
2. Корнелія Кравчук 66% співпадіння. Співпала по наступним критаріям: стать, зріст, дохід, місто, освіта. Не співпали наступні критерії: жодне із хобі, вік, професія.

Висновки

Під час виконання кваліфікаційної роботи бакалавра проаналізовано предметну область, проведено огляд та дослідження її особливостей. Також розглянуто застосування фреймового подання знань для задач класифікації та пошуку відповідності. Для подібних задач були проаналізовані існуючі рішення.

Розроблений метод можна покращити наступним способом, додаванням подвійного порівняння анкет. Тобто при запиті «користувача 1» система також порівнювала і його анкету із фреймом ідеалом каднидатів із високим відсотком співпадіння.

Також у подальшому можна розробити мобільну версію додатку.

Дякую за увагу!

UNICHECK

Ім'я користувача:
Кафедра КН

ID перевірки:
1015408404

Дата перевірки:
03.06.2023 20:04:29 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
03.06.2023 20:06:34 EEST

ID користувача:
100005671

Назва документа: **КН-19-1 Жук**

Кількість сторінок: **66** Кількість слів: **10104** Кількість символів: **75554** Розмір файлу: **1.94 MB** ID файлу: **1015071849**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

3.42%
Схожість

Найбільша схожість: **1.5%** з джерелом з Бібліотеки (ID файлу: **1014963757**)

3.06% Джерела з Інтернету	287	Сторінка 68
2.4% Джерела з Бібліотеки	88	Сторінка 70

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування	17 сторінок

Sat Jun 03 19:11:19 EEST 2023, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 7.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 11%**

	Документ		Сумарний збіг по Базі Даних	
ID: 114634 Назва: КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА Додано в БД: 2023-06-03 Автора: Д.І. Жук Керівники: О.В. Собко Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	56067	870	5059 (9%)	70 (8%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ КАФЕДРИ КОМП'ЮТЕРНИХ НАУК
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: *Спосіб оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств*

Автор: *студент групи КН-19-1 Жук Дмитро Іванович*

Спеціальність: *122 – Комп'ютерні науки*

Освітня програма: *освітньо-професійна*

Науковий керівник: *викладач Собко О.В.*

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	***відповідає***
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі Жука Д.І., не є плагіатом, оскільки: запозичення розміщені в розділі огляду існуючих підходів, не описують безпосередньо авторську роботу і не стосуються її результатів; усі запозичення фрагментарні; серед запозичень знаходяться загальновідомі терміни, скорочення та матеріали статей.

Обсяг запозичень, визначений системами виявлення збігів/ ідентичності/схожості, складає:

- за системою Anti-Plagiarism: 7%;

- за системою Unicheck: 3.42 %.

Керівник роботи _____ *Олена СОБКО*

Гарант ОП _____ *Олександр МАЗУРЕЦЬ*

Завідувач кафедри КН _____ *Олександр БАРМАК*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

ВІДГУК НАУКОВОГО КЕРІВНИКА

на кваліфікаційну роботу бакалавра

студента *гр. КН-19-1 Жука Дмитра Івановича*

за темою *Спосіб оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств*

1. Актуальність теми

Актуальним завданням, яке потребує аналізу і досліджується у даній роботі, є визначення ефективності застосування фреймового подання знань для оцінювання сумісності анкет користувачів соціальної інтелектуальної системи на прикладі сайту знайомств. Для ефективного використання автоматизованих систем необхідною є програмна реалізація способу оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств.

2. Відповідність роботи предметній області Стандарту спеціальності 122 Комп'ютерні науки

За описом предметної області, об'єктом дослідження є процес автоматизованого оцінювання сумісності анкет користувачів на прикладі сайту знайомств. Предметом дослідження є моделі, методи, алгоритми та засоби для визначення рівня сумісності анкет користувачів для соціальної інтелектуальної вебсистеми знайомств Метою роботи є розробка та програмна реалізація способу оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств. Під час вирішення даного завдання були використані математичні моделі, методи та алгоритми для розв'язання теоретичних і практичних проблем, які виникають при розробці інформаційних технологій. Виконана кваліфікаційна робота бакалавра відповідає стандарту бакалавра спеціальності 122 – Комп'ютерні науки.

3. Професійні та особистісні якості бакалавра

У ході роботи над кваліфікаційною роботою бакалавра Жук Дмитро Іванович проявив себе як дисциплінований студент, виконуючи всі етапи дослідження вчасно і належним чином. Як під час написання пояснювальної записки, так і при розробці прикладного програмного забезпечення, продемонстрував достатній рівень компетентності та результативності.

Жук Дмитро Іванович успішно опанував професійні навички в галузі "Комп'ютерні науки" і продемонстрував значні знання та вміння в розробці програмного забезпечення. Знання, що були набуті під час навчання, виявились на достатньо високому рівні, що сприяло досягненню успішних результатів у роботі.

4. Ступінь самостійності під час виконання кваліфікаційної роботи

Студент Жук Дмитро Іванович самостійно виконував усі поставлені завдання, тому результати роботи є його особистим надбанням.

5. Ступінь оволодіння методами дослідження

При реалізації кваліфікаційної роботи показав достатній рівень компетентностей та володіння необхідними інструментами, методами, методиками та технологіями предметної області комп'ютерних наук.

6. Повнота та якість розкриття теми роботи

Тема роботи розкрита повністю та цілком якісно, проаналізовано предметну область і відповідно до цього поставлено ряд завдань, що були повністю виконано.

7. Логічність, послідовність, аргументованість, літературна грамотність викладення матеріалу

Матеріал в роботі викладено логічно, послідовно та аргументовано. Літературна грамотність тексту роботи на високому рівні.

8. Можливість практичного застосування кваліфікаційної роботи бакалавра, окремих її частин

Застосування способу оцінювання сумісності анкет користувачів за фреймовим поданням знань на сайтах знайомств може позитивно вплинути на користувачів, забезпечуючи їм більш якісні можливості знайти потенційних партнерів, зіставляємих з їхніми інтересами та вимогами.

9. Висновок про можливість допуску кваліфікаційної роботи бакалавра до захисту, на яку оцінку заслуговує робота

*Враховуючи високий рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка «**відмінно**» .*

Керівник О. Собко *викладач каф. КН Олена СОБКО*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра

студента *гр. КН-19-1 Жука Дмитра Івановича*

за темою: *Спосіб оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств*

1. Актуальність обраної теми

Тема "Спосіб оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту знайомств" є актуальною, оскільки вона відповідає потребам зростаючої популярності онлайн-сервісів знайомств та покликана покращити користувацький досвід шляхом підвищення точності оцінювання сумісності між користувачами.

2. Повнота розкриття мети та завдань роботи

Мета та завдання роботи за темою «Спосіб оцінювання сумісності анкет користувачів за фреймовим поданням знань для соціальної інтелектуальної системи на прикладі сайту» знайомств розкриті повністю.

3. Зміст кожного розділу роботи

Розділ включає докладний огляд теоретичних та практичних аспектів, пов'язаних з предметною областю роботи. У розділі 2 проводиться огляд фреймового підходу до структурування та подання знань. Автор розглядає основні концепції та принципи фреймової моделі, яка дозволяє організовувати та систематизувати інформацію про користувачів. Також в цьому розділі спроєктовано інформаційну структуру системи та відповідну базу даних. У розділі 3 розкрито деталі архітектури системи, вказуючи на різні компоненти та їх функціональні можливості. Пояснено реалізацію алгоритму оцінювання сумісності між користувачами на основі фреймового подання знань. Описано процес взаємодії та порівняння фреймів користувачів для визначення ступеня сумісності. Також в даному розділі проаналізовано ефективність розробленого способу.

4. Оцінка розробленої інформаційної системи, її практична цінність

Важливим внеском автора є розробка способу оцінювання сумісності між користувачами на основі фреймового підходу. Алгоритм, запропонований у роботі, використовує зіставлення фреймів для визначення ступеня сумісності між користувачами. Він аналізує спільні фрейми, їхні значення та взаємодії, щоб зробити

припущення про потенційну сумісність між парами користувачів. Такий підхід дозволяє системі надавати рекомендації щодо сумісних співрозмовників або потенційних партнерів на сайті знайомств.

5. Якість оформлення кваліфікаційної роботи бакалавра

Кваліфікаційна робота студента Жука Дмитра Івановича оформлена згідно вимог до структури та нормоконтролю.

6. Недоліки кваліфікаційної роботи бакалавра

В ряді випадків у пояснювальній записці використовуються повністю терміни, скорочення для яких було вже введено у переліку посилань. Надто дрібний шрифт на деяких рисунках пояснювальної записки. Наведене не впливає на кінцевий результат виконання роботи.

7. Загальний висновок (допускається чи не допускається до захисту), та оцінка на яку заслуговує кваліфікаційна робота.

Враховуючи рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка « відмінно *».*

Рецензент Бедосенко К.А.