**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

**Факультет інформаційних технологій**

| | |
|---|---|
| **ПОГОДЖЕНО** | **ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ** |
| **Декан факультету** | **Завідувач кафедри комп'ютерних наук** |
| **_____ Ігор БОЛБОТ** | **_____ Белла ГОЛУБ** |
| (підпис) | (підпис) |
| «___» _____ 2026 р. | «___» _____ 2026 р. |

# БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

**на тему:**

**«**Програмне забезпечення з оцінки багато критеріального індексу ризику для вразливих груп населення**»**

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

**Гарант освітньої програми**
к.т.н., доцент _____ **Ганна ВАЙГАНГ**
(підпис)

**Керівник бакалаврської кваліфікаційної роботи** _____ **Андрій Британ**
науковий ступінь та вчене звання (підпис)

**Виконав** _____ **Ілля Шандра**
(підпис)

**Київ-2026**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

**Факультет інформаційних технологій**

**ЗАТВЕРДЖУЮ**
**Завідувач кафедри комп'ютерних наук**

к.т.н., доцент _____ Белла ГОЛУБ
(підпис)

«___» _____ 2025 р.

**ЗАВДАННЯ**
**ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ**
**ЗДОБУВАЧУ**

**Шандрі Іллі Анатолійовичу**
(прізвище, ім'я, по батькові)

Спеціальність «121 Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення»
Тема бакалаврської кваліфікаційної роботи
«Програмне забезпечення з оцінки багато критеріального індексу ризику для вразливих груп населення»

затверджена наказом від «19» грудня 2025 р. № 3196 «С»
Термін подання завершеної роботи на кафедру «22» травня 2026 р.
Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту): _____
Перелік питань, що підлягають дослідженню:
Аналіз предметної області та методів оцінки соціального ризику вразливих групнаселення, проєктування інформаційного забезпечення, Розробка програмного забезпечення, Рекомендації щодо впровадження та експлуатації системи.
Перелік графічних документів (за необхідності).
_____
Дата видачі завдання «19» грудня 2025 р.

**Керівник бакалаврської кваліфікаційної роботи** _____ (підпис) **АНДРІЙ БРИТАН**

**Завдання взяв до виконання** _____ (підпис) **ІЛЛЯ ШАНДРА**

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

ВГН – вразливі групи населення

ER – Entity-Relationship (сутність-зв'язок)

HTML – HyperText Markup Language (мова гіпертекстової розмітки)

CSS – Cascading Style Sheets (каскадні таблиці стилів)

JS – JavaScript

API – Application Programming Interface (прикладний програмний інтерфейс)

ASP.NET – веб-фреймворк від Microsoft

СУБД – система управління базами даних

SQL – Structured Query Language (мова структурованих запитів)

UML – Unified Modeling Language (уніфікована мова моделювання)

UI – User Interface (інтерфейс користувача)

MVC – Model-View-Controller (патерн проєктування)

PDF – Portable Document Format

DOCX – формат документів Microsoft Word

# ВСТУП

Актуальність теми. В умовах реформування системи соціального захисту України питання своєчасного виявлення та підтримки вразливих груп населення набуває особливої ваги. Демографічні зміни, наслідки збройного конфлікту та економічна нестабільність суттєво збільшили кількість осіб, які потребують соціального супроводу. За даними Міністерства соціальної політики України, понад 5 мільйонів громадян належать до вразливих категорій населення, що потребують регулярного моніторингу та адресної підтримки з боку соціальних служб.

Соціальні працівники нині змушені самостійно опрацьовувати великі обсяги неструктурованої інформації про підопічних, що призводить до суб'єктивності оцінок та нерівномірного розподілу ресурсів допомоги. Відсутність уніфікованих критеріїв оцінювання зумовлює ситуацію, за якої однакові соціальні випадки можуть отримувати принципово різний пріоритет залежно від конкретного фахівця, що їх розглядає. Це не лише знижує ефективність соціального захисту, але й ставить під сумнів справедливість розподілу обмежених ресурсів державної підтримки.

Відсутність єдиного автоматизованого інструменту для об'єктивної оцінки соціального ризику унеможливлює системний підхід до пріоритизації підтримки. Більшість наявних рішень або є вузькоспеціалізованими зарубіжними продуктами, що не адаптовані до українського законодавства та реалій соціальної роботи, або являють собою прості електронні таблиці без можливості автоматичного розрахунку та формування звітності. Жодне з проаналізованих рішень не забезпечує повного циклу роботи: від первинної реєстрації підопічного до автоматичного формування рекомендацій і друку офіційного висновку.

Розробка програмного забезпечення, що реалізує багатокритеріальну модель оцінки ризику, дозволяє стандартизувати цей процес, підвищити прозорість рішень і скоротити час на виявлення осіб, які потребують першочергової уваги. Впровадження методу зваженої суми з науково обґрунтованими коефіцієнтами вагомості для кожного критерію забезпечує відтворюваність результатів: за однакових вхідних даних система завжди формує ідентичний висновок незалежно від людського фактору. Це принципово важливо в контексті підзвітності соціальних служб та захисту прав підопічних.

Практична значущість розробленого застосунку підсилюється можливістю його масштабування на рівень районних та обласних соціальних служб. Уніфікований формат даних відкриває перспективи подальшої інтеграції з державними реєстрами, що дозволить перейти від ізольованої роботи окремих установ до єдиного інформаційного простору соціального захисту.

Мета розробки. Проєктування та програмна реалізація веб-застосунку «СоціоРизик» для багатокритеріальної оцінки індексу ризику вразливих груп населення із формуванням рекомендацій та звітів.

Для досягнення поставленої мети визначено такі завдання: провести системний аналіз предметної області соціального захисту та огляд існуючих програмних рішень; розробити математичну модель багатокритеріальної оцінки ризику; спроєктувати архітектуру системи та реляційну базу даних; реалізувати серверну та клієнтську частини застосунку; забезпечити розмежування прав доступу за ролями; реалізувати формування звітів; провести функціональне тестування.

Об'єктом розробки є процес оцінювання соціального ризику вразливих груп населення в умовах діяльності соціальних служб України.

Предметом розробки є методи, алгоритми та програмні засоби автоматизованої багатокритеріальної оцінки індексу ризику з підтримкою прийняття рішень.

Методи та технології. У роботі використано методи системного аналізу, об'єктно-орієнтованого проєктування та багатокритеріального аналізу рішень. Для математичного моделювання застосовано метод зваженої суми (Weighted Sum Model), що є одним із найбільш поширених і прозорих методів підтримки прийняття рішень у задачах з кількісно визначеними критеріями. Для реалізації системи обрано стек: ASP.NET Core (C#) для серверної частини, HTML5/CSS3/JavaScript для клієнтської частини, реляційна СУБД для зберігання даних. Взаємодія між клієнтом і сервером реалізована через REST API з JWT-автентифікацією. Проєктування виконано засобами UML-моделювання із застосуванням діаграм варіантів використання, класів, компонентів та розміщення.

Апробація. Основні результати роботи опубліковано у тезах конференції. Тези прийнято та опубліковано.

Структура записки. Пояснювальна записка складається з 4 розділів основної частини та містить 62 сторінки, 21 рисунок, 5 таблиць, 19 використаних джерел та 6 додатків. Розділ 1 – системний аналіз предметної області. Розділ 2 – інформаційне забезпечення. Розділ 3 – розробка програмного забезпечення. Розділ 4 – рекомендації щодо впровадження та експлуатації системи.

# 1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Опис предметної області

Вразливі групи населення – це особи/сім'ї, які мають найвищий ризик потрапляння у складні життєві обставини через вплив несприятливих зовнішніх та/або внутрішніх чинників. Таке визначення закріплене у Законі України «Про соціальні послуги» № 2671-VIII від 17.01.2019. Відповідно до цього ж закону, малозабезпечена особа – це особа, середньомісячний сукупний дохід якої за один квартал не перевищує двох прожиткових мінімумів.

Соціальний ризик – імовірність погіршення соціального стану особи внаслідок поєднання несприятливих факторів: матеріального становища, стану здоров'я, соціальної ізоляції, житлових умов та рівня особистої безпеки. Управління соціальним ризиком є ключовим завданням системи соціального захисту.

**Демографічна ситуація та масштаб проблеми.** Станом на 2023 рік чисельність населення на підконтрольній Україні території становила 31,5 млн осіб, причому на вікову категорію 60+ припадає 27%, тоді як діти до 18 років становлять лише 15% наявного населення.

За даними Державної служби статистики України, особи похилого віку (65 років і старіші) складають 17,64% населення країни. В абсолютних цифрах це понад 5,4 млн осіб, які потребують систематичного моніторингу стану їх соціального благополуччя.

Станом на 1 січня 2023 року в Україні налічувалося понад 2,7 млн осіб з інвалідністю, що становить 6,6% від загальної чисельності населення. Серед них – 222,3 тисячі осіб з інвалідністю першої групи, 900,8 тисячі – другої групи, 1 мільйон 416 тисяч – третьої групи, а також 163,9 тисячі дітей з інвалідністю.

Протягом 2023 року понад 230 тис. людей в Україні отримали статус особи з інвалідністю, з них 138,1 тис. чоловіків та 92,5 тис. жінок.

**Структура системи соціального захисту.** Відповідно до Закону № 2671-VIII, визначення потреб населення у соціальних послугах здійснюється на підставі соціально-демографічних даних про вразливі групи населення та результатів оцінювання потреб особи/сім'ї у соціальних послугах. Система надання соціальних послуг в Україні охоплює центри соціальних служб, територіальні центри соціального обслуговування, будинки-інтернати для громадян похилого віку та осіб з інвалідністю. Фахівці з соціальної роботи щодня збирають та обробляють інформацію про стан підопічних, складають індивідуальні плани підтримки та розподіляють ресурси між одержувачами послуг.

**Проблема суб'єктивності оцінювання.** Наявна практика оцінки потреб підопічних переважно спирається на досвід та інтуїцію конкретного соціального працівника. Відсутність єдиного формалізованого інструменту призводить до: нерівномірного розподілу ресурсів допомоги між підопічними; неможливості об'єктивного порівняння ступеня ризику між різними особами; відсутності документованої та відтворюваної методики, яку можна перевірити або оскаржити; значних витрат часу на ручну обробку даних. Саме для вирішення цих проблем розроблено систему «СоціоРизик», яка реалізує стандартизовану багатокритеріальну модель оцінки соціального ризику.

**Характеристика вразливих груп, що охоплюються системою.** У розробленій системі підтримуються чотири основні вразливі групи. *Люди похилого віку* – особи у віці 65 років і старші, що характеризуються підвищеним ризиком погіршення стану здоров'я, соціальної ізоляції та неспроможності самостійного ведення господарства. *Діти* – особи до 18 років, що перебувають у несприятливих сімейних або соціальних умовах і потребують захисту від

жорстокого поводження, бездоглядності або злиденності. *Особи з інвалідністю* – особи, у яких є стійке фізичне, розумове, інтелектуальне або сенсорне порушення, що у взаємодії з різними бар'єрами може заважати їхній повній участі в суспільстві. *Малозабезпечені громадяни* – особи або сім'ї, сукупний дохід яких не забезпечує задоволення базових життєвих потреб.

## 1.2 Аналіз вимог до програмної системи

На основі аналізу предметної області та консультацій з потенційними користувачами сформовано перелік функціональних та нефункціональних вимог до системи «СоціоРизик».

Функціональні вимоги:

- ведення бази даних підопічних з прив'язкою до вразливої групи;
- проведення оцінки ризику за п'ятьма зваженими критеріями;
- автоматичний розрахунок індексу ризику та класифікація рівня;
- формування персоналізованих рекомендацій за результатами оцінки;
- журнал оцінок з фільтрацією за групою та рівнем ризику;
- формування звітів у форматах PDF та DOCX;
- розмежування доступу за трьома ролями: соціальний працівник, аналітик, адміністратор.

Нефункціональні вимоги:

- веб-інтерфейс з підтримкою браузерів Chrome 90+, Firefox 88+, Edge 90+;
- час відповіді системи на запит не більше 2 секунд;
- захищене зберігання паролів (хешування bcrypt);
- адаптивний дизайн інтерфейсу (Bootstrap 5).

## 1.3 Моделювання предметної області

Для побудови ефективного програмного забезпечення важливо не лише розуміти логіку функціонування предметної області, але й вміти структурувати інформацію про неї у вигляді чітких моделей. Моделювання предметної області дозволяє описати її основні об'єкти, процеси, взаємозв'язки та сценарії використання. Це сприяє більш глибокому аналізу потреб користувачів, визначенню вимог до системи, а також уникненню непорозумінь між замовником і розробником.

У випадку системи оцінки соціального ризику моделювання дає змогу виявити ключових учасників взаємодії - соціальних працівників, аналітиків та адміністраторів, - а також описати послідовність дій у процесі проведення оцінки: від вибору підопічного до збереження результату та формування звіту.

Для опису предметної області використовується нотація UML (Unified Modeling Language) - уніфікована мова моделювання, яка дає можливість графічно відображати функціональні й логічні аспекти системи. У межах даного підрозділу наведено три типи UML-діаграм: діаграма прецедентів, діаграма послідовності та діаграма активності.

### 1.3.1 Діаграма прецедентів

Діаграма прецедентів (use case diagram) є одним із ключових інструментів моделювання функціональних вимог до системи. Вона дає змогу візуалізувати основних учасників процесів (акторів) і типові дії (прецеденти), які вони виконують у межах програмного забезпечення. Такий тип діаграм дозволяє сформувати цілісне уявлення про функціональність системи ще до початку її розробки.

Діаграма прецедентів є ефективним засобом для ідентифікації ключових процесів, що відбуваються в межах предметної області, а також для встановлення меж відповідальності між учасниками. Вона дозволяє ще на початковому етапі аналізу визначити, які саме функції виконує кожен із суб'єктів взаємодії та які дії ними ініціюються. Таким чином, діаграма прецедентів виступає як основа для подальшої деталізації сценаріїв роботи та розробки логіки програмної системи.

Основні елементи діаграми прецедентів включають акторів і прецеденти. Актори - це зовнішні суб'єкти, які взаємодіють з системою для досягнення певної мети. Прецеденти - це функціональні сценарії, що описують, яким чином система задовольняє потреби актора. Між прецедентами можуть існувати зв'язки: «include» (обов'язкове включення одного прецеденту до іншого) та «extend» (необов'язкове розширення поведінки).

На рис. 1.1 представлено діаграму прецедентів системи «СоціоРизик».

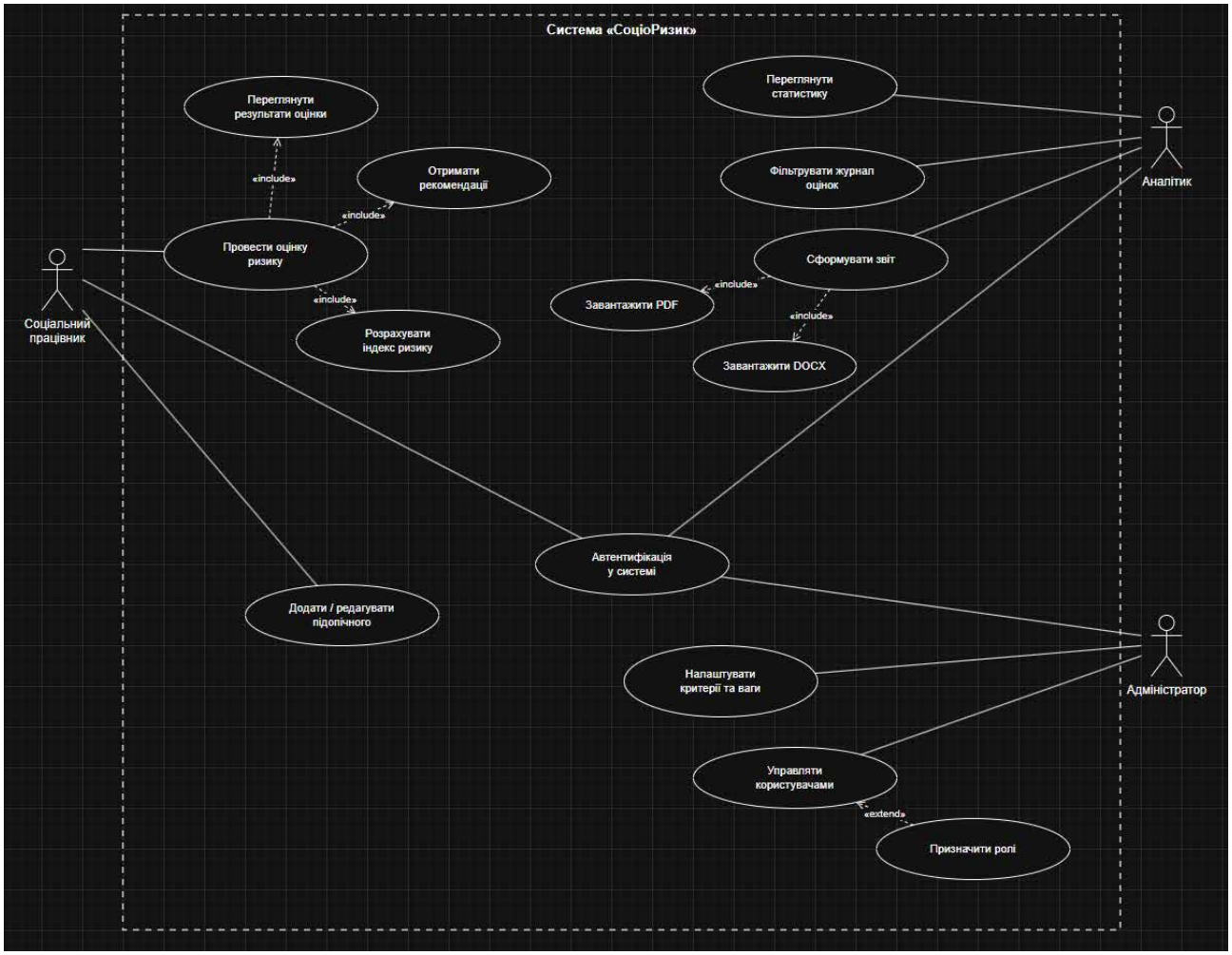



Рис. 1.1 - Діаграма прецедентів веб-застосунку «СоціоРизик»

**Опис акторів**

Виокремлено трьох ключових акторів, які взаємодіють із системою.

*Соціальний працівник* - основний користувач системи, що безпосередньо веде роботу з підопічними. Він здійснює реєстрацію нових підопічних, редагує їх дані, проводить оцінку ризику шляхом заповнення форми з критеріями та переглядає результати оцінок.

*Аналітик* - співробітник соціальної служби, відповідальний за аналіз загальної ситуації. Він переглядає статистику оцінок у розрізі груп та рівнів ризику, фільтрує журнал оцінок і формує звітні документи для керівництва.

*Адміністратор* - технічний фахівець, що забезпечує коректне функціонування системи. Він налаштовує критерії оцінювання та їх вагові коефіцієнти, управляє обліковими записами користувачів та призначає їм ролі.

**Опис прецедентів**

Автентифікація у системі - спільний для всіх акторів прецедент. Користувач вводить логін та пароль; система перевіряє облікові дані та надає доступ відповідно до призначеної ролі.

*Соціальний працівник:*

- додавання / редагування підопічного - введення або оновлення персональних даних особи, що перебуває на соціальному обліку, з прив'язкою до відповідної вразливої групи;
- проведення оцінки ризику - заповнення форми з п'ятьма критеріями та ініціювання розрахунку індексу; цей прецедент включає три підпорядкованих через зв'язок «include»: розрахунок індексу ризику, формування рекомендацій та перегляд результатів оцінки.

*Аналітик:*

- перегляд статистики - ознайомлення зі зведеними даними щодо розподілу підопічних за рівнями ризику та вразливими групами;
- фільтрація журналу оцінок - відбір записів за критеріями групи, рівня ризику або часового діапазону;
- формування звіту - генерація підсумкового документа; прецедент включає два підпорядкованих: завантаження у форматі PDF та завантаження у форматі DOCX.

*Адміністратор:*

- налаштування критеріїв та ваг - додавання, редагування або деактивація критеріїв оцінювання та зміна їх вагових коефіцієнтів;
- управління користувачами - створення, редагування та блокування облікових записів;
- призначення ролей - розширення прецеденту «управляти користувачами» через зв'язок «extend»: присвоєння або зміна ролі конкретного користувача.

Діаграма прецедентів демонструє загальну структуру функціональної взаємодії у системі «СоціоРизик» та дає змогу зрозуміти ролі учасників і межі відповідальності кожного з них перед переходом до детального проєктування.

### 1.3.2 Діаграма послідовності

Діаграма послідовності (sequence diagram) є одним із ключових інструментів моделювання мовою UML, що призначений для візуалізації динамічного аспекту системи. На відміну від статичних діаграм класів, вона акцентує увагу на часовій послідовності передачі повідомлень між об'єктами. У контексті розробки системи «СоціоРизик» дана діаграма дозволяє детально простежити шлях проходження даних від моменту ініціації запиту соціальним працівником до моменту остаточної фіксації результатів у базі даних.

Основними структурними компонентами діаграми є **лінії життя (lifelines)**, що представляють окремі модулі або сутності системи, та **повідомлення**, що відображають виклики методів або передачу даних. Синхронні процеси позначаються суцільними стрілками, тоді як повернення результату (відповідь) - пунктирними. Для моделювання складних алгоритмічних процесів використано комбіновані фрагменти: **loop** для ітераційної обробки критеріїв, що забезпечує гнучкість розрахунку незалежно від кількості вхідних параметрів.

Розроблена діаграма послідовності, наведена на рис. 1.2, деталізує сценарій «Провести оцінку ризику», який є центральним функціональним вузлом системи. В процесі взаємодії задіяно п’ять логічних компонентів:

Соціальний працівник - зовнішній актор, що ініціює процес.

Веб-інтерфейс (Браузер) - клієнтська частина, відповідальна за візуалізацію та збір даних.

AssessmentController - серверний компонент на базі ASP.NET Core, що координує потоки даних.

RiskCalculator - сервіс бізнес-логіки, де реалізовано математичний апарат оцінки.

БД (SQL Server) - рівень персистентності для надійного зберігання інформації.

Діаграма послідовності системи «СоціоРизик» представлена у Додатку А

**Опис послідовності взаємодії компонентів**

Процес розпочинається з ініціації запиту соціальним працівником на перехід до форми створення нової оцінки. Браузер надсилає відповідний запит до серверного контролера AssessmentController. Для забезпечення актуальності даних контролер звертається до бази даних, виконуючи запит на отримання структури критеріїв та доступних варіантів відповідей. Після отримання даних від SQL Server, контролер формує та повертає клієнтській частині динамічну HTML-форму, адаптовану під конкретні потреби оцінювання.

На наступному етапі, після безпосереднього заповнення анкети соціальним працівником, веб-інтерфейс агрегує введені дані та надсилає їх на сервер. Контролер, виступаючи в ролі посередника, передає масив обраних відповідей до спеціалізованого сервісу RiskCalculator.

Внутрішня логіка обробки даних у RiskCalculator реалізована через циклічний алгоритм (фрагмент loop). Для кожного обраного критерію виконується обчислення часткового показника з урахуванням вагових коефіцієнтів, що поступово формують загальний інтегральний індекс ризику. Після завершення циклу сервіс виконує низку внутрішніх операцій: проводить верифікацію отриманого індексу щодо встановлених нормативних порогів для визначення рівня вразливості та автоматично підбирає комплекс індивідуальних рекомендацій.

Сформований об'єкт із повним результатом оцінки повертається до контролера, який ініціює процедуру збереження. Дані розподіляються між таблицями бази даних (загальна інформація про оцінку та деталізовані результати за кожним пунктом). Після отримання підтвердження від бази даних, система виконує фінальну дію - перенаправлення користувача на сторінку результатів, де відображається вичерпна інформація про стан вразливої особи та запропонований план подальших дій.

### 1.3.3 Діаграма активності

Діаграма активності (activity diagram) є фундаментальним інструментом моделювання в межах мови UML, який використовується для детального опису динаміки бізнес-процесів або складних алгоритмічних операцій. На відміну від діаграми послідовності, де основна увага приділяється обміну повідомленнями між об’єктами, діаграма активності фокусується на логічній структурі потоку керування та послідовності виконання дій. Це дозволяє наочно представити не лише лінійні сценарії, а й умовні розгалуження, паралельні гілки виконання та точки синхронізації процесів.

Ефективність використання даної діаграми полягає у можливості абстрагуватися від технічної реалізації методів і зосередитись на загальному воркфлоу (workflow) системи. Ключовими елементами, використаними при побудові, є:

- Початковий вузол (акцептор), що ініціює старт процесу;
- Дії (activity nodes), які відображають атомарні кроки обробки інформації;
- Вузли прийняття рішень (decision nodes) у формі ромбів, що визначають подальший шлях виконання залежно від виконання певних логічних умов;
- Кінцевий вузол, який фіксує успішне або регламентоване завершення сценарію.

Окрему роль відіграють зони відповідальності (swimlanes), які дозволяють чітко розмежувати логіку функціонування між різними суб'єктами та модулями системи, що забезпечує прозорість розподілу функцій.

Розроблена діаграма активності описує комплексний бізнес-процес «Провести оцінку ризику» і наведена на рис. 1.3 (Додаток Б). Структурно діаграма розділена на три вертикальні зони відповідальності: соціальний працівник (користувач), інформаційна система «СоціоРизик» (програмна логіка) та база даних (рівень зберігання).

**Опис потоку активності та логічних переходів**

Життєвий цикл процесу розпочинається у зоні відповідальності соціального працівника з ініціації перегляду списку підопічних. Після ідентифікації та вибору конкретної особи, фахівець активує функцію проведення оцінки, що слугує сигналом для системи до підготовки робочого середовища.

Система «СоціоРизик» приймає запит та переходить до етапу динамічного формування інтерфейсу. Для цього здійснюється звернення до бази даних з метою отримання актуальних довідників: переліку критеріїв оцінки та відповідних варіантів відповідей з їхніми ваговими значеннями. Після успішного завантаження даних, веб-інтерфейс рендерить форму та відображає її соціальному працівнику.

Основний етап взаємодії передбачає введення соціальним працівником отриманих у ході інтерв'ю даних за п'ятьма ключовими категоріями: рівень матеріального доходу, функціональний стан здоров'я, ступінь соціальної ізоляції, якість житлових умов та рівень особистої безпеки. Після відправки форми система переходить до критичного вузла прийняття рішення «Дані валідні?». На цьому етапі виконується автоматизована верифікація:

Якщо виявлено некоректність або неповні дані, система генерує повідомлення про помилку, повертаючи потік управління до етапу редагування форми.

У разі успішної валідації, ініціюється внутрішній обчислювальний алгоритм.

Обчислювальний блок виконує розрахунок інтегрального індексу ризику, використовуючи метод зваженої суми. Отриманий результат зіставляється з еталонними пороговими значеннями для автоматичного визначення рівня вразливості (низький, середній або високий). Одночасно з цим сервіс формує перелік індивідуальних рекомендацій, фокусуючись на тих проблемних зонах, де було набрано максимальну кількість балів.

Завершальна фаза процесу передбачає синхронне виконання двох операцій: перманентне збереження результатів у базі даних та відображення звітної сторінки користувачу. Після ознайомлення з результатами, соціальний працівник має змогу прийняти рішення щодо необхідності документального оформлення оцінки (вузол «Сформувати звіт?»). При позитивному виборі система здійснює генерацію офіційного звіту у форматі PDF або DOCX, реєструє документ у таблиці Report та надає можливість його завантаження. Весь бізнес-процес завершується у кінцевому вузлі, забезпечуючи цілісність та збереженість усіх внесених змін.

## 1.4 Огляд інформаційних джерел та існуючих рішень за темою розробки

Невід'ємною частиною аналізу предметної області є дослідження вже наявних програмних рішень, що вирішують схожі або суміжні задачі. Такий огляд дозволяє, з одного боку, запозичити успішні підходи та уникнути відомих помилок, а з іншого - чітко визначити незаповнену нішу, яку має зайняти розроблюваний продукт. У межах даного підрозділу розглянуто чотири програмні системи, що використовуються в соціальній сфері для обліку підопічних або оцінки їхнього стану: Bonterra Apricot, Bonterra Penelope, Casebook та система SDQ (Strengths and Difficulties Questionnaire).

### 1.4.1 Bonterra Apricot (США)

Bonterra Apricot - це хмарна платформа для управління випадками (case management), розроблена компанією Social Solutions (нині Bonterra) та орієнтована на некомерційні організації й державні агенції у сфері соціальних послуг. Система забезпечує інтуїтивні робочі процеси, потужні функції автоматизації та розширені інструменти звітності для відстеження результатів та оптимізації програм.

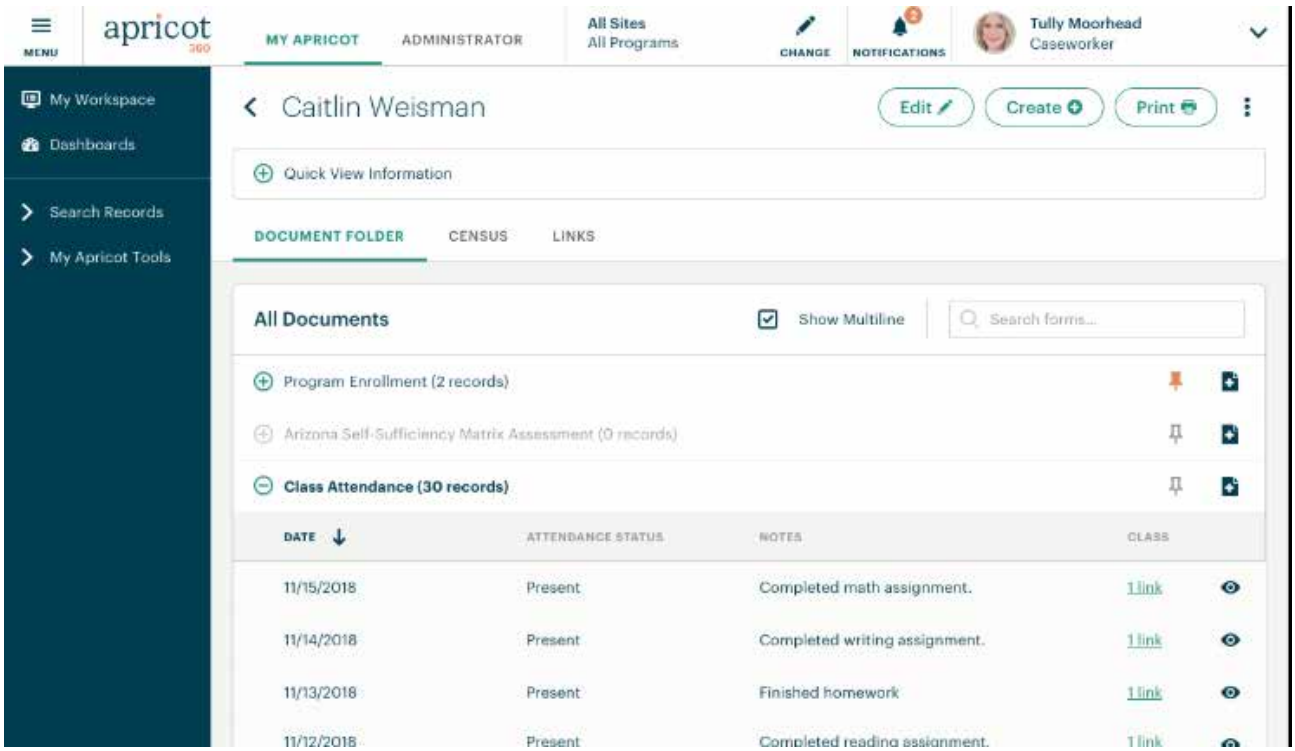


Рис. 1.2 - Інтерфейс системи Bonterra Apricot

Система підтримує широкий спектр типів полів для будь-якого набору даних: випадаючі списки, числові поля, текстові поля, поля розрахунків, демографічні дані, перемикачі та поля дати. Необмежене зв'язування записів дозволяє зіставляти дані за допомогою полів посилань для побудови звітів із складними структурами даних. Платформа розміщується у середовищі Amazon Web Services, що забезпечує захист, резервне копіювання та відповідність вимогам. Дозволи доступу на рівні форм і записів гарантують, що співробітники бачать лише необхідні дані клієнтів.

*Переваги:* розвинена система звітності та аналітики; гнучкі налаштовувані форми; підтримка ролей і дозволів; інтеграція з AWS; понад 90 000 користувачів у світі.

*Недоліки:* платформа є складною у впровадженні - процес реалізації передбачає щотижневі зустрічі з фахівцем протягом чотирьох тижнів, і перші два з них витрачаються лише на ознайомлення з функціоналом; відсутня українська локалізація; виключно комерційний продукт з непрозорим

ціноутворенням; не має вбудованого механізму багатокритеріальної оцінки ризику як самостійного модуля.

### 1.4.2 Casebook (США)

Casebook - це хмарна платформа для управління соціальними послугами, орієнтована на державні й некомерційні організації. Система пропонує інструменти для ведення справ, оцінки потреб клієнтів, моніторингу програм та формування звітності. На відміну від Apricot та Penelope, Casebook позиціонується як більш доступне за ціною рішення для організацій малого та середнього розміру та має стартову вартість від 20 доларів США на користувача на місяць.

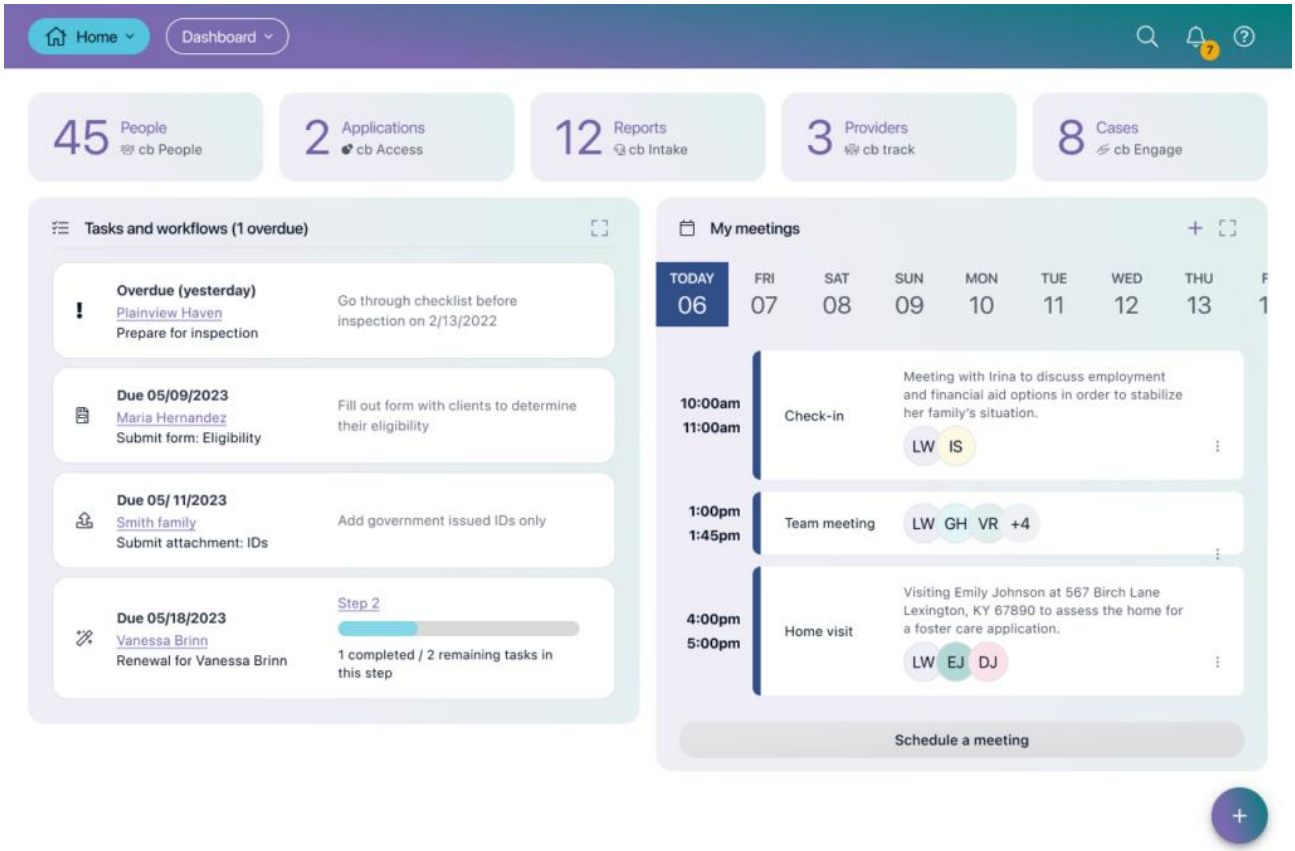



Рис. 1.3 - Інтерфейс системи Casebook

Casebook підтримує формування індивідуальних планів підтримки, ведення журналу взаємодій із клієнтом та базову звітність. Проте система не передбачає автоматизованого розрахунку кількісного індексу ризику на основі зважених критеріїв - оцінка здійснюється описово, без числового результату.

*Переваги:* відносно доступна ціна; зрозумілий інтерфейс; підходить для менших організацій; хмарне розгортання без потреби у власній інфраструктурі.

*Недоліки:* відсутня українська локалізація та адаптація до вітчизняного законодавства; немає кількісної моделі оцінки ризику; обмежені можливості налаштування критеріїв; виключно англомовний продукт; ціна недоступна для більшості вітчизняних соціальних служб.

### 1.4.3 Bonterra Penelope (Канада)

Penelope - це хмарна система управління випадками, розроблена канадською компанією Athena Software (нині входить до складу Bonterra). Система включає планування клієнтів, нотатки, автоматизовані робочі процеси, налаштовувані смарт-форми, SMS та email-нагадування, інструменти комунікації персоналу, більше 130 вбудованих звітів і односпрямовану передачу до MS Exchange.

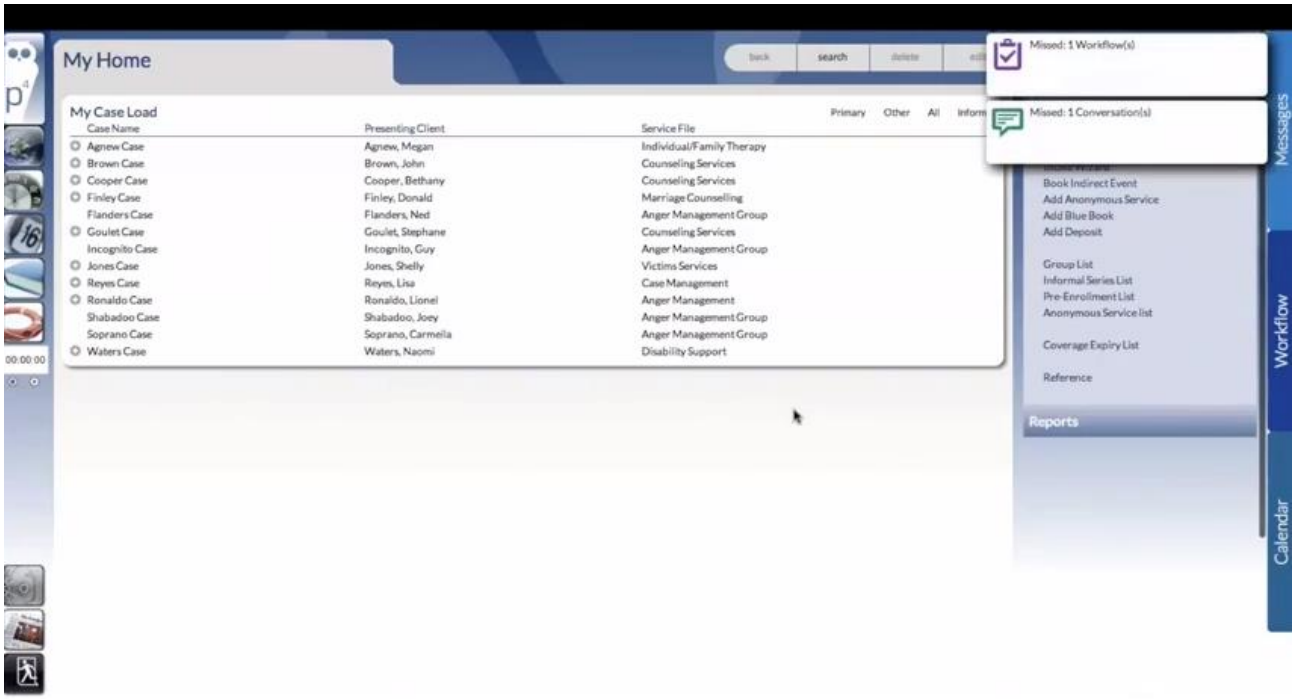



Рис. 1.4 - Інтерфейс системи Bonterra Penelope

Penelope довіряють національні урядові відомства, поліція та військові служби, університети, мережі постачальників соціальних послуг і середні та великі некомерційні організації по всьому світу. Система є повноцінною для документування зустрічей з клієнтами, оцінок, направлень, нотаток, семінарів і зборів агенції. Дані можна отримувати через ODBC-з'єднання та готувати їх для звітності та прогнозів при написанні грантів або моніторингу програм.

*Переваги:* широка функціональність для управління випадками; 130+ вбудованих звітів; налаштовувані смарт-форми; мобільна версія; інтеграція з MS Exchange та Tableau.

*Недоліки:* ціноутворення базується на моделі Named Active Users з мінімумом 25 користувачів та одноразовою платою за впровадження; відсутня українська локалізація; система орієнтована на великі організації та не підходить для малих соціальних служб; немає вбудованого модуля зваженої оцінки ризику.

Порівняльний аналіз існуючих рішень

За результатами огляду проведено порівняльний аналіз розглянутих систем за ключовими критеріями, що є визначальними для потреб вітчизняних соціальних служб. Результати наведено у табл. 1.1.

Таблиця 1.1 - Порівняльний аналіз існуючих програмних рішень

| Критерій | Apricot | Penelope | Casebook | «СоціоРизик» |
|---|---|---|---|---|
| Відкритість/ ліцензія | Комерційна | Комерційна | Комерційна | Власна розробка |
| Українська локалізація | Ні | Ні | Ні | Так |
| Багатокритеріальна оцінка ризику | Ні | Ні | Ні | Так |
| Числовий індекс ризику | Ні | Ні | Ні | Так |
| Підтримка ролей | Так | Так | Так | Так |
| Вбудована звітність PDF/DOCX | Частково | Частково | Ні | Так |
| Складність впровадження | Висока | Висока | Середня | Низька |
| Придатність для малих служб | Ні | Ні | Частково | Так |
| Адаптація до законодавства України | Ні | Ні | Ні | Так |

За результатами порівняльного аналізу встановлено, що жодне з розглянутих рішень не відповідає повною мірою потребам вітчизняних соціальних служб. Комерційні платформи Apricot, Penelope та Casebook є

потужними інструментами управління справами, однак вони не мають вбудованого механізму кількісної оцінки соціального ризику на основі зважених критеріїв. Жодна з них не локалізована для України та не адаптована до вітчизняного законодавства. Складність і вартість впровадження роблять їх недоступними для більшості соціальних служб громад.

Розроблюваний застосунок «СоціоРизик» заповнює виявлену нішу: він реалізує стандартизовану багатокритеріальну модель оцінки ризику з числовим індексом, підтримує чотири вразливі групи, забезпечує розмежування доступу за ролями, формування звітів та українськомовний інтерфейс - при цьому орієнтований на швидке розгортання в умовах малих соціальних служб без потреби у значних ресурсах впровадження.

## 1.5 Постановка завдання

На підставі проведеного аналізу предметної області, вимог та огляду існуючих рішень сформульовано завдання розробки системи «СоціоРизик»:

– реалізувати модель зваженої суми для розрахунку індексу ризику: risk_index = Σ(score_i × weight_i);
– підтримати чотири вразливі групи: люди похилого віку, діти, особи з інвалідністю, малозабезпечені;
– визначити п'ять критеріїв: рівень доходу (0,30), стан здоров'я (0,25), соціальна ізоляція (0,20), житлові умови (0,15), безпека (0,10);
– класифікувати ризик на 4 рівні: низький (0–0,30), середній (0,30–0,55), високий (0,55–0,75), критичний (0,75–1,0);
– реалізувати розмежування доступу за трьома ролями;
– забезпечити генерацію звітів у форматах PDF та DOCX;
– провести функціональне тестування системи.

# 2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

## 2.1 Логічна модель даних у вигляді ER-діаграми

Логічна модель даних відіграє ключову роль у процесі проєктування інформаційних систем, оскільки дозволяє формалізувати структуру зберігання інформації, визначити сутності предметної області та зв'язки між ними, забезпечуючи таким чином основу для створення фізичної структури бази даних. За допомогою ER-діаграми (Entity-Relationship diagram, діаграми «сутність-зв'язок») можна візуально відобразити цю логіку, що значно спрощує розуміння структури інформаційної системи на етапі проєктування [джерело].

У розробленій системі «СоціоРизик» були визначені такі ключові сутності: користувачі системи, вразливі групи населення, підопічні, критерії оцінювання, варіанти відповідей для критеріїв, оцінки ризику, деталізація оцінок та звіти. Кожна з цих сутностей має набір атрибутів, що ідентифікують її або описують відповідні характеристики. Для забезпечення зв'язків між сутностями використовуються зовнішні ключі, що дозволяє підтримувати цілісність і узгодженість даних у базі.

ER-діаграма розробленої логічної моделі наведена на рис. 2.1.

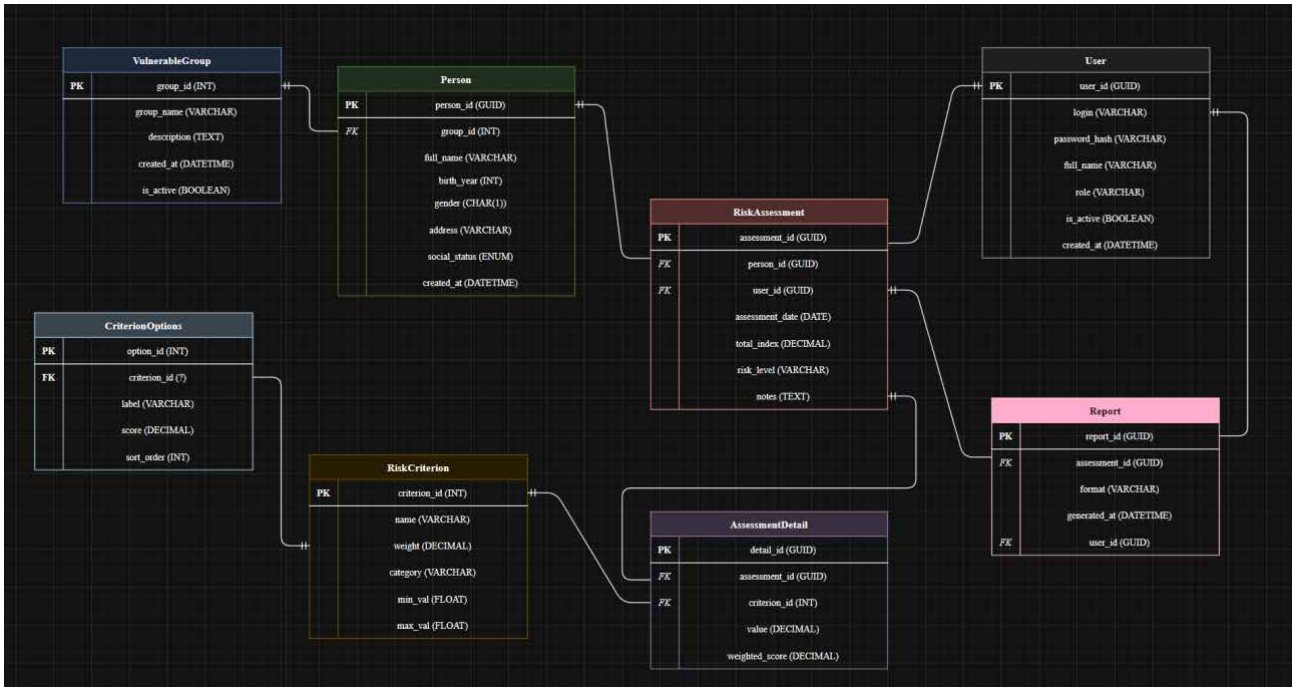



Рис. 2.1 - ER-діаграма логічної моделі даних системи «СоціоРизик»

Опис сутностей та атрибутів

**Users (Користувачі системи)** - базова сутність, що зберігає облікові записи всіх користувачів:

- UserId - унікальний ідентифікатор користувача (первинний ключ, автоінкремент);
- Login - унікальний логін для входу в систему (NVARCHAR(50), NOT NULL, UNIQUE);
- PasswordHash - хеш пароля у форматі bcrypt (NVARCHAR(256), NOT NULL);
- FullName - повне ім'я користувача (NVARCHAR(100), NOT NULL);
- Role - роль у системі: «SocialWorker», «Analyst» або «Admin» (NVARCHAR(20), NOT NULL, CHECK constraint);
- IsActive - ознака активності облікового запису (BIT, DEFAULT 1);
- CreatedAt - дата та час створення запису (DATETIME2, DEFAULT GETDATE()).

**VulnerableGroups (Вразливі групи)** - довідникова сутність, що класифікує підопічних:

- GroupId - унікальний ідентифікатор групи (первинний ключ);
- GroupName - назва вразливої групи (NVARCHAR(100), NOT NULL, UNIQUE);
- Description - опис характеристик групи (NVARCHAR(500), NULL).

**Persons (Підопічні)** - центральна сутність для зберігання даних про осіб, що перебувають на соціальному обліку:

- PersonId - унікальний ідентифікатор підопічного (первинний ключ);
- GroupId - посилання на вразливу групу (зовнішній ключ → VulnerableGroups);
- FirstName - ім'я підопічного (NVARCHAR(50), NOT NULL);
- LastName - прізвище підопічного (NVARCHAR(50), NOT NULL);
- DateOfBirth - дата народження (DATE, NOT NULL);
- Address - адреса проживання (NVARCHAR(200), NULL);
- Phone - контактний телефон (NVARCHAR(20), NULL);
- Notes - додаткові примітки (NVARCHAR(500), NULL);
- CreatedAt - дата реєстрації в системі (DATETIME2, DEFAULT GETDATE()).

**RiskCriterion (Критерії оцінювання)** - довідникова сутність, що визначає параметри багатокритеріальної оцінки:

- CriterionId - унікальний ідентифікатор критерію (первинний ключ);
- Name - назва критерію (NVARCHAR(100), NOT NULL, UNIQUE);
- Description - опис критерію (NVARCHAR(500), NULL);
- Weight - ваговий коефіцієнт критерію у діапазоні від 0 до 1 (DECIMAL(4,2), NOT NULL, CHECK constraint);

- IsActive - ознака активності критерію (BIT, DEFAULT 1);
- SortOrder - порядок відображення (INT, DEFAULT 0).

Важливою бізнес-вимогою є те, що сума вагових коефіцієнтів усіх активних критеріїв повинна дорівнювати 1,00. Наразі система містить п'ять критеріїв: рівень доходу (0,30), стан здоров'я (0,25), соціальна ізоляція (0,20), житлові умови (0,15), безпека (0,10).

**CriterionOptions (Варіанти відповідей)** - дочірня сутність, що містить конкретні варіанти для вибору в кожному критерії:

- OptionId - унікальний ідентифікатор варіанту (первинний ключ);
- CriterionId - посилання на критерій (зовнішній ключ → RiskCriterion, ON DELETE CASCADE);
- OptionText - текстовий опис варіанту (NVARCHAR(200), NOT NULL);
- Score - числове значення варіанту у діапазоні від 0,00 до 1,00, де 0 - найкращий стан, 1 - найгірший (DECIMAL(3,2), NOT NULL, CHECK constraint);
- SortOrder - порядок відображення варіантів (INT, DEFAULT 0).

**RiskAssessment (Оцінки ризику)** - ключова транзакційна сутність, що фіксує факт проведення оцінки:

- AssessmentId - унікальний ідентифікатор оцінки (первинний ключ);
- PersonId - посилання на підопічного (зовнішній ключ → Persons);
- UserId - посилання на соціального працівника, який проводив оцінку (зовнішній ключ → Users);
- RiskIndex - підсумковий індекс ризику від 0,00 до 1,00, розрахований за формулою зваженої суми (DECIMAL(4,2), NOT NULL);
- RiskLevel - рівень ризику: «Low», «Medium», «High» або «Critical» (NVARCHAR(20), NOT NULL, CHECK constraint);

- AssessedAt - дата та час проведення оцінки (DATETIME2, DEFAULT GETDATE());
- Notes - додаткові нотатки соціального працівника (NVARCHAR(1000), NULL).

**AssessmentDetail (Деталізація оцінки)** - дочірня сутність, що зберігає результат по кожному окремому критерію в межах оцінки:

- DetailId - унікальний ідентифікатор запису деталізації (первинний ключ);
- AssessmentId - посилання на оцінку (зовнішній ключ → RiskAssessment, ON DELETE CASCADE);
- CriterionId - посилання на критерій (зовнішній ключ → RiskCriterion);
- OptionId - посилання на обраний варіант відповіді (зовнішній ключ → CriterionOptions);
- Score - числове значення обраного варіанту на момент оцінки (DECIMAL(3,2), NOT NULL);
- WeightedScore - зважена оцінка (Score × Weight критерію), що є внеском у підсумковий індекс (DECIMAL(4,3), NOT NULL);
- UNIQUE constraint на парі (AssessmentId, CriterionId) - кожен критерій оцінюється лише один раз у межах однієї оцінки.

**Report (Звіти)** - сутність для зберігання метаданих про згенеровані звіти:

- ReportId - унікальний ідентифікатор звіту (первинний ключ);
- AssessmentId - посилання на оцінку, за якою сформовано звіт (зовнішній ключ → RiskAssessment, ON DELETE CASCADE);
- UserId - посилання на користувача, що ініціював генерацію (зовнішній ключ → Users);

- Format - формат звіту: «PDF» або «DOCX» (NVARCHAR(10), NOT NULL, CHECK constraint);
- FilePath - шлях до збереженого файлу на сервері (NVARCHAR(500), NOT NULL);
- GeneratedAt - дата та час генерації (DATETIME2, DEFAULT GETDATE()).

Опис зв'язків

Одна вразлива група може охоплювати багатьох підопічних, але кожен підопічний належить рівно до однієї групи (зв'язок «один до багатьох»). Зв'язок реалізовано через зовнішній ключ GroupId у таблиці Persons, що посилається на первинний ключ GroupId у таблиці VulnerableGroups.

Один критерій може мати багато варіантів відповідей, але кожен варіант належить лише до одного критерію (зв'язок «один до багатьох»). Зв'язок реалізовано через зовнішній ключ CriterionId у таблиці CriterionOptions з ON DELETE CASCADE - при видаленні критерію автоматично видаляються всі його варіанти.

Один підопічний може мати багато оцінок ризику з часом, але кожна оцінка належить рівно одному підопічному (зв'язок «один до багатьох»). Зв'язок реалізовано через зовнішній ключ PersonId у таблиці RiskAssessment.

Один користувач (соціальний працівник) може проводити багато оцінок, але кожна оцінка виконана лише одним користувачем (зв'язок «один до багатьох»). Зв'язок реалізовано через зовнішній ключ UserId у таблиці RiskAssessment.

Одна оцінка деталізується по кількох критеріях, і кожен запис деталізації належить рівно до однієї оцінки (зв'язок «один до багатьох»). Зв'язок

реалізовано через зовнішній ключ AssessmentId у таблиці AssessmentDetail з ON DELETE CASCADE.

Одна оцінка може мати кілька звітів у різних форматах (PDF та DOCX), але кожен звіт належить рівно до однієї оцінки (зв'язок «один до багатьох»). Зв'язок реалізовано через зовнішній ключ AssessmentId у таблиці Report.

Перевірка відповідності нормальним формам

Щоб забезпечити ефективне зберігання, обробку та уникнення надлишковості даних, важливо привести модель до форми, що відповідає принципам нормалізації. Нормалізація - це процес організації атрибутів та відношень у базі даних таким чином, щоб зменшити дублювання та забезпечити цілісність. Основними нормальними формами є перша (1НФ), друга (2НФ) і третя (3НФ). У розробленій моделі дотримано всіх трьох форм, що підтверджується наведеним нижче аналізом.

**Перша нормальна форма (1НФ).** Перша нормальна форма вимагає, щоб усі атрибути таблиць були атомарними - не ділилися на підчастини та не містили множинних значень. У розробленій моделі кожен стовпець кожної таблиці містить лише одне неподільне значення. Наприклад, таблиця Persons зберігає ім'я та прізвище підопічного в окремих полях FirstName та LastName, а не в одному загальному полі. Таблиця AssessmentDetail не містить масивів чи списків критеріїв - кожен запис відображає оцінку рівно одного критерію в межах однієї оцінки. У кожній таблиці визначено первинний ключ, що забезпечує унікальність записів. Відсутність вкладених списків або повторювальних груп атрибутів підтверджує дотримання першої нормальної форми.

**Друга нормальна форма (2НФ).** Друга нормальна форма вимагає, щоб усі неключові атрибути були повністю функціонально залежними від усього

первинного ключа, а не від його частини. Це правило актуальне для таблиць зі складеними первинними ключами. У розробленій моделі єдина таблиця з потенційно складеним ключем - AssessmentDetail, де унікальність запису визначається парою (AssessmentId, CriterionId). Атрибути Score та WeightedScore залежать одночасно від обох складових - від конкретної оцінки та конкретного критерію, а не від кожного з них окремо. Решта таблиць мають прості первинні ключі (один стовпець), тому до них вимога 2НФ виконується автоматично. Відповідність другій нормальній формі підтверджена.

**Третя нормальна форма (3НФ).** Третя нормальна форма виключає транзитивні залежності - ситуації, коли неключовий атрибут залежить не безпосередньо від первинного ключа, а через інший неключовий атрибут. У розробленій моделі це правило також дотримано. Розглянемо на конкретних прикладах. У таблиці RiskAssessment поле RiskLevel залежить безпосередньо від AssessmentId через значення RiskIndex, а не через будь-який інший атрибут. Дані про вразливу групу (GroupName, Description) не зберігаються в таблиці Persons - вони винесені до окремої таблиці VulnerableGroups і пов'язані зовнішнім ключем. Аналогічно, текстовий опис критерію та його ваговий коефіцієнт не дублюються в таблиці AssessmentDetail - замість цього зберігається лише зовнішній ключ CriterionId та зафіксоване на момент оцінки значення Score. Такий підхід унеможливлює транзитивну залежність: якщо адміністратор змінить ваговий коефіцієнт критерію, це не вплине на архів попередніх оцінок, оскільки WeightedScore було збережено в момент розрахунку.

Таким чином, логічна модель даних системи «СоціоРизик» повністю відповідає вимогам трьох нормальних форм. Це дозволяє мінімізувати обсяг надлишкової інформації, уникнути логічних аномалій під час оновлення чи видалення даних, а також забезпечує узгодженість і зручність подальшої роботи з базою даних.

## 2.2 Діаграма пакетів

Організаційна та архітектурна структура програмного забезпечення є одним із найбільш критичних аспектів проєктування складних інформаційних систем. Вона безпосередньо визначає не лише поточний стан системи, а й її потенціал до масштабування, супроводу, модифікації та тестування. Для ефективного управління складністю розробки та формування чітких меж між різними функціональними шарами застосовується декомпозиція.

Діаграма пакетів (package diagram) є спеціалізованим різновидом структурних діаграм мови UML, яка призначена для високорівневого відображення архітектурного устрою системи у вигляді укрупнених логічних блоків - пакетів, а також взаємозв'язків між ними. На відміну від деталізованих діаграм класів, цей інструмент дозволяє абстрагуватися від конкретної програмної реалізації та оцінити архітектуру системи як сукупність автономних підсистем. Зв'язки між пакетами на схемі реалізовані у вигляді спрямованих стрілок із ключовим стереотипом <<import>>, що вказує на спрямованість залежностей та архітектурні обмеження. Використання даної діаграми є критично важливим для превентивного аналізу показників зачеплення (coupling) та зв'язності (cohesion) модулів ще на етапі проєктування системи, запобігаючи виникненню циклічних залежностей (spaghetti-architecture).

Архітектурне рішення системи оцінки багатокритеріального ризику «СоціоРизик» базується на принципах багатошарової архітектури (Layered/Clean Architecture), що тісно переплікається з класичним патерном декомпозиції інтерфейсу. Такий підхід забезпечує неухильне виконання фундаментальних принципів об'єктно-орієнтованого проєктування SOLID, зокрема:

Принципу єдиної відповідальності (Single Responsibility Principle, SRP) - кожен пакет ізолює в собі виключно логіку свого рівня (візуалізація, обчислення, збереження даних);

Принципу інверсії залежностей (Dependency Inversion Principle, DIP) - взаємодія між високорівневими та низькорівневими компонентами здійснюється через слабке зчеплення, де шари оперують абстракціями, мінімізуючи пряму залежність від конкретних реалізацій.

Розроблена діаграма пакетів інформаційної системи «СоціоРизик» наведена на рис. 2.2.

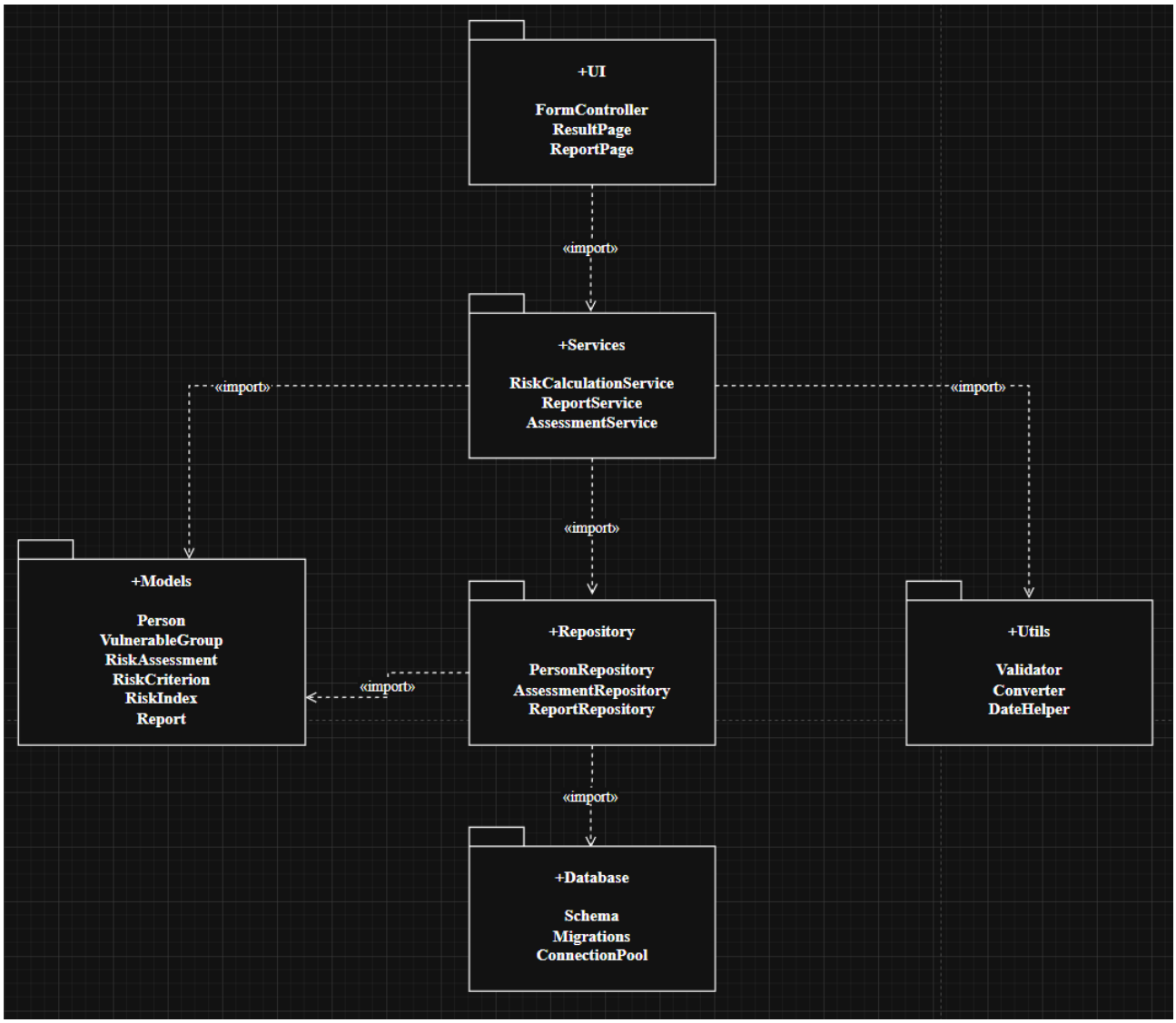



Рис. 2.2 - Діаграма пакетів системи «СоціоРизик»

У ході проєктування внутрішньої структури системи було виділено та згруповано шість базових пакетів, кожен з яких виконує суворо окреслену роль у загальному життєвому циклі обробки даних.

**Опис функціональних пакетів системи:**

Пакет +UI (User Interface) - представляє зовнішній клієнт-серверний шар системи, відповідальний за взаємодію з кінцевим користувачем (соціальним працівником). До його складу входять компоненти керування логікою відображення: FormController (контролер підготовки та обробки інтерактивних анкет), ResultPage (модуль візуалізації підсумкових показників ризику) та ReportPage (інтерфейсний блок керування звітністю). Даний шар є модулем верхнього рівня; він ініціює процеси та імпортує можливості пакету +Services для передачі користувацького введення на обробку.

Пакет +Services - є центральним ядром системи, де зосереджена вся ключова бізнес-логіка та аналітичні алгоритми комплексу. Пакет містить класи: RiskCalculationService (реалізація математичного апарату зваженого оцінювання), ReportService (модуль координації процесів документообігу) та AssessmentService (загальна оркестрація та управління сесіями оцінювання). Пакет +Services має високий ступінь зв'язності та імпортує пакети +Models, +Repository та +Utils для виконання розрахунків та передачі команд на збереження.

Пакет +Repository - реалізує класичний архітектурний патерн Repository, виступаючи ізолюючим прошарком між бізнес-логікою та рівнем безпосереднього доступу до сховища даних. Він містить класи: PersonRepository, AssessmentRepository та ReportRepository. Головне завдання цього пакету - абстрагувати вищі шари від специфіки побудови баз даних чи особливостей ORM-систем. Він безпосередньо взаємодіє з моделями даних та імпортує інфраструктурні можливості пакету +Database.

Пакет +Models - є фундаментальним доменом системи, що містить описи інформаційних сутностей (доменних моделей), якими оперують усі інші

компоненти програмного комплексу. Сюди інтегровано класи: Person (картка особи), VulnerableGroup (категорії вразливості), RiskAssessment (загальний запис оцінки), RiskCriterion (критерії аналізу), RiskIndex (параметри розрахованих показників) та Report (метадані сформованих документів). Характерною рисою цього пакету є його повна автономність - він не має жодних зовнішніх імпортів (<<import>>), що робить його "чистим" ядром системи згідно з канонами чистої архітектури.

Пакет +Utils (Utilities) - забезпечує систему допоміжним інструментарієм наскрізного функціонування. До його складу входять компоненти: Validator (забезпечення коректності та очищення вхідних соціологічних даних), Converter (трансформація форматів об'єктів) та DateHelper (робота з часовими мітками та періодами моніторингу). Цей шар надає утиліти для пакету +Services.

Пакет +Database - представляє найнижчий інфраструктурний рівень системи, що відповідає за фізичне і логічне структурування збереженої інформації. Він включає підсистеми Schema (опис реляційної структури), Migrations (модулі контролю версій та оновлення бази даних) та ConnectionPool (механізм оптимізації та утримання каналів зв'язку зі сховищем). Залежить від зовнішнього сервера баз даних.

## 2.3 Вибір системи управління базами даних

Вибір системи управління базами даних (СУБД) є одним із фундаментальних технічних рішень при розробці інформаційної системи, оскільки він безпосередньо впливає на продуктивність, надійність, масштабованість та складність підтримки програмного забезпечення. Неправильний вибір СУБД на етапі проєктування може призвести до значних витрат на переробку системи у майбутньому [джерело].

### 2.3.1 Обґрунтування вибору СУБД

Для вибору оптимального рішення проведено порівняльний аналіз трьох реляційних СУБД, що є найбільш поширеними у веб-розробці на платформі .NET: Microsoft SQL Server, PostgreSQL та SQLite. Результати порівняння наведено у табл. 2.1.

Таблиця 2.1 - Порівняльний аналіз СУБД

| **Критерій** | **MS SQL Server** | **PostgreSQL** | **SQLite** |
|---|---|---|---|
| Ліцензія (безкоштовна версія) | Express Edition | Відкрита (MIT) | Відкрита |
| Підтримка EF Core | Офіційний провайдер Microsoft | Npgsql (сторонній) | Офіційний провайдер Microsoft |
| Транзакції та ACID | Повна підтримка | Повна підтримка | Обмежено (без WAL за замовч.) |
| Конкурентний доступ | Так | Так | Ні (file lock) |
| Збережені процедури | Так (T-SQL) | Так (PL/pgSQL) | Ні |
| Інтеграція з Visual Studio | Вбудована (SQL Server Object Explorer) | Через плагін | Через плагін |
| Простота розгортання | Середня | Середня | Висока |
| Придатність для багатокористувацького веб-застосунку | Так | Так | Ні |

SQLite виключено на першому ж етапі аналізу з принципових міркувань: ця СУБД є файловою системою зберігання та не підтримує повноцінний конкурентний доступ кількох користувачів одночасно. Оскільки система

«СоціоРизик» передбачає паралельну роботу кількох соціальних працівників, SQLite є категорично непридатною.

Вибір між SQL Server та PostgreSQL ґрунтувався на таких аргументах на користь SQL Server. По-перше, провайдер Microsoft.EntityFrameworkCore.SqlServer розробляється безпосередньо командою Microsoft разом із самим EF Core, що гарантує максимальну сумісність, своєчасну підтримку нових версій та найкращу продуктивність. По-друге, SQL Server має вбудовану інтеграцію з Visual Studio через SQL Server Object Explorer та безкоштовний інструмент SQL Server Management Studio (SSMS), що суттєво прискорює розробку та налагодження запитів. По-третє, безкоштовна версія SQL Server Express підтримує бази даних до 10 ГБ, що цілком достатньо для потреб системи «СоціоРизик». По-четверте, T-SQL має дещо зручніший синтаксис для написання збережених процедур та тригерів порівняно з PL/pgSQL для розробників, що вже мають досвід роботи з Microsoft-стеком.

Таким чином, для реалізації системи «СоціоРизик» обрано Microsoft SQL Server 2019 Express.

### 2.3.2 Підключення до бази даних через Entity Framework Core

Для взаємодії з базою даних обрано ORM-фреймворк Entity Framework Core (EF Core) версії 8.0 - офіційний об'єктно-реляційний маппер від Microsoft для платформи .NET. EF Core дозволяє розробнику працювати з базою даних через об'єкти C# без написання SQL-запитів вручну, використовуючи підхід Code First, де схема бази даних автоматично генерується з класів-сутностей.

Основні переваги EF Core у контексті системи «СоціоРизик»: система міграцій дозволяє відслідковувати зміни схеми БД у системі контролю версій Git; підтримка LINQ-запитів дозволяє писати типобезпечні запити до бази

даних мовою C#; механізм відстеження змін (change tracking) автоматично визначає, які об'єкти потрібно оновити в БД; вбудована підтримка транзакцій спрощує реалізацію атомарних операцій збереження оцінки разом із деталями.

Рядок підключення до бази даних налаштовано у файлі конфігурації appsettings.json, а реєстрація контексту в DI-контейнері виконується у файлі Program.cs. Клас AppDbContext успадковує DbContext та містить набори DbSet<TEntity> для кожної таблиці бази даних.

## 2.4 Розробка структури інформаційної бази

У процесі створення інформаційної системи особливу увагу приділено розробці структури інформаційної бази, оскільки саме вона є основою зберігання, обробки та взаємодії з даними. Ретельно спроектована структура забезпечує логічну цілісність, ефективність виконання запитів та можливість масштабування у майбутньому.

Структура бази даних системи «СоціоРизик» містить не лише основні таблиці, а й збережену процедуру, тригер та налаштованих користувачів із розмежованими правами доступу. Такий підхід дозволяє автоматизувати обробку даних, підвищити рівень безпеки системи та забезпечити контроль цілісності інформації на рівні СУБД.

### 2.4.1 Створення бази даних

Першим кроком є створення бази даних з кириличним зіставленням символів. На рис. 2.4 наведено SQL-код ініціалізації бази даних «SocioRyzyk».

```
IF DB_ID('SocioRyzyk') IS NOT NULL
    DROP DATABASE SocioRyzyk;
GO

CREATE DATABASE SocioRyzyk
    COLLATE Cyrillic_General_CI_AI;
GO
```

Рис. 2.3 - SQL-код створення бази даних «SocioRyzyk»

Параметр COLLATE Cyrillic_General_CI_AI забезпечує коректне сортування та порівняння рядків із кириличними символами. Після виконання цього скрипту можна переходити до формування таблиць та інших об'єктів бази даних.

### 2.4.2 Створення таблиць

Усі вісім таблиць бази даних створено за допомогою команди CREATE TABLE з визначенням типів даних, обмежень NOT NULL, CHECK-обмежень та зовнішніх ключів. На рис. 2.4 наведено фрагмент SQL-коду створення центральної таблиці RiskAssessment, а на рис. 2.5 - таблиці AssessmentDetail, що деталізує оцінку по кожному критерію.

```
CREATE TABLE RiskAssessment (
    AssessmentId  INT IDENTITY(1,1) PRIMARY KEY,
    PersonId      INT NOT NULL,
    UserId        INT NOT NULL,
    RiskIndex     DECIMAL(4,2)  NOT NULL
                  CONSTRAINT CK_Assess_Index CHECK (RiskIndex >= 0 AND RiskIndex <= 1),
    RiskLevel     NVARCHAR(20)  NOT NULL
                  CONSTRAINT CK_Assess_Level CHECK (RiskLevel IN ('Low','Medium','High','Critical')),
    AssessedAt    DATETIME2     NOT NULL DEFAULT GETDATE(),
    Notes         NVARCHAR(1000) NULL,
    CONSTRAINT FK_Assessment_Person FOREIGN KEY (PersonId)
        REFERENCES Persons(PersonId),
    CONSTRAINT FK_Assessment_User FOREIGN KEY (UserId)
        REFERENCES Users(UserId)
);
GO
```

Рис. 2.4 - SQL-код створення таблиці RiskAssessment

```
CREATE TABLE AssessmentDetail (
    DetailId      INT IDENTITY(1,1) PRIMARY KEY,
    AssessmentId  INT NOT NULL,
    CriterionId   INT NOT NULL,
    OptionId      INT NOT NULL,
    Score         DECIMAL(3,2) NOT NULL,
    WeightedScore DECIMAL(4,3) NOT NULL,
    CONSTRAINT FK_Detail_Assessment FOREIGN KEY (AssessmentId)
        REFERENCES RiskAssessment(AssessmentId)
        ON DELETE CASCADE,
    CONSTRAINT FK_Detail_Criterion FOREIGN KEY (CriterionId)
        REFERENCES RiskCriterion(CriterionId),
    CONSTRAINT FK_Detail_Option FOREIGN KEY (OptionId)
        REFERENCES CriterionOptions(OptionId),
    CONSTRAINT UQ_Detail_AssessmentCriterion UNIQUE (AssessmentId, CriterionId)
);
GO
```

Рис. 2.5 - SQL-код створення таблиці AssessmentDetail

Повний код створення всіх таблиць та індексів наведено в Додатку Б.

# 3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка програмного забезпечення є центральним етапом створення інформаційної системи, що передбачає проєктування архітектури застосунку, визначення структури класів та їх взаємодії, вибір технологічного стеку та безпосередню реалізацію функціональних модулів. У цьому розділі описано всі ключові аспекти розробки веб-застосунку «СоціоРизик»: від визначення абстракцій предметної області до алгоритмів розрахунку індексу ризику та програмування модулів.

## 3.1 Абстракції предметної області

Абстракції предметної області є першим кроком у переході від концептуального розуміння задачі до конкретної програмної реалізації. Кожна абстракція відповідає ключовому поняттю предметної області та представляється у вигляді класу з набором атрибутів і методів. Правильне виокремлення абстракцій дозволяє уникнути надлишкового зв'язування між модулями та спрощує подальшу підтримку системи [джерело].

У системі «СоціоРизик» виокремлено такі основні абстракції.

User (Користувач) - представляє обліковий запис особи, що працює з системою. Містить атрибути: ідентифікатор, логін, хеш пароля, повне ім'я, роль (SocialWorker, Analyst, Admin), ознака активності. Методи: автентифікація, перевірка ролі.

VulnerableGroup (Вразлива група) - довідникова абстракція, що класифікує підопічних: люди похилого віку, діти, особи з інвалідністю, малозабезпечені. Містить назву та опис групи.

Person (Підопічний) - центральна сутність системи, що представляє особу, яка перебуває на соціальному обліку. Атрибути: ідентифікатор, посилання на вразливу групу, ПІБ, дата народження, адреса, телефон, нотатки.

RiskCriterion (Критерій оцінювання) - визначає один вимір оцінки соціального ризику. Атрибути: назва, опис, ваговий коефіцієнт, ознака активності, порядок відображення.

CriterionOption (Варіант відповіді) - конкретний варіант для вибору в межах критерію, що несе числове значення від 0,00 до 1,00. Пов'язаний з конкретним критерієм.

RiskAssessment (Оцінка ризику) - фіксує факт проведення оцінки конкретного підопічного конкретним соціальним працівником. Містить підсумковий індекс, рівень ризику, дату та нотатки.

AssessmentDetail (Деталь оцінки) - деталізує оцінку в розрізі одного критерію: зберігає обраний варіант, його числове значення та зважену оцінку.

RiskResult (Результат розрахунку) - об'єкт передачі даних (DTO), що повертається сервісом RiskCalculator: містить індекс ризику, рівень та список рекомендацій. Не зберігається в базі даних безпосередньо.

Report (Звіт) - метадані згенерованого документа: посилання на оцінку, формат (PDF/DOCX), шлях до файлу, дата генерації.

На рис. 3.1 наведено діаграму абстракцій предметної області.

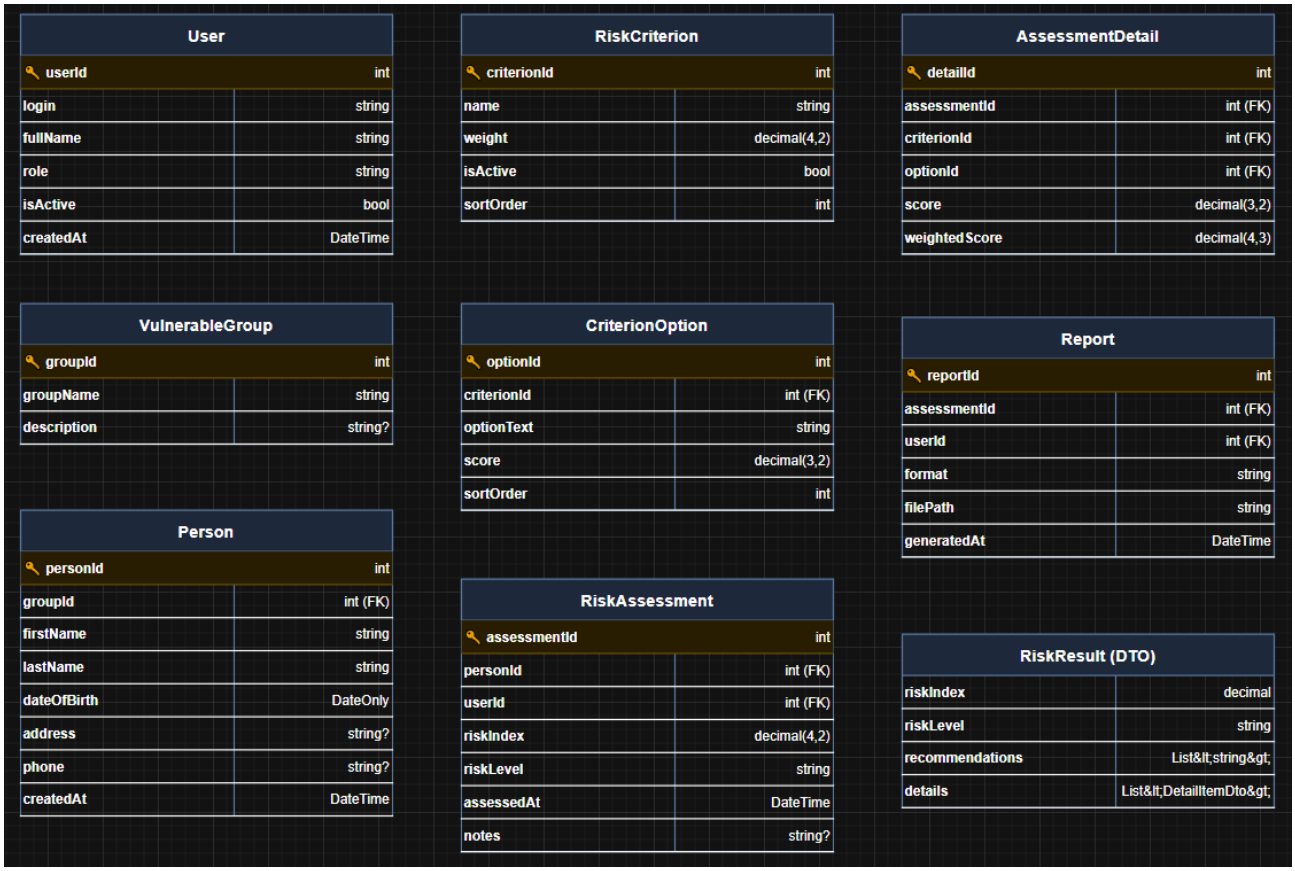



Рис. 3.1 - Абстракції предметної області системи «СоціоРизик»

## 3.2 Моделювання структури та взаємодії об'єктів

На основі визначених абстракцій розроблено діаграму класів з асоціаціями та виділено кооперації - групи взаємодіючих класів, що разом реалізують окрему функціональність системи.

### 3.2.1 Діаграма класів

Діаграма класів відображає статичну структуру системи: класи, їх атрибути, методи та зв'язки між ними. Між класами визначено такі типи зв'язків: асоціація (загальний зв'язок між класами), агрегація (ціле-частина без жорсткої залежності) та залежність (один клас використовує інший). На рис. 3.2 наведено загальну діаграму класів системи «СоціоРизик».

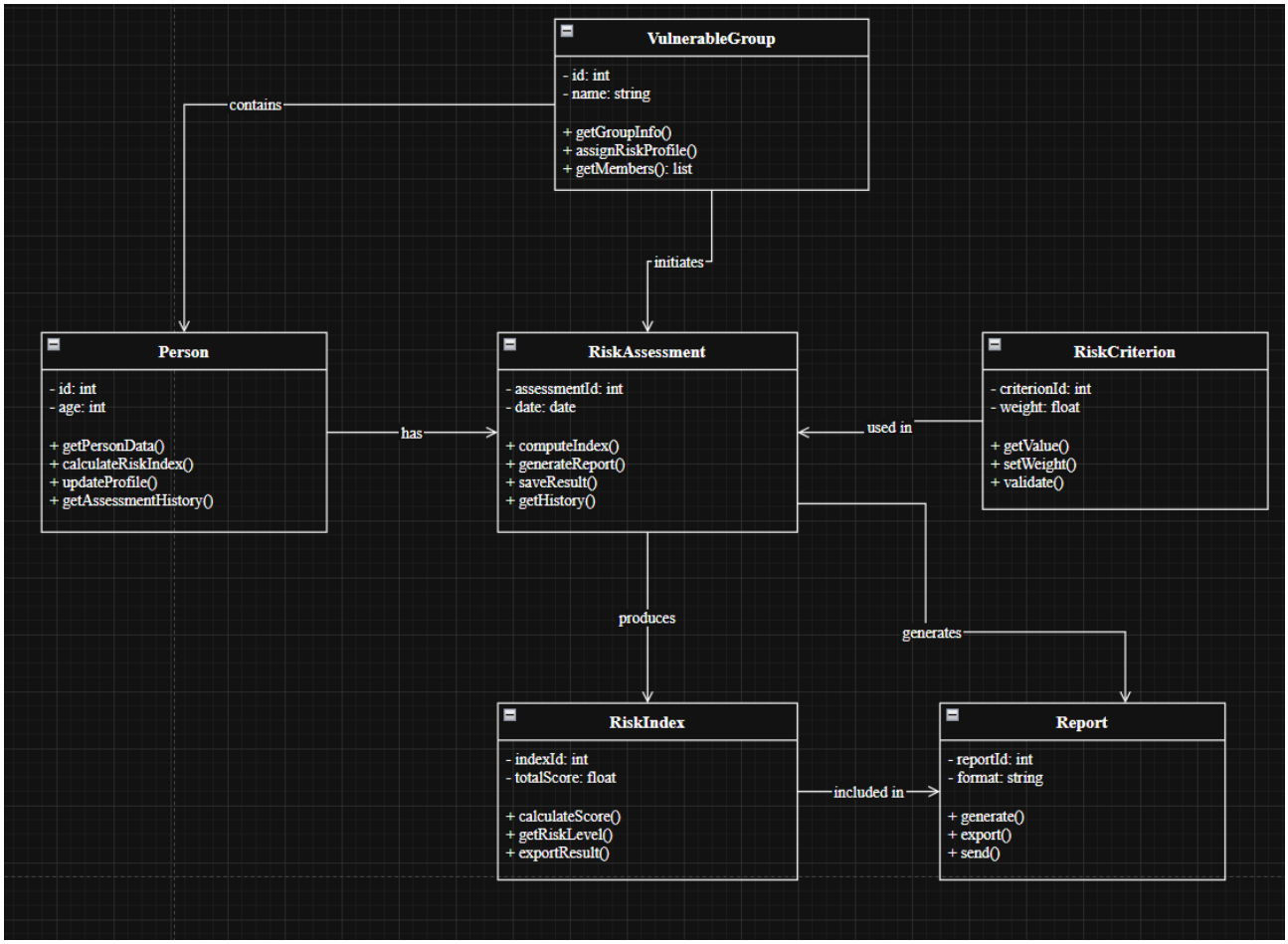



Рис. 3.2 - Діаграма класів системи «СоціоРизик»

### 3.2.2 Кооперація 1: Оцінка ризику

Перша кооперація об'єднує класи, що реалізують основний бізнес-процес системи - проведення оцінки соціального ризику. До неї входять: AssessmentController, AssessmentService, RiskCalculator, IAssessmentRepository, AssessmentRepository.

AssessmentController приймає HTTP-запит із заповненою формою оцінки та делегує обробку до AssessmentService. Сервіс викликає RiskCalculator для обчислення індексу ризику та рекомендацій, після чого через інтерфейс IAssessmentRepository зберігає результат у базі даних. Конкретну реалізацію доступу до даних забезпечує AssessmentRepository. На рис. 3.3 наведено діаграму цієї кооперації

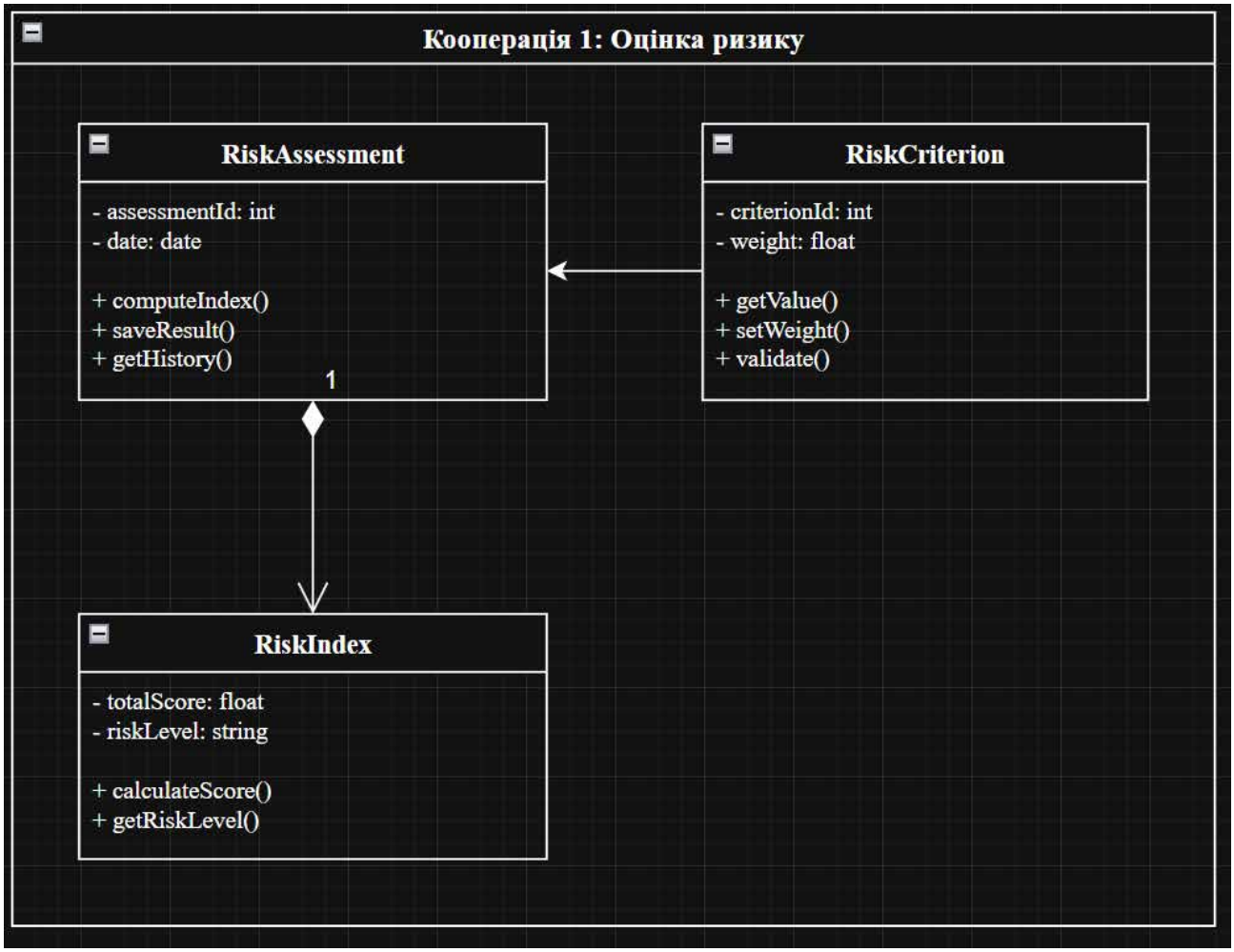



Рис. 3.3 - Діаграма кооперації: проведення оцінки ризику

### 3.2.3 Кооперація 2: Управління особами

Друга кооперація охоплює функціональність роботи з підопічними: PersonController, PersonService, PersonRepository. PersonController обробляє запити на додавання, редагування та пошук підопічних. PersonService містить бізнес-логіку валідації даних та фільтрації. PersonRepository реалізує доступ до таблиці Persons через EF Core. На рис. 3.4 наведено діаграму цієї кооперації.

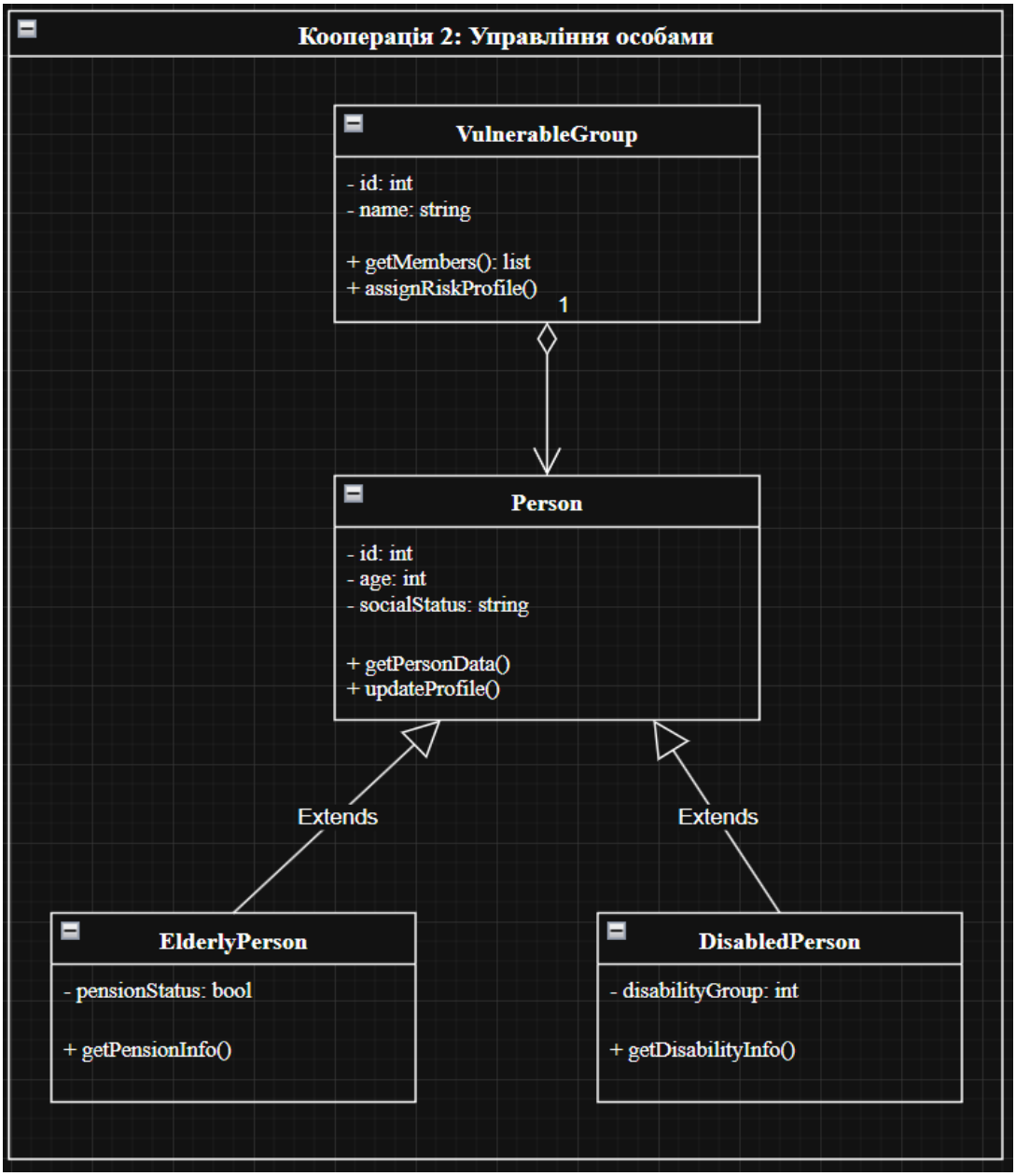



Рис. 3.4 - Діаграма кооперації: управління підопічними

### 3.2.4 Кооперація 3: Формування звіту

Третя кооперація реалізує генерацію звітів у різних форматах. До неї входять: ReportController, ReportGenerator, IReportGenerator, PdfBuilder, DocxBuilder, ReportRepository. ReportGenerator використовує патерн Стратегія: залежно від запитаного формату він делегує побудову документа до PdfBuilder або DocxBuilder, обидва з яких реалізують інтерфейс IReportGenerator. Після

генерації файл зберігається на сервері, а метадані - у таблиці Report. На рис. 3.5 наведено діаграму цієї кооперації.

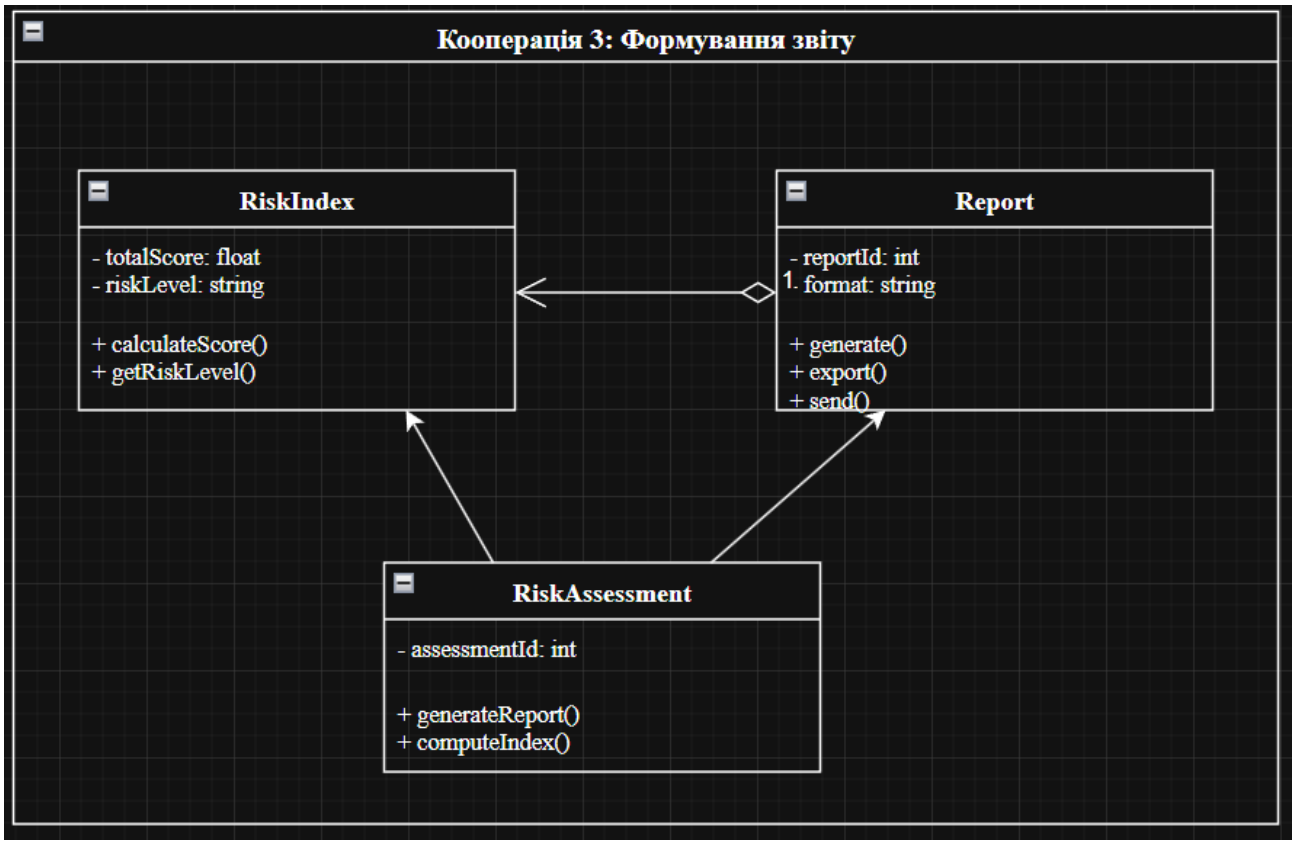



Рис. 3.5 - Діаграма кооперації: формування звіту

## 3.3 Вибір інструментарію для створення прикладного програмного забезпечення

Вибір інструментів розробки є важливим рішенням, що впливає на продуктивність розробника, підтримуваність коду та можливості розгортання системи. Нижче обґрунтовано вибір кожного з ключових компонентів технологічного стеку.

ASP.NET Core 8.0 (C#) обрано як основу серверної частини. Платформа є кросплатформеною (Windows та Linux), має вбудований DI-контейнер, підтримує Razor Pages для серверного рендерингу, а також містить вбудовану

систему автентифікації ASP.NET Identity з підтримкою ролей. Порівняно з альтернативами (Django, Laravel, Spring Boot), ASP.NET Core має найтіснішу інтеграцію з Entity Framework Core та SQL Server, що є визначальним для даного стеку.

HTML5, CSS3, JavaScript + Bootstrap 5 використано для клієнтської частини. Відмова від SPA-фреймворків (React, Angular, Vue) обґрунтована меншою складністю розгортання та підтримки - Razor Pages генерують HTML на сервері, що не потребує окремого API-шару та спрощує архітектуру для команди з одного розробника.

Entity Framework Core 8.0 - ORM для доступу до бази даних. Детальне обґрунтування вибору наведено у підрозділі 2.3.

Serilog використано для структурованого логування. Serilog записує події у структурованому форматі (JSON), що дозволяє фільтрувати та аналізувати логи. Налаштовано два «приймачі» (sinks): запис у файл (rolling file з ротацією по добі) та вивід у консоль під час розробки. Логуються: факти автентифікації (успішний вхід, невдала спроба), проведення кожної оцінки (PersonId, UserId, RiskIndex), помилки та необроблені винятки.

QuestPDF використано для генерації PDF-звітів. Бібліотека є безкоштовною для некомерційного використання, має fluent API та не залежить від зовнішніх компонентів на відміну від iTextSharp.

DocumentFormat.OpenXml (SDK від Microsoft) використано для генерації DOCX-звітів. Бібліотека є офіційною, безкоштовною та не має залежностей від Microsoft Office.

Git + GitHub використано як систему контролю версій. Міграції EF Core зберігаються у репозиторії, що дозволяє відтворити схему БД на будь-якому середовищі командою dotnet ef database update.

Visual Studio 2022 Community використано як IDE. Містить вбудований SQL Server Object Explorer, відладчик, профайлер та підтримку EF Core міграцій.

## 3.4 Алгоритмізація та програмування програмних модулів

Алгоритмізація програмних модулів полягає у формалізації бізнес-логіки системи у вигляді алгоритмів перед їх програмною реалізацією. У системі «СоціоРизик» ключовими з точки зору алгоритмічної складності є два модулі: розрахунок індексу ризику та генерація рекомендацій.

### 3.4.1 Алгоритм розрахунку індексу ризику

Основу математичної моделі системи становить метод зваженої суми (Weighted Sum Model, WSM) - один із найпоширеніших методів багатокритеріального аналізу рішень [джерело]. Індекс ризику обчислюється за формулою:

$risk\_index = \Sigma\ (score\_i \times weight\_i),\ де\ i = 1..n$

де score_i - числове значення обраного варіанту відповіді для і-го критерію (від 0,00 до 1,00), weight_i - ваговий коефіцієнт і-го критерію, сума всіх ваг дорівнює 1,00, n - кількість активних критеріїв.

Отриманий індекс класифікується на чотири рівні ризику: низький (0,00–0,30), середній (0,30–0,55), високий (0,55–0,75) та критичний (0,75–1,00).

На рис. 3.9 наведено реалізацію методу Calculate класу RiskCalculator.

```
public RiskResult Calculate(List<(RiskCriterion Criterion, CriterionOption Option)> items)
{
    decimal riskIndex = 0;
    var details = new List<DetailItemDto>();

    foreach (var (criterion, option) in items)
    {
        var weighted = Math.Round(option.Score * criterion.Weight, 3);
        riskIndex += weighted;

        details.Add(new DetailItemDto
        {
            CriterionName = criterion.Name,
            Weight = criterion.Weight,
            OptionText = option.OptionText,
            Score = option.Score,
            WeightedScore = weighted
        });
    }

    riskIndex = Math.Round(riskIndex, 2);

    var level = DetermineLevel(riskIndex);
    var recommendations = GenerateRecommendations(items);

    return new RiskResult
    {
        RiskIndex = riskIndex,
        RiskLevel = level,
        Recommendations = recommendations,
        Details = details
    };
}
```

Рис. 3.6 - Реалізація методу Calculate класу RiskCalculator

### 3.4.2 Алгоритм генерації рекомендацій

Після розрахунку індексу система формує перелік рекомендацій на основі значень окремих критеріїв. Алгоритм перебирає всі деталі оцінки та для кожного критерію, чий score перевищує порогове значення 0,50, додає відповідну рекомендацію з довідника. Рекомендації сортуються за спаданням зваженого балу критерію - найкритичніші проблеми виводяться першими.

На рис. 3.7 наведено реалізацію методу GenerateRecommendations.

```
private List<string> GenerateRecommendations(
    List<(RiskCriterion Criterion, CriterionOption Option)> items)
{
    return items
        .Where(x => x.Option.Score > Threshold)
        .OrderByDescending(x => x.Option.Score * x.Criterion.Weight)
        .Where(x => _recommendations.ContainsKey(x.Criterion.CriterionId))
        .Select(x => _recommendations[x.Criterion.CriterionId])
        .ToList();
}
```

Рис. 3.7 - Реалізація методу GenerateRecommendations

### 3.4.3 Програмування ключових модулів

У Додатку Д наведено реалізацію методу CreateAssessment сервісного класу AssessmentService, що оркеструє весь процес: валідація вхідних даних, виклик RiskCalculator, збереження результату через репозиторій у єдиній транзакції.

На рис. 3.8 наведено приклад методу контролера AssessmentController, що обробляє POST-запит на проведення оцінки, виконує валідацію ModelState та повертає результат у форматі JSON.

```
// POST /api/assessments
[HttpPost]
[Authorize(Roles = "SocialWorker,Admin")]
Ссылок: 0
public async Task<IActionResult> Create(AssessmentCreateDto dto)
{
    if (!ModelState.IsValid) return BadRequest(ModelState);

    var result = await assessmentService.CreateAsync(dto, CurrentUserId);
    return CreatedAtAction(nameof(GetById), new { id = result.AssessmentId }, result);
}
```

Рис. 3.8 - Реалізація обробника POST-запиту в AssessmentController

# 4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

## 4.1 Тестування системи

Тестування програмного забезпечення системи оцінки багатокритеріального ризику для вразливих груп населення «СоціоРизик» проводилося методом функціонального тестування («чорного ящика»). Цей метод дозволяє перевірити відповідність поведінки системи заявленим функціональним вимогам без аналізу внутрішньої структури вихідного коду.

Головною метою етапу верифікації є виявлення можливих логічних помилок при розрахунках, перевірка коректності обмежень доступу та забезпечення стійкості системи при роботі з користувацькими даними. Для кожного тестового випадку (тест-кейсу) було чітко визначено: унікальний ідентифікатор, опис цільового сценарію, набір вхідних даних, очікуваний результат згідно з технічним завданням, а також фактично отриманий результат.

Результати проведених випробувань та верифікації модулів інформаційної системи зведені в таблиці 4.1.

Таблиця 4.1 - Результати проведених випробувань та верифікації модулів інформаційної системи

| ІД | Опис тестового сценарію | Вхідні дані | Очікуваний результат | Фактичний результат | Статус |
|---|---|---|---|---|---|
| TC-01 | Розрахунок мінімального граничного | Усі відповіді з мінімальною | $riskIndex = 0.00$; Рівень: «Низький» | $riskIndex = 0.00$; Рівень: «Низький» | Успішно |

| | | | | | |
|---|---|---|---|---|---|
| | значення індексу ризику | вагою (Score = 0) | | | |
| ТС-02 | Розрахунок нижньої межі середнього рівня ризику | Набір відповідей, що дає суму балів 0.30 | $riskIndex = 0.30$; Рівень: «Середній» | $riskIndex = 0.30$; Рівень: «Середній» | Успішно |
| ТС-03 | Розрахунок межі переходу до високого рівня ризику | Набір відповідей, що дає суму балів 0.55 | $riskIndex = 0.55$; Рівень: «Високий» | $riskIndex = 0.55$; Рівень: «Високий» | Успішно |
| ТС-04 | Розрахунок критичного значення індексу ризику | Набір відповідей, що дає суму балів 0.75 | $riskIndex = 0.75$; Рівень: «Критичний» | $riskIndex = 0.75$; Рівень: «Критичний» | Успішно |
| ТС-05 | Розрахунок максимального граничного значення індексу ризику | Усі відповіді з максимальною вагою (Score = 1) | $riskIndex = 1.00$; Рівень: «Критичний» | $riskIndex = 1.00$; Рівень: «Критичний» | Успішно |
| ТС-06 | Перевірка обмеження доступу Соціального працівника до | Перехід за адресою /Admin/Criteria під обліковим | Перенаправлення на сторінку відмови у доступі (HTTP 403 Forbidden) | Доступ заблоковано, код 403 | Успішно |

| | | | | | |
|---|---|---|---|---|---|
| | адмін-панелі | записом працівника | | | |
| ТС-07 | Перевірка прав доступу Аналітика на зміну конфігурації критеріїв | Натискання кнопки «Редагувати вагу критерію» під роллю Аналітика | Кнопка заблокована, при спробі прямого запиту - помилка доступу | Дія заблокована інтерфейсом та сервером | Успішно |
| ТС-08 | Створення нової картки підопічного (Внесення особи) | Заповнення форми: ПІБ, дата народження, категорія вразливості, статус | Створення запису в таблиці Person, присвоєння унікального PersonId, успіх | Запис додано, користувача перенаправлено до профілю | Успішно |
| ТС-09 | Перевірка повного доступу Адміністратора до модифікації вагових коефіцієнтів | Зміна коефіцієнта критерію безпеки з 0.2 на 0.3 під роллю Адміна | Дані успішно оновлені в БД, виведено повідомлення «Конфігурацію збережено» | Коефіцієнт змінено, записи в БД оновлено | Успішно |
| ТС-10 | Генерація звітного документа за результатами | Натискання кнопки «Експорт у PDF» на | Формування та автоматичне завантаження валідного | Файл завантажено, структура та шрифти | Успішно |

| | | | | | |
|---|---|---|---|---|---|
| | оцінки у форматі PDF | сторінці результатів особи | файлу .pdf із таблицями | коректні | |
| ТС -11 | Генерація звітного документа у форматі DOCX | Натискання кнопки «Експорт у Word» на сторінце результатів особи | Формування та автоматичне завантаження файлу .docx зі збереженням форматування | Файл сформовано, структура відповідає шаблону | Успішно |

Для підтвердження працездатності та коректності виконання зафіксованих тестових сценаріїв на рисунках 4.1–4.2 наведено графічні протоколи верифікації ключових модулів системи «СоціоРизик».

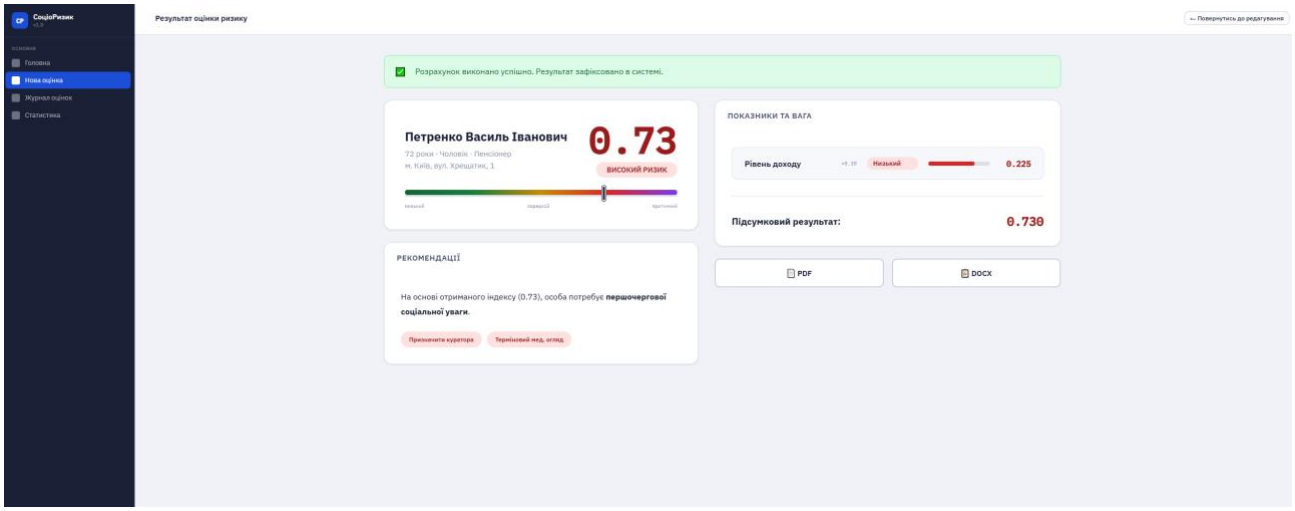



Рис. 4.1 - Графічний інтерфейс системи під час успішного розрахунку індексу ризику (Тест-кейс ТС-03)

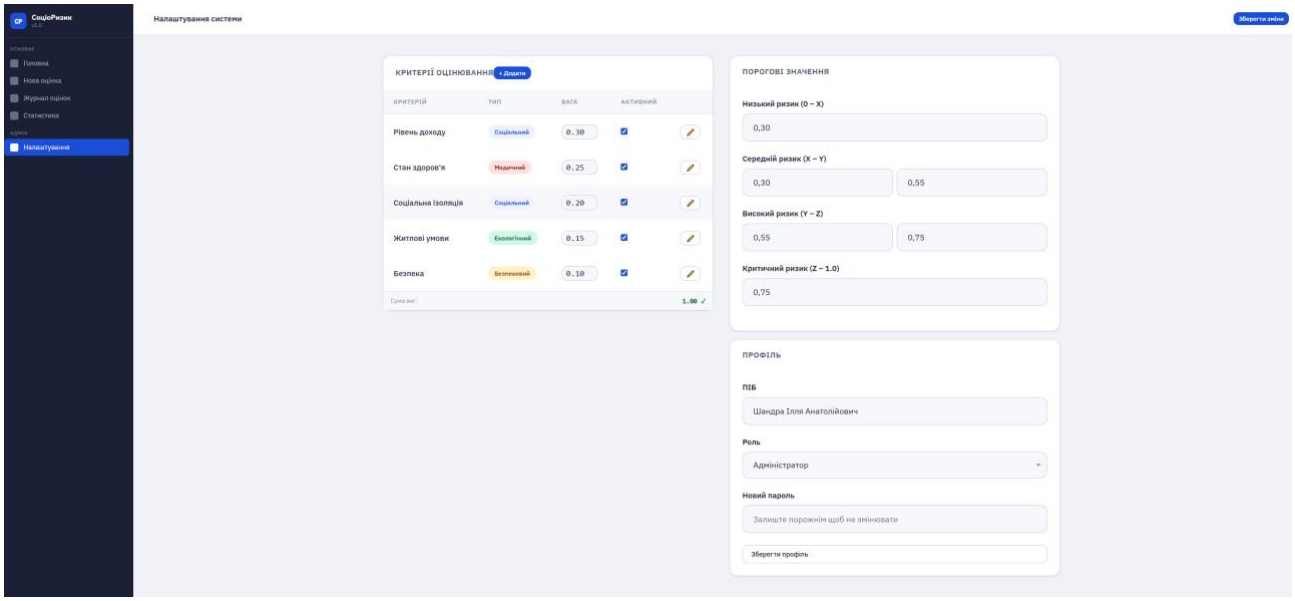



Рис. 4.2 – Графічний інтерфейс сторінки адміністратора під час успішної зміни вагового коефіцієнта критерію безпеки (Тест-кейс ТС-09)

Аналіз результатів функціонального тестування свідчить про повну відповідність розробленого програмного забезпечення «СоціоРизик» критеріям надійності, точності обчислень інтегральних індексів та безпеки збереження соціологічних даних. Всі 11 базових тест-кейсів завершилися зі статусом «Успішно», критичних дефектів або логічних розбіжностей під час випробувань виявлено не було.

## 4.2 Вимоги до апаратного та програмного забезпечення

Надійне та безперебійне функціонування інформаційної системи оцінки багатокритеріального ризику для вразливих груп населення «СоціоРизик» безпосередньо залежить від раціонального підбору технічних засобів та системного програмного оточення. Правильно спроектована інфраструктурна

база покликана гарантувати високу відмовостійкість, мінімальний час відгуку на запити користувачів, безпеку конфіденційних персональних і соціологічних даних, а також підтримку стабільної багатокористувацької роботи персоналу соціальних служб у режимі реального часу [24].

Впроваджений програмний комплекс базується на класичній парадигмі розподіленої клієнт-серверної архітектури. Робочі місця користувачів (соціальних працівників, аналітиків) оснащуються веб-інтерфейсом, який відображається через браузер і виступає в ролі клієнтської частини, що бере на себе функції візуалізації інформації та первинної валідації введення.

Клієнтський рівень забезпечує соціальному працівнику інтерактивну форму для фіксації соціологічних маркерів, трансформує дії оператора у структуровані запити до серверної частини та виконує десеріалізацію отриманих відповідей для виведення на монітор кінцевих індексів та графіків. Серверна вертикаль забезпечує фонову обробку даних: обчислювальний модуль фокусується на ітераційному розрахунку зважених оцінок за критеріями, а Database Server - на транзакційній стійкості при фіксації результатів.

Для практичного впровадження інформаційного комплексу «СоціоРизик» визначено мінімальні та рекомендовані параметри інфраструктурного середовища.

### 4.2.1 Вимоги до апаратного забезпечення (Hardware)

Конфігурація технічних засобів повинна враховувати необхідність швидкої математичної обробки багатокритеріальних матриць та генерації звітів великого обсягу. Граничні параметри обладнання наведено в таблиці 4.2.

Таблиця 4.2 – Гранічні параметри обладнення для системи “Соціоризик”

| Обчислювальний | Параметр | Мінімальні технічні характеристики |
|---|---|---|

| вузол | компонента | |
|---|---|---|
| Робоче місце соціального працівника (Клієнт) | Центральний процесор (CPU) | Intel Core i3 / AMD Ryzen 3 або аналогічний з частотою від 2.0 GHz |
| | Оперативна пам'ять (RAM) | Не менше 4 ГБ |
| | Дискова підсистема (SSD/HDD) | Від 200 МБ вільного простору (для кешу браузера та збереження локальних PDF-звітів) |
| | Дисплей та графіка | Інтегрована графічна плата, монітор з роздільною здатністю від 1366x768 |
| | Мережевий інтерфейс | Мережева карта Ethernet або бездротовий модуль Wi-Fi |
| Серверна інфраструктура (Web + DB) | Центральний процесор (CPU) | Багатоядерний серверний процесор Intel Xeon / AMD EPYC (мінімум 4 фізичні ядра) |
| | Operational Memory (RAM) | Не менше 16 ГБ (для стабільної обробки конкурентних запитів від багатьох філій) |
| | Дискова підсистема (Storage) | Від 100 ГБ вільного місця на швидких SSD-накопичувачах (під БД та логи системи) |
| | Мережевий інтерфейс | Гігабітний адаптер (1 Gbps) для забезпечення мінімальних затримок при з'єднанні |

### 4.2.2 Вимоги до апаратного забезпечення (Hardware)

Для розгортання та належного функціонування середовища виконання системних модулів, на терміналах користувачів та серверах має бути інстальовано базове та прикладне програмне забезпечення, перелік якого зафіксовано в таблиці 4.3.

Таблиця 4.3 – Базове необхідне програмне забезпечення для системи “Соціоризик”

| Логічний рівень | Тип програмного забезпечення | Найменування програмного продукту / Версія |
|---|---|---|
| Клієнтський термінал | Операційна система | Microsoft Windows 10 / 11, macOS або дистрибутиви Linux із графічною оболонкою |
| | Прикладне ПЗ | Сучасний веб-браузер (Google Chrome, Mozilla Firefox, Microsoft Edge, Opera) |
| | Мережеві налаштування | Наявність доступу до мережі Інтернет або корпоративної ЛОМ з активованим стеком TCP/IP |
| Серверний сегмент | Операційна система | Microsoft Windows Server (2019/2022) або Linux (Ubuntu Server 22.04 LTS і вище) |
| | Виконавче середовище | Платформа .NET Runtime 8.0 (для запуску хоста додатка) |
| | Система управління БД | MS SQL Server (версія 2019 і вище, включаючи редакцію Express) |
| | Системні компоненти | Веб-сервер (IIS для Windows або Nginx / Kestrel для Linux-інфраструктури) |
| | Засоби безпеки | Налаштований брандмауер, сертифікат |

| | | |
|---|---|---|
| | | SSL/TLS для шифрування трафіку (HTTPS) |

Строге дотримання зазначених інфраструктурних параметрів під час розгортання та практичної експлуатації системи «СоціоРизик» дозволить повністю нівелювати ризики відмови обладнання, забезпечить високу швидкість реагування на запити фахівців соціономічного профілю та гарантує надійний захист інформації відповідно до вимог чинного законодавства щодо захисту персональних даних.

# ВИСНОВКИ

У ході виконання бакалаврської кваліфікаційної роботи розроблено програмне забезпечення з оцінки багатокритеріального індексу ризику для вразливих груп населення – веб-застосунок «СоціоРизик».

У процесі роботи виконано такі завдання:

- проаналізовано предметну область соціального захисту вразливих груп населення, визначено основні проблеми суб'єктивності оцінювання та нерівномірного розподілу ресурсів; встановлено, що відсутність стандартизованого інструменту є ключовим чинником неефективності існуючих процесів;

- проведено огляд існуючих програмних рішень у сфері соціального моніторингу, виявлено їх переваги та недоліки; обґрунтовано доцільність розробки власного рішення, адаптованого до умов українських соціальних служб;

- розроблено багатокритеріальну модель оцінки ризику методом зваженої суми за п'ятьма критеріями: рівень доходу, стан здоров'я, соціальна ізоляція, житлові умови та безпека; обґрунтовано коефіцієнти вагомості кожного критерію та визначено порогові значення для чотирьох рівнів ризику;

- спроєктовано реляційну базу даних з 8 таблицями, розроблено ER-діаграму та доведено відповідність третій нормальній формі, що забезпечує цілісність даних та відсутність надмірності;

- розроблено комплекс UML-діаграм: варіантів використання, класів, пакетів, компонентів та розміщення, що забезпечило чітку документацію архітектурних рішень;

- реалізовано веб-застосунок на стеку ASP.NET Core (C#) + HTML5/CSS3/JavaScript із REST API та JWT-автентифікацією; серверна частина організована за принципом розподілу відповідальності з окремими шарами контролерів, сервісів та репозиторіїв;

- реалізовано систему ролей: соціальний працівник, аналітик, адміністратор із відповідним розмежуванням прав доступу до функцій системи;

- реалізовано формування звітів у форматах PDF та DOCX за результатами кожної проведеної оцінки;

- проведено функціональне тестування: пройдено 11 тест-кейсів, що охоплюють ключові сценарії роботи системи; виявлені в ході тестування помилки виправлено.

Розроблений застосунок забезпечує стандартизацію процесу оцінювання соціального ризику, зменшує суб'єктивність прийняття рішень і підвищує прозорість роботи соціальних служб. Система дозволяє оперативно виявляти осіб із критичним рівнем ризику та своєчасно направляти їм необхідну допомогу. Завдяки розмежуванню ролей кожен учасник процесу має доступ лише до тих функцій, які відповідають його повноваженням, що підвищує безпеку роботи з персональними даними підопічних.

Практична цінність розробки полягає в можливості її реального впровадження в районних та міських центрах соціального обслуговування без значних витрат на інфраструктуру, оскільки застосунок функціонує як веб-сервіс і не вимагає встановлення клієнтського програмного забезпечення на робочих місцях фахівців.

Напрямами подальшого розвитку системи є: розробка мобільного застосунку для проведення оцінок безпосередньо під час виїзних візитів до підопічних; інтеграція з Єдиним реєстром соціальних послуг для

автоматичного підтягування наявних даних про підопічних; впровадження аналітичного модуля для прогнозування динаміки ризику на основі історії попередніх оцінок; розширення переліку критеріїв оцінювання відповідно до галузевих стандартів соціальної роботи.

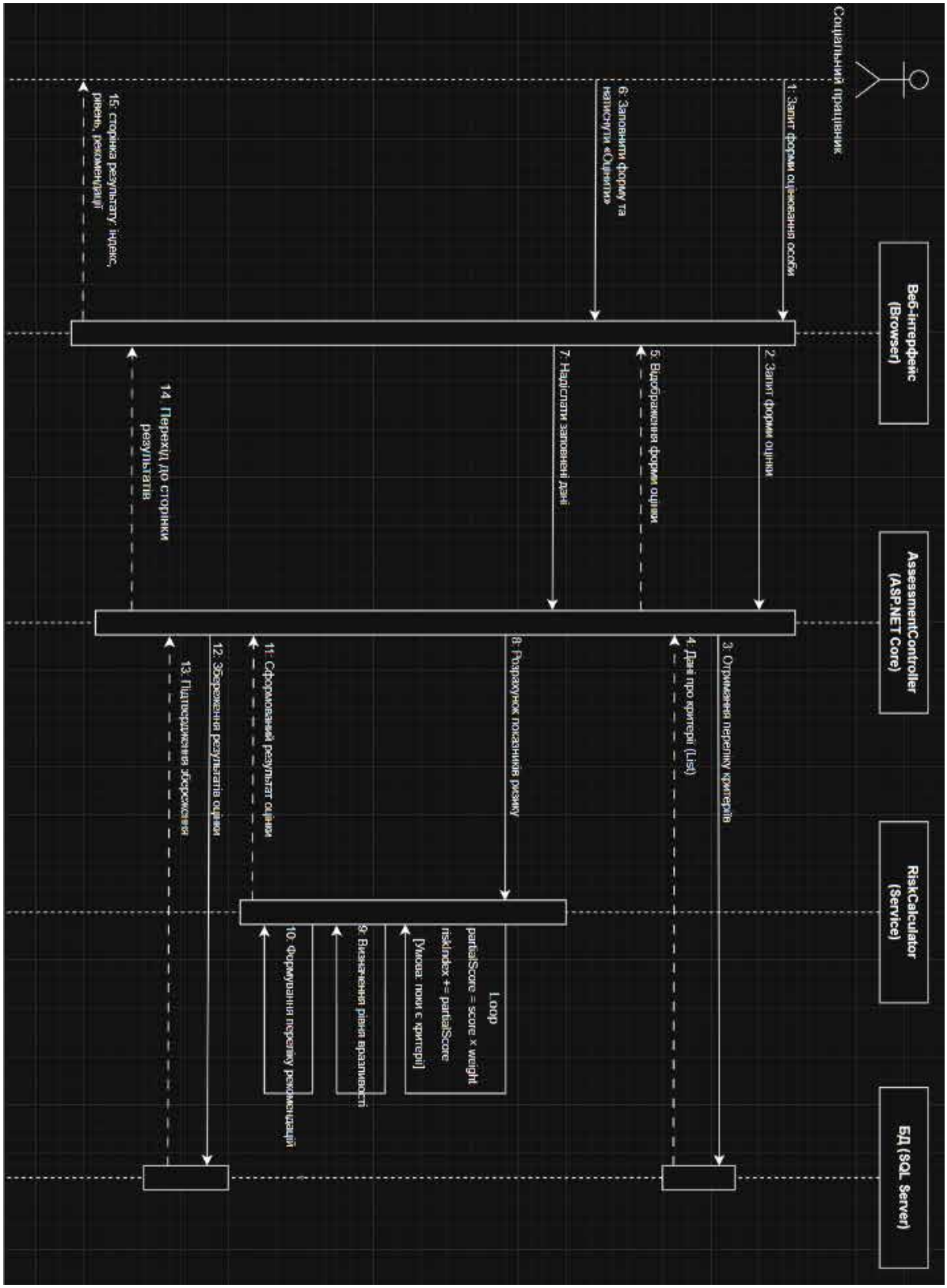
# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Microsoft Learn. C# Programming Guide – [Електронний ресурс] – режим доступу: https://learn.microsoft.com/en-us/dotnet/csharp/ (дата звернення: 05.03.2026).
2. Microsoft Learn. ASP.NET Core Documentation – [Електронний ресурс] – режим доступу: https://learn.microsoft.com/en-us/aspnet/core/ (дата звернення: 06.03.2026).
3. Bootstrap 5 Documentation. – Bootstrap Team, 2024. – [Електронний ресурс] – режим доступу: https://getbootstrap.com/docs/5.3/getting-started/introduction/ (дата звернення: 11.03.2026).
4. Закон України «Про соціальні послуги» від 17.01.2019 № 2671-VIII – [Електронний ресурс] – режим доступу: https://zakon.rada.gov.ua/laws/show/2671-19 (дата звернення: 10.02.2026).
5. Постанова КМУ «Про затвердження критеріїв, за якими оцінюється ступінь ризику від провадження господарської діяльності у сфері надання соціальних послуг» від 03.03.2021 № 183 – [Електронний ресурс] – режим доступу: https://zakon.rada.gov.ua/laws/show/183-2021-%D0%BF (дата звернення: 12.02.2026).
6. Hwang C. L., Yoon K. Multiple Attribute Decision Making: Methods and Applications. – Berlin: Springer-Verlag, 1981. – 259 p.
7. Triantaphyllou E. Multi-Criteria Decision Making Methods: A Comparative Study. – Dordrecht: Kluwer Academic Publishers, 2000. – 320 p.
8. Гніденко М. П., Ільїн О. В. Методи прийняття рішень в інформаційних системах. – К.: ДУТ, 2018. – 142 с. – С. 24-28.
9. Томашевський В. М. Моделювання систем. – К.: Видавнича група BHV, 2005. – 352 с. – С. 9.

10. MDN Web Docs. Web APIs – [Електронний ресурс] – режим доступу: https://developer.mozilla.org/en-US/docs/Web/API (дата звернення: 05.03.2026).
11. MDN Web Docs. HTML: HyperText Markup Language – [Електронний ресурс] – режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 06.03.2026).
12. Entity Framework Core Documentation. – Redmond: Microsoft Press, 2024. – [Електронний ресурс] – режим доступу: https://learn.microsoft.com/en-us/ef/core/ (дата звернення: 11.03.2026).
13. Що таке діаграма варіантів використання UML – [Електронний ресурс] – режим доступу: https://www.mindonmap.com/uk/blog/what-is-a-uml-use-case-diagram/ (дата звернення: 18.03.2026).
14. Як будувати UML-діаграми – [Електронний ресурс] – режим доступу: https://dou.ua/forums/topic/40575/ (дата звернення: 22.03.2026).
15. What is a Logical Data Model? – [Електронний ресурс] – режим доступу: https://www.tibco.com/glossary/what-is-a-logical-data-model (дата звернення: 02.04.2026).
16. Нормалізація баз даних: основні правила та класичні нормальні форми – [Електронний ресурс] – режим доступу: https://uk.code-guild.com/database-normalization/ (дата звернення: 05.04.2026).
17. Саунова Т. О. Визначення та оцінка соціальних ризиків вразливих верств населення. Вісник ХНАУ, 2019. – № 3. – С. 115-122.
18. Застосування патерну MVC в сучасних веб-додатках на платформі .NET – [Електронний ресурс] – режим доступу: https://itvdn.com/ua/blog/articles/mvc-pattern (дата звернення: 20.04.2026).
19. Serilog Documentation. Structured logging for .NET apps – [Електронний ресурс] – режим доступу: https://serilog.net/ (дата звернення: 14.04.2026).

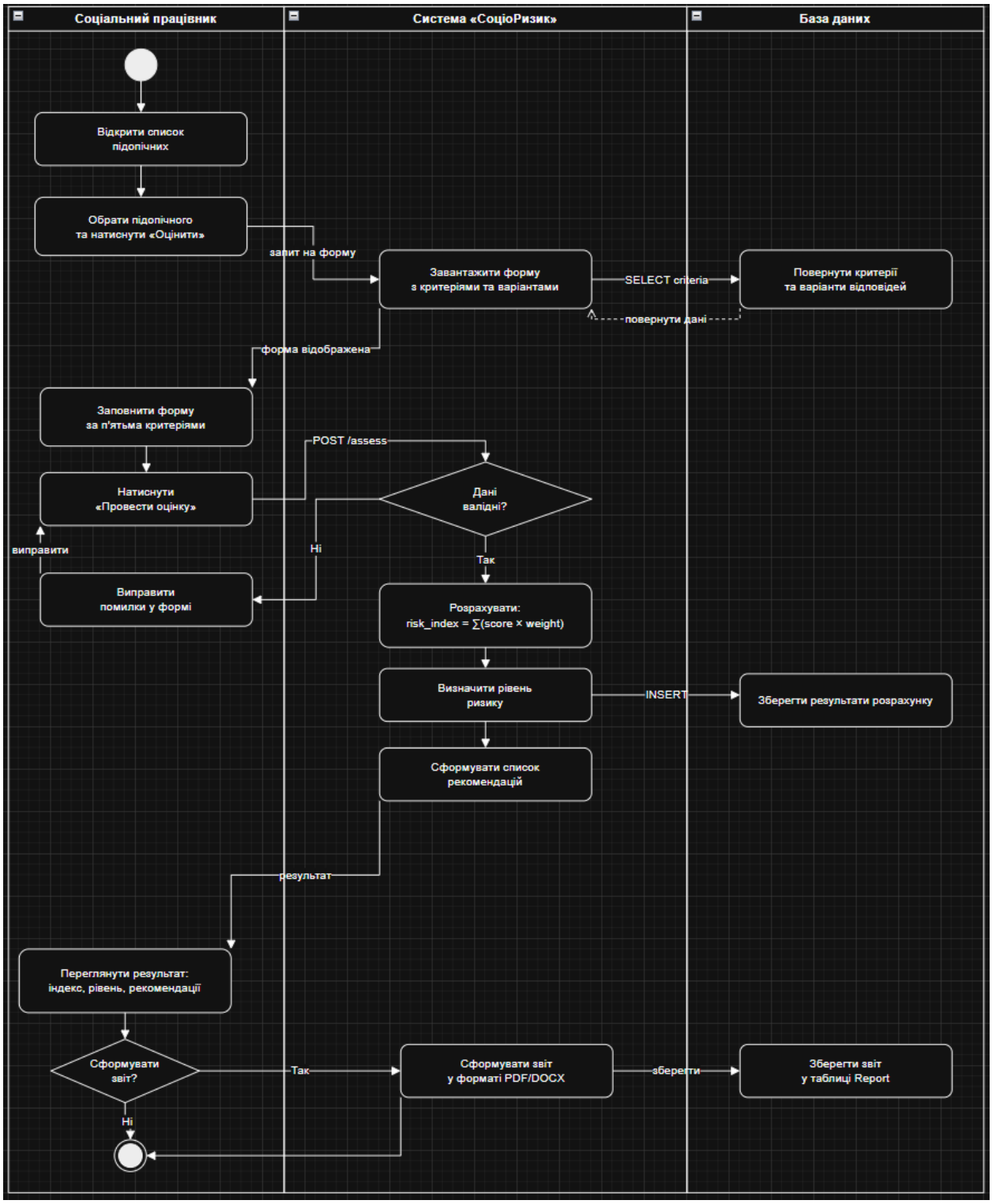
Додаток А

# Діаграма послідовності



Додаток Б

# Діаграма активності



Додаток В

# SQL-скрипти бази даних

```
USE master;
GO

IF DB_ID('SocioRyzyk') IS NOT NULL
    DROP DATABASE SocioRyzyk;
GO

CREATE DATABASE SocioRyzyk
    COLLATE Cyrillic_General_CI_AI;
GO

USE SocioRyzyk;
GO

CREATE TABLE Users (
    UserId        INT            NOT NULL IDENTITY(1,1),
    Login         NVARCHAR(50)   NOT NULL,
    PasswordHash  NVARCHAR(256)  NOT NULL,
    FullName      NVARCHAR(100)  NOT NULL,
    Role          NVARCHAR(20)   NOT NULL
                  CONSTRAINT     CK_Users_Role     CHECK     (Role     IN
('SocialWorker','Analyst','Admin')),
    IsActive      BIT            NOT NULL DEFAULT 1,
    CreatedAt     DATETIME2      NOT NULL DEFAULT GETDATE(),
    CONSTRAINT PK_Users         PRIMARY KEY (UserId),
    CONSTRAINT UQ_Users_Login   UNIQUE      (Login)
);
GO

CREATE TABLE VulnerableGroups (
    GroupId     INT           NOT NULL IDENTITY(1,1),
    GroupName   NVARCHAR(100) NOT NULL,
    Description NVARCHAR(500) NULL,
    CONSTRAINT PK_VulnerableGroups       PRIMARY KEY (GroupId),
    CONSTRAINT UQ_VulnerableGroups_Name  UNIQUE      (GroupName)
);
GO

CREATE TABLE Persons (
    PersonId    INT           NOT NULL IDENTITY(1,1),
    GroupId     INT           NOT NULL,
    FirstName   NVARCHAR(50)  NOT NULL,
    LastName    NVARCHAR(50)  NOT NULL,
    DateOfBirth DATE          NOT NULL,
    Address     NVARCHAR(200) NULL,
    Phone       NVARCHAR(20)  NULL,
    Notes       NVARCHAR(500) NULL,
    CreatedAt   DATETIME2     NOT NULL DEFAULT GETDATE(),
    CONSTRAINT PK_Persons           PRIMARY KEY (PersonId),
    CONSTRAINT FK_Persons_Group     FOREIGN KEY (GroupId)
        REFERENCES VulnerableGroups(GroupId)
        ON DELETE RESTRICT
        ON UPDATE CASCADE
);
GO

CREATE INDEX IX_Persons_GroupId  ON Persons(GroupId);
CREATE INDEX IX_Persons_LastName ON Persons(LastName);
```

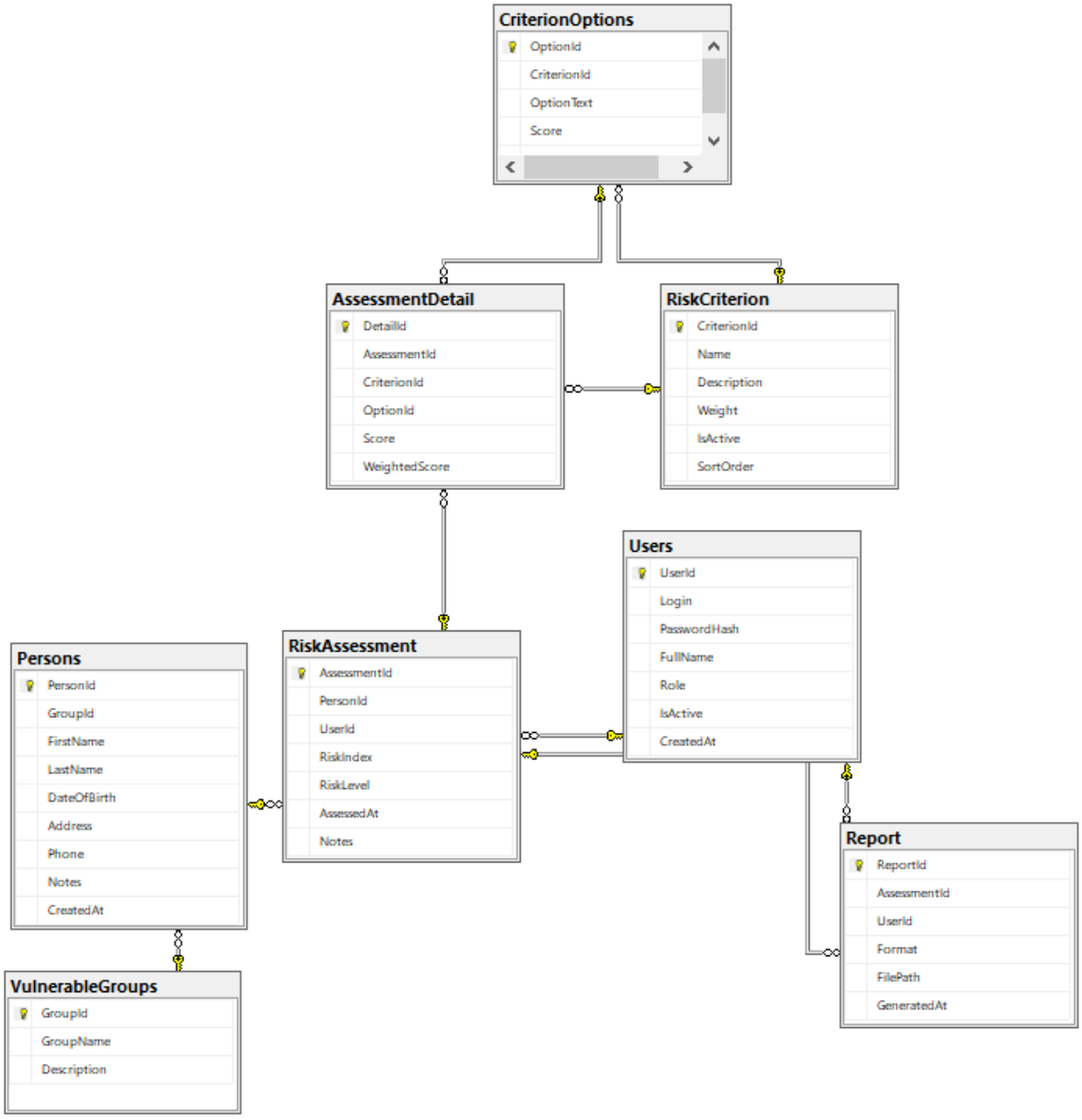
```
GO

CREATE TABLE RiskCriterion (
    CriterionId  INT            NOT NULL IDENTITY(1,1),
    Name         NVARCHAR(100)  NOT NULL,
    Description  NVARCHAR(500)  NULL,
    Weight       DECIMAL(4,2)   NOT NULL
                 CONSTRAINT CK_Criterion_Weight CHECK (Weight > 0 AND Weight <= 1),
    IsActive     BIT            NOT NULL DEFAULT 1,
    SortOrder    INT            NOT NULL DEFAULT 0,
    CONSTRAINT PK_RiskCriterion       PRIMARY KEY (CriterionId),
    CONSTRAINT UQ_RiskCriterion_Name  UNIQUE      (Name)
);
GO

CREATE TABLE CriterionOptions (
    OptionId    INT           NOT NULL IDENTITY(1,1),
    CriterionId INT           NOT NULL,
    OptionText  NVARCHAR(200) NOT NULL,
    Score       DECIMAL(3,2)  NOT NULL
                CONSTRAINT CK_Option_Score CHECK (Score >= 0 AND Score <= 1),
    SortOrder   INT           NOT NULL DEFAULT 0,
    CONSTRAINT PK_CriterionOptions         PRIMARY KEY (OptionId),
    CONSTRAINT FK_CriterionOptions_Crit    FOREIGN KEY (CriterionId)
        REFERENCES RiskCriterion(CriterionId)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
GO

CREATE INDEX IX_CriterionOptions_CriterionId ON CriterionOptions(CriterionId);
GO

CREATE TABLE RiskAssessment (
    AssessmentId  INT           NOT NULL IDENTITY(1,1),
    PersonId      INT           NOT NULL,
    UserId        INT           NOT NULL,
    RiskIndex     DECIMAL(4,2)  NOT NULL
                  CONSTRAINT CK_Assess_Index CHECK (RiskIndex >= 0 AND RiskIndex <= 1),
    RiskLevel     NVARCHAR(20)  NOT NULL
                  CONSTRAINT       CK_Assess_Level       CHECK       (RiskLevel       IN
('Low','Medium','High','Critical')),
    AssessedAt    DATETIME2     NOT NULL DEFAULT GETDATE(),
    Notes         NVARCHAR(1000) NULL,
    CONSTRAINT PK_RiskAssessment          PRIMARY KEY (AssessmentId),
    CONSTRAINT FK_Assessment_Person       FOREIGN KEY (PersonId)
        REFERENCES Persons(PersonId)
        ON DELETE RESTRICT,
    CONSTRAINT FK_Assessment_User         FOREIGN KEY (UserId)
        REFERENCES Users(UserId)
        ON DELETE RESTRICT
);
GO

CREATE INDEX IX_RiskAssessment_PersonId   ON RiskAssessment(PersonId);
CREATE INDEX IX_RiskAssessment_UserId     ON RiskAssessment(UserId);
CREATE INDEX IX_RiskAssessment_AssessedAt ON RiskAssessment(AssessedAt DESC);
GO

CREATE TABLE AssessmentDetail (
    DetailId      INT          NOT NULL IDENTITY(1,1),
    AssessmentId  INT          NOT NULL,
    CriterionId   INT          NOT NULL,
    OptionId      INT          NOT NULL,
    Score         DECIMAL(3,2) NOT NULL,
    WeightedScore DECIMAL(4,3) NOT NULL,
```

```
    CONSTRAINT PK_AssessmentDetail             PRIMARY KEY (DetailId),
    CONSTRAINT FK_Detail_Assessment            FOREIGN KEY (AssessmentId)
        REFERENCES RiskAssessment(AssessmentId)
        ON DELETE CASCADE,
    CONSTRAINT FK_Detail_Criterion             FOREIGN KEY (CriterionId)
        REFERENCES RiskCriterion(CriterionId)
        ON DELETE RESTRICT,
    CONSTRAINT FK_Detail_Option                FOREIGN KEY (OptionId)
        REFERENCES CriterionOptions(OptionId)
        ON DELETE RESTRICT,
    CONSTRAINT UQ_Detail_AssessmentCriterion   UNIQUE (AssessmentId, CriterionId)
);
GO

CREATE INDEX IX_AssessmentDetail_AssessmentId ON AssessmentDetail(AssessmentId);
GO

CREATE TABLE Report (
    ReportId      INT           NOT NULL IDENTITY(1,1),
    AssessmentId  INT           NOT NULL,
    UserId        INT           NOT NULL,
    Format        NVARCHAR(10)  NOT NULL
                  CONSTRAINT CK_Report_Format CHECK (Format IN ('PDF','DOCX')),
    FilePath      NVARCHAR(500) NOT NULL,
    GeneratedAt   DATETIME2     NOT NULL DEFAULT GETDATE(),
    CONSTRAINT PK_Report              PRIMARY KEY (ReportId),
    CONSTRAINT FK_Report_Assessment   FOREIGN KEY (AssessmentId)
        REFERENCES RiskAssessment(AssessmentId)
        ON DELETE CASCADE,
    CONSTRAINT FK_Report_User         FOREIGN KEY (UserId)
        REFERENCES Users(UserId)
        ON DELETE RESTRICT
);
GO

CREATE INDEX IX_Report_AssessmentId ON Report(AssessmentId);
GO

INSERT INTO VulnerableGroups (GroupName, Description) VALUES
('Люди похилого віку',  'Особи у віці 65 років і старші'),
('Діти',                'Особи до 18 років, що перебувають у несприятливих умовах'),
('Особи з інвалідністю','Особи з фізичними, розумовими або сенсорними порушеннями'),
('Малозабезпечені',     'Особи або сім''ї з доходом нижче двох прожиткових мінімумів');
GO

INSERT INTO RiskCriterion (Name, Description, Weight, SortOrder) VALUES
('Рівень доходу',    'Співвідношення доходу особи до прожиткового мінімуму', 0.30, 1),
('Стан здоров''я',   'Наявність хронічних захворювань, обмежень дієздатності',0.25, 2),
('Соціальна ізоляція','Частота соціальних контактів, наявність підтримки',  0.20, 3),
('Житлові умови',    'Якість та безпека житла, наявність комунальних послуг', 0.15, 4),
('Безпека',          'Рівень особистої безпеки, наявність ризиків насильства', 0.10, 5);
GO

INSERT INTO CriterionOptions (CriterionId, OptionText, Score, SortOrder) VALUES
(1, 'Дохід перевищує 3 прожиткових мінімуми',          0.00, 1),
(1, 'Дохід від 2 до 3 прожиткових мінімумів',          0.25, 2),
(1, 'Дохід від 1 до 2 прожиткових мінімумів',          0.50, 3),
(1, 'Дохід менше 1 прожиткового мінімуму',             0.75, 4),
(1, 'Відсутній дохід або борги',                       1.00, 5);
GO

INSERT INTO CriterionOptions (CriterionId, OptionText, Score, SortOrder) VALUES
(2, 'Хронічних захворювань немає',                     0.00, 1),
(2, 'Є одне легке хронічне захворювання',              0.25, 2),
(2, 'Є кілька хронічних захворювань або інвалідність', 0.50, 3),
(2, 'Важка інвалідність, потреба в постійному догляді',0.75, 4),
```

```
(2, 'Термінальний стан або важка деменція',           1.00, 5);
GO

INSERT INTO CriterionOptions (CriterionId, OptionText, Score, SortOrder) VALUES
(3, 'Активне соціальне життя, широке коло підтримки',  0.00, 1),
(3, 'Є родичі або друзі, що підтримують',              0.25, 2),
(3, 'Обмежені контакти, переважно дистанційно',        0.50, 3),
(3, 'Поодинокі контакти раз на тиждень або рідше',     0.75, 4),
(3, 'Повна соціальна ізоляція',                        1.00, 5);
GO

INSERT INTO CriterionOptions (CriterionId, OptionText, Score, SortOrder) VALUES
(4, 'Власне упорядковане житло з усіма зручностями',   0.00, 1),
(4, 'Орендоване житло задовільного стану',             0.25, 2),
(4, 'Незадовільний стан житла або комунальні борги',   0.50, 3),
(4, 'Аварійне житло або відсутність опалення',         0.75, 4),
(4, 'Відсутність постійного житла (бездомність)',      1.00, 5);
GO

INSERT INTO CriterionOptions (CriterionId, OptionText, Score, SortOrder) VALUES
(5, 'Безпечне середовище, відсутність загроз',         0.00, 1),
(5, 'Незначні конфлікти в оточенні',                   0.25, 2),
(5, 'Є ознаки систематичних конфліктів',               0.50, 3),
(5, 'Ризик домашнього насильства або зловживань',      0.75, 4),
(5, 'Підтверджене насильство або критична небезпека',  1.00, 5);
GO

-- Адміністратор за замовчуванням
-- PasswordHash = bcrypt('Admin@2026')
INSERT INTO Users (Login, PasswordHash, FullName, Role) VALUES
('admin', '$2a$12$placeholder_hash_change_me', 'Адміністратор системи', 'Admin');
GO

PRINT 'База даних SocioRyzyk створена успішно!';
GO
```

Додаток Г

# Фізична модель бази даних



Додаток Д

# Реалізація методу CreateAssessment сервісного класу AssessmentService

```
public async Task<AssessmentResultDto> CreateAsync(AssessmentCreateDto dto, int userId)
{
    var optionIds = dto.SelectedOptions.Values.ToList();
    var options = await db.CriterionOptions
        .Include(o => o.Criterion)
        .Where(o => optionIds.Contains(o.OptionId))
        .ToListAsync();

    var items = options
        .Select(o => (o.Criterion, Option: o))
        .ToList();

    var result = calc.Calculate(items);

    await using var tx = await db.Database.BeginTransactionAsync();
    try
    {
        var assessment = new RiskAssessment
        {
            PersonId = dto.PersonId,
            UserId = userId,
            RiskIndex = result.RiskIndex,
            RiskLevel = result.RiskLevel,
            Notes = dto.Notes,
            AssessedAt = DateTime.UtcNow
        };
        db.RiskAssessments.Add(assessment);
        await db.SaveChangesAsync();

        var details = items.Select(x => new AssessmentDetail
        {
            AssessmentId = assessment.AssessmentId,
            CriterionId = x.Criterion.CriterionId,
            OptionId = x.Option.OptionId,
            Score = x.Option.Score,
            WeightedScore = Math.Round(x.Option.Score * x.Criterion.Weight, 3)
        });
        db.AssessmentDetails.AddRange(details);
        await db.SaveChangesAsync();
        await tx.CommitAsync();

        return await GetByIdAsync(assessment.AssessmentId)
            ?? throw new InvalidOperationException("Помилка збереження оцінки");
    }
    catch
    {
        await tx.RollbackAsync();
        throw;
    }
}
```

Додаток Е

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

**Факультет інформаційних технологій**

## Декларація про академічну доброчесність та використання ШІ

## ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Шандра Ілля Анатолійович, здобувач НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

Моя бакалаврська кваліфікаційна робота (проєкт) на тему «Програмне забезпечення з оцінки багатокритеріального індексу ризику для вразливих груп населення» є результатом моєї особистої праці.

1. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
2. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
    - [✓] інструменти генеративного ШІ не використовувалися.
    - [✓] інструменти ШІ (вказати назву: ChatGPT, Gemini, Claude, DeepL, _____інші (вказати назву)) були використані для:
        - [✓] перекладу іншомовних джерел;
        - [ ] стилістичного редагування та перевірки граматики;
        - [✓] допомоги у написанні/відлагодженні програмного коду;
        - [ ] інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«___» _____ 2026 р _____ **Ілля Шандра**

(підпис)