Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавра

зі спеціальності 121 – Інженерія програмного забезпечення

На тему: Розробка мобільного додатку для підвищення безпеки жінок та надання ресурсів для самозахисту

Засвідчую, що в цій кваліфікаційній роботі немає
запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ-21-1
О.А. Геращенко

Керівник
кваліфікаційної роботи _____ А. В. Козиков

Завідувач кафедри _____ А. М. Стрюк

Кривий Ріг
2025

Криворізький національний університет

Факультет: Інформаційних технологій

Кафедра: Моделювання та програмного забезпечення

Ступінь вищої освіти: бакалавр

Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. Кафедри

___ А. М. Стрюк

«__» ___ 2025 р.

ЗАВДАННЯ

на кваліфікаційну роботу

студенту групи ІПЗ-21-1 Геращенко Олександри Анатоліївни

1. Тема: Розробка мобільного додатку для підвищення безпеки жінок та надання ресурсів для самозахисту затверджено наказом по КНУ № 200С від «10» квітня 2025 р.

2. Термін подання студентом закінченої роботи: « 10 » червня 2025р.

3. Вихідні дані по роботі: розробка повинна забезпечити користувачкам доступ до відео- та текстових матеріалів з самозахисту, правової інформації, інструкцій щодо дій у небезпечних ситуаціях, можливість комунікації з іншими користувачками через чат. Додаток повинен гарантувати конфіденційність даних, підтримувати функції реєстрації/авторизації, зворотного зв’язку та персоналізації профілю.

4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): провести аналіз існуючих додатків для підвищення безпеки, розробити архітектуру мобільного додатку, реалізувати функціонал згідно з технічними

вимогами, забезпечити безпеку персональних даних, протестувати додаток та оцінити його ефективність.

5. Перелік ілюстративного матеріалу: структурна схема додатку, блок-схеми алгоритмів функцій, діаграми взаємодії, інтерфейсні екрани додатку (знімки екрана).

Календарний план:

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	Аналіз літературних джерел та існуючих мобільних рішень	10.04 – 17.04
2	Формування цілей, завдань і вимог до мобільного додатку	18.04 – 20.04
3	Проєктування архітектури додатку, розробка Use Case-діаграми	21.04 – 25.04
4	Створення структури інтерфейсу, макетів екрану та навігації	26.04 – 01.05
5	Реалізація функціоналу реєстрації, авторизації та налаштування профілю	02.05 – 07.05
6	Розробка модулів перегляду навчальних матеріалів, правових статей, плану захисту	08.05 – 14.05
7	Створення інтерфейсу адміністратора: управління контентом, модерація, обробка звернень	15.05 – 23.05
8	Проведення тестування функціональності, стабільності та безпеки додатку	24.05 – 26.05
10	Підготовка матеріалів для третього розділу роботи	27.05 – 29.05
11	Оформлення пояснювальної записки	30.05 – 03.06
12	Створення презентації та підготовка доповіді	04.06 – 07.06

Дата видачі завдання: « 10.04 » 2025 р.

Студент О.А. Геращенко

Керівник роботи А.В. Козиков

РЕФЕРАТ

БЕЗПЕКА, САМОЗАХИСТ, МОБІЛЬНИЙ ДОДАТОК, ЧАТ, ВІДЕОМАТЕРІАЛИ, ПРАВОВА ІНФОРМАЦІЯ.

Пояснювальна записка: 96 с., 1 табл., 33 рис., 1 дод., 9 джерел.

Метою кваліфікаційної роботи є розробка мобільного додатку для підвищення безпеки жінок та надання їм доступу до інформаційних та комунікаційних ресурсів щодо самозахисту.

Дослідження стосуються існуючих мобільних рішень для особистої безпеки та в першу чергу стосуються їх переваг та обмежень. Він отримує функціональну рамку щодо розробки програми: потреба в реєстрації та входу, навчальних матеріалів, що мають доступ, планувальник надзвичайних ситуацій, чат для комунікацій користувачів та налаштування особистого профілю.

Що ще важливіше, ця робота стосується аспекту безпеки даних. Доступ до користувачів буде реалізований на аутентифікації на основі токенів. Розробка додатків буде здійснена за допомогою мови програмування Visual Studio та C#. Тестування проводили на функціональності та надійності системи.

ABSTRACT

SECURITY, SELF-DEFENSE, MOBILE APPLICATION, CHAT, VIDEO MATERIALS, LEGAL INFORMATION

Thesis in 96 p., 1 tab., 33 fig., 1 app., 9 references.

In short, this thesis would perhaps develop an application for the mobile phone through which they would be improved regarding women's safety and information regarding communication resources relating to self-defense.

Research in the thesis refers to existing mobile solutions for personal safety and primarily deals with their advantages and limitations. It derives a functional framework concerning the application being developed: the need for registration and login, training materials to be accessed, an emergency situation planner, a chat for user communications, and personal profile settings.

More importantly, this work addresses the aspect of data security. User access would be implemented on token-based authentication. Application development will be done using Visual Studio and C# programming language. Testing was carried out on functionality and reliability of the system.

ЗМІСТ

ВСТУП

Сьогодні особиста безпека жінок стала одним із найактуальніших викликів сучасного суспільства. Адже зростання випадків насильства, домагання, переслідування та інших загроз фізичному й психологічному здоров'ю жінок з кожним днем все більш необхідно вимагає інноваційних способів захисту, ефективних механізмів реагування на кризові ситуацію та допомоги постраждалим. Сучасні цифрові технологій та надзвичайна доступність мобільних пристроїв відкривають безпрецедентні можливості для створення ефективних інструментів, які допомагають жінкам орієнтуватися в небезпечних ситуаціях, оперативно отримувати допомогу та встановлювати зв'язок із іншими в разі надзвичайних подій.

З огляду на постійне зростання нових користувачів мобільних застосунків, зростає необхідність розробки застосунків, які б поєднували в собі функції: захисту, навчання і навчальної комунікації. А, як відомо, суттєвий розрив між попитом на якісні, зручні у використанні застосунки для безпеки жінок і їхньою наявністю на ринку, зокрема в Україні. Цей дисбаланс підкреслює актуальність розробки мобільного додатку, що надаватиме ресурси з самооборони, засоби комунікації, навчальні матеріали та практичні рекомендації з сценаріїв дій для надзвичайних ситуаціях.

Головна мета цього дослідження – спроєктувати та розробити практичний і безпечний мобільний додаток, який підвищуватиме рівень особистої безпеки жінок, надаючи обґрунтовані рекомендації та ресурси для вивчення технік самооборони й стратегій реагування на надзвичайні ситуації.

Для досягнення поставленої мети, необхідно виконати такі завдання:

- аналіз існуючих мобільних додатків для цілей самозахисту та безпеки;
- визначити функціональні вимоги до мобільного додатка;

- розробити архітектуру та інтерфейс додатка;
- реалізувати основні функції: реєстрація/аутентифікація, навчальні матеріали, комунікаційний модуль (чат), сповіщення, підтримка;
- захистити особисті дані користувачок за допомогою токенів доступу;
- протестувати функціональність додатка та перевірити його ефективність.

Результати дослідження можуть бути використані в індустрії інформаційних технологій, особливо в розробці соціально орієнтованих мобільних додатків. Додаток може використовуватися неурядовими організаціями, які працюють над розширенням прав жінок та створенням обізнаності та навичок самозахисту для загальної аудиторії.

Як підсумок, розробка цього мобільного додатка є не лише значним технічним досягненням, а й має великий соціальний вплив на захист і підтримку жінок. Поєднуючи технології з потребами суспільства, проєкт вирішує критичну проблему безпеки, демонструючи, як цифрові іновації можуть служити гуманітарним цілям.

1 СУЧАСНИЙ СТАН І АКТУАЛЬНІСТЬ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1.1 Критичний аналіз літературних джерел за темою

Сучасна дослідницька база містить велику кількість праць, присвячених проблематиці безпеки жінок, агресії та технологічним рішенням, які допомагають жінкам під час надзвичайних ситуацій. Мобільні додатки визнані універсальним спектром інструментів, здатними технічно забезпечувати безпеку жінок у повсякденному житті. Дослідження в цій галузі активно розвиваються.

Як в Україні, так і за кордоном є додатки, розроблені для підвищення безпеки жінок. Аналіз цих рішень є підґрунтям для оцінки поточної техніки безпеки та створення нових підходів. Експерти зазначають, що такі доповнення зазвичай виконують три основні функції: екстрену комунікацію, надання інформації та базову взаємодію з правоохоронними органами або членами спільноти.

У межах аналізу розглядаються функціональні можливості, обмеження та популярність таких застосунків серед користувачів.

WhiteRibbonUa – український мобільний додаток WhiteRibbonUkraine, розроблений Асоціацією «Біла стрічка Україна»,в межах глобальної кампанії проти насильства чоловіків над жінками[1].

Застосунок має простий і зручний інтерфейс, однак йому бракує інтерактивних функцій, як от чату чи персоналізованих планів дій, а також можливостей налаштування під потреби користувача.

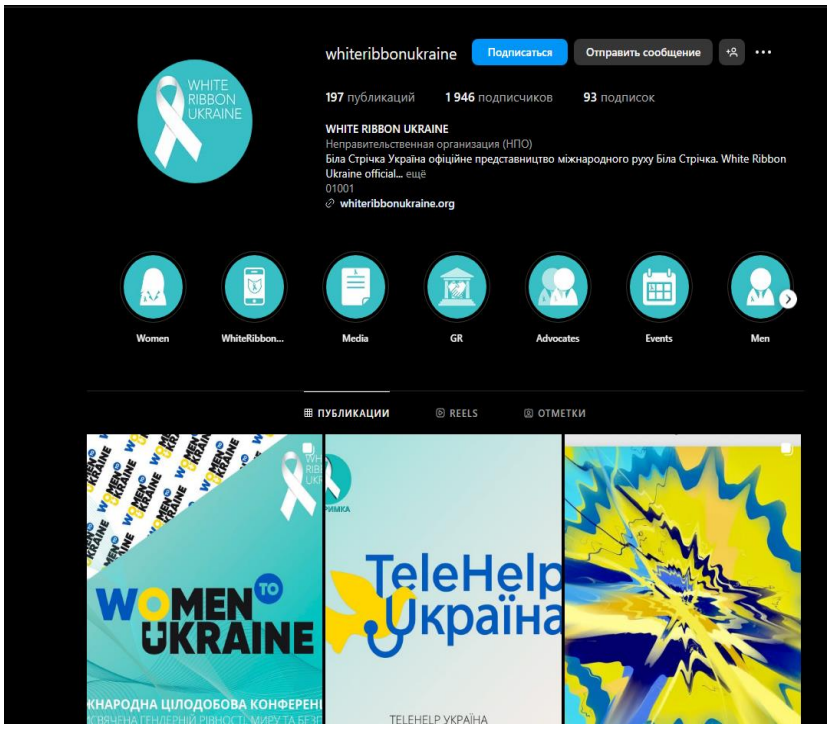

Рисунок 1.1 – Сторінка в Instagram White Ribbon Ukraine

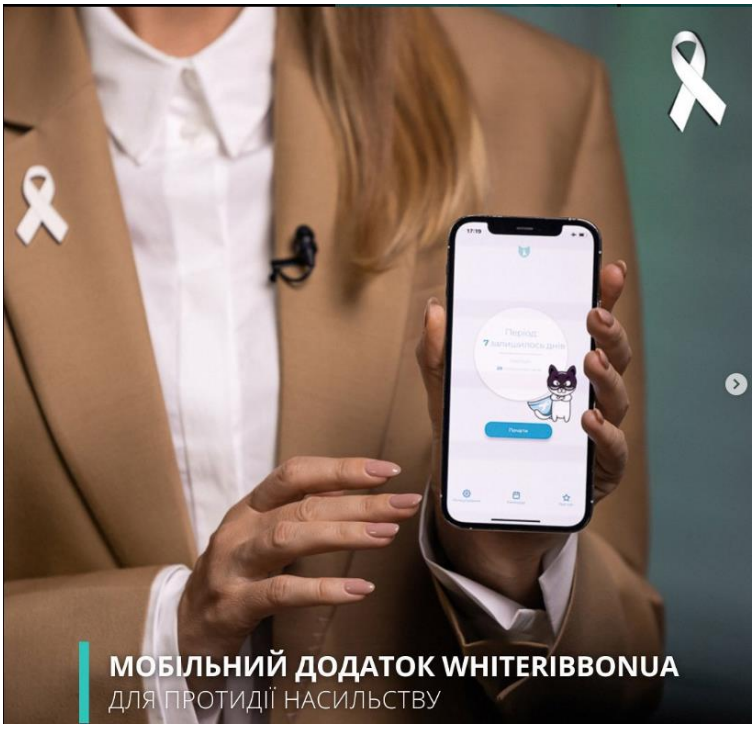

Рисунок 1.2 – Мобільний додаток WHITERIBBONUA

За даними «Української правди», додаток був запущений в листопаді в рамках кампанії «16 днів проти насильства». Розробники підкреслюють, що він корисний не лише жінкам, які шукають захисту, а й усім, хто прагне краще

зрозуміти проблему насильства та способи її розв'язання. Мультинаціональний додаток з можливістю відправки місцезнаходження родичам, SOS, симуляції виклику поліції, освітніх статей, відеоінструкцією та самозахист. Має рейтингову систему небезпечних зон, користувачі оновлюють її. Мінуси – Потребує стабільного інтернет-з'єднання та у рамках безкоштовної версії доступно не тільки скорочений функціонал. Плюси – Популярність, особливо, серед англомовної аудиторії; високі оцінки за багатофункціональність.

Український мобільний застосунок «Крила» створений для підтримки жінок, які опинилися в складних життєвих обставинах. Він надає функції безпеки, психологічну підтримку, мотиваційний контент та доступ до контактів центрів допомоги. Водночас відсутні активні елементи самооборони, конкретні плани дій чи комплексні навчальні матеріали.

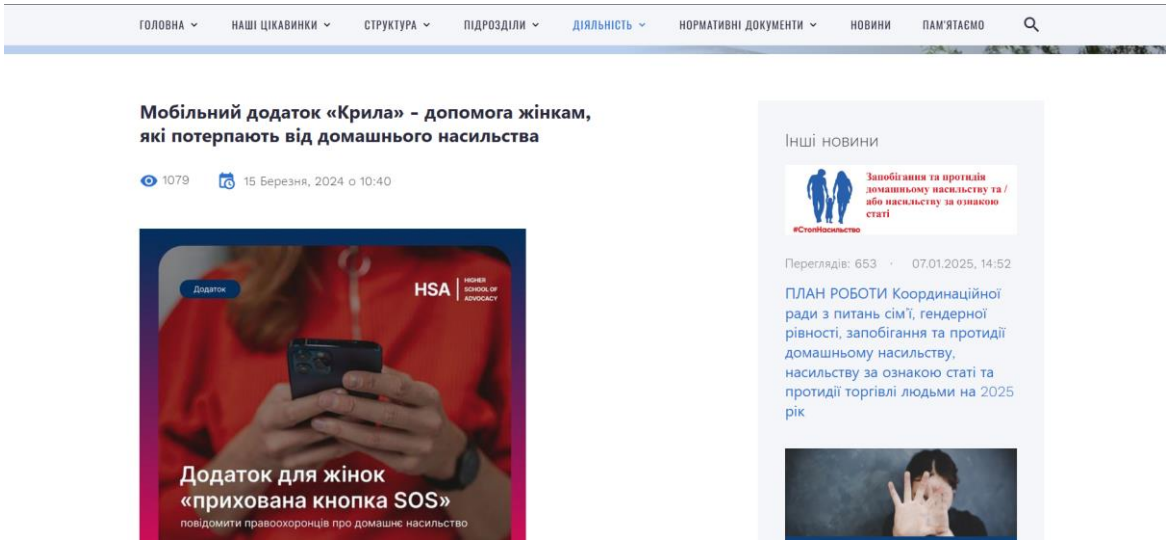

Рисунок 1.3 – Інформація про застосунок «Крила»

За наявною інформацією [2], додаток орієнтований на жінок, які змушені співіснувати з кривдниками, тому застосунок працює під виглядом звичайного калькулятора, що дозволяє зберігати анонімність і мінімізувати ризики викриття.

SafeUp – міжнародний мобільний застосунок, заснований на принципі взаємодопомоги спільноти. Його функціонал включає мережу опікунів, волонтерську допомогу в надзвичайних ситуаціях, відстеження користувача довіреними контактами, аудіо моніторинг для автоматичного виявлення потенційної небезпеки, систему сповіщень і глобальну мапу безпечних і ризикованих зон.

Основні функції:

- мережа Хранительок;
- волонтери, які допоможуть у надзвичайних ситуаціях;
- відстеження користувача через довірених контактів;
- аудіо моніторинг навколишнього середовища для автоматичного виявлення потенційно небезпечних ситуацій;
- система сповіщень для швидкого зв'язку з екстреними контактами;
- глобальна карта безпечних місць та зон підвищеного ризику .

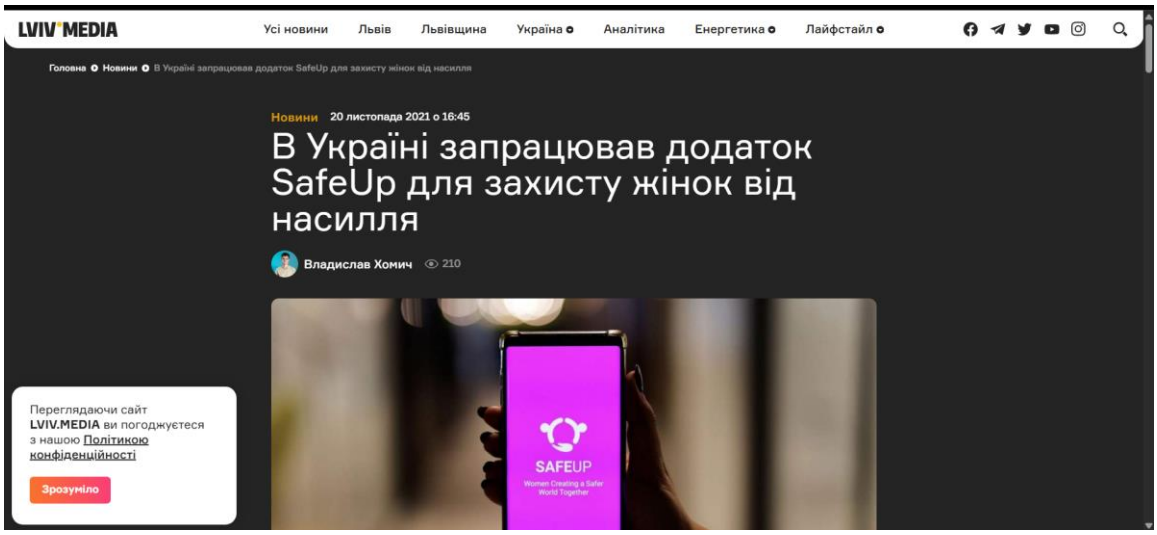

Рисунок 1.4 – Сайт «LVIV MEDIA» про додаток SafeUp

За повідомленням телеканалу «Вікна» [3] SafeUp запропонував новаторський підхід, створивши глобальну жіночу спільноту, яка надає взаємну підтримку в уразливі моменти. Додаток працює онлайн, допомагаючи користувачкам і надаючи їм можливість допомагати іншим.

Аналіз наявних мобільних застосунків для безпеки жінок дозволяє виокремити такі групи функціональних можливостей:

- реакція на надзвичайні ситуації:
    1) тривожна кнопка для швидкого виклику допомоги;
    2) автоматичне інформування довірених осіб про небезпеку;
    3) геолокація для відстеження місцезнаходження користувача ;
    4) аудіо- та відеозапис інцидентів.
- функції соціальної підтримки:
    1) створення мережі для швидкого реагування на потреби користувачів;
    2) контакт спеціалізованих організацій та служб підтримки;
    3) анонімність при спілкуванні з психологами та юристами.

    Безумовно, в таких рішеннях є величезний потенціал, але аналітичні дослідження показують деякі суттєві обмеження:

З наявних даних про популярність користувачів можна сказати, що:

- WhiteRibbonUA: станом на 2023 рік нараховував понад 10 000 завантажень в Україні, проте дані про активних користувачів обмежені;
- "Крила": за даними розробників, додаток має близько 5 000 активних користувачів-жінок, переважно з великих міст України;
- SafeUp: понад 100 000 користувачів по всьому світу, але менше 5% з України.

Центр координації надання безоплатної правової допомоги провів дослідження, яке виявило істотні недоліки у наявних механізмах правового захисту жінок в Україні. У звіті згадуються значні прогалини в системі реагування на випадки насильства, деякі з них:

- низький рівень взаємодії між різними службами (поліцією, соціальними службами, медичними установами), що ускладнює комплексну підтримку постраждалих;
- обмежене обізнання жінок про права та наявні ресурси захисту, що перешкоджає своєчасному зверненню за допомогою;

- відсутність належної інфраструктури підтримки, особливо в сільських районах та малих містах.

Європейський інститут з питань гендерної рівності (EIGE) зробив значний внесок у розуміння використання цифрових технологій для захисту жінок, наголошуючи на важливості рішень, що враховують потреби різних груп — жінок з інвалідністю, літніх, представниць нацменшин та інших вразливих осіб.

Аналіз сучасних мобільних додатків виявляє їхню обмеженість — часто бракує підтримки, освітніх функцій і можливостей для комунікації. Дослідження підкреслюють потребу в інструментах, що не лише реагують на загрозу, а й сприяють її запобіганню — шляхом формування навичок самозахисту та зниження тривожності. У цьому контексті розробка інноваційного додатку є цілком обґрунтованою як теоретично, так і практично.

1.2 Актуальність теми кваліфікаційної роботи

За даними ООН Жінки, 35% жінок у світі хоча б раз зазнавали фізичного або сексуального насильства, не враховуючи психологічного чи економічного [4]. Лише 40% постраждалих звертаються по допомогу через страх, недовіру або брак інформації.

В Україні ситуація ще тривожніша: понад 60% українок ставали жертвами насильства, а щороку понад 1,1 млн жінок зазнають домашнього насильства, з яких офіційно фіксується лише 10–15% [5]. Війна й соціальні кризи лише поглибили ризики.

Основні тенденції:

- зростання випадків насильства в періоди криз [6];
- низький рівень звернень до поліції (10–15%) [7];
- лише третина постраждалих отримує належну допомогу [8].

Це підтверджує потребу в нових цифрових інструментах захисту — доступних, ефективних і придатних до використання в будь-який момент.

Смартфони відіграють ключову роль у повсякденному житті. До 2024 року понад 70% населення України матимуть доступ до мобільного зв’язку, а понад 65% — до мобільного інтернету. Це створює широкі можливості для впровадження захисних мобільних додатків на Android та iOS.

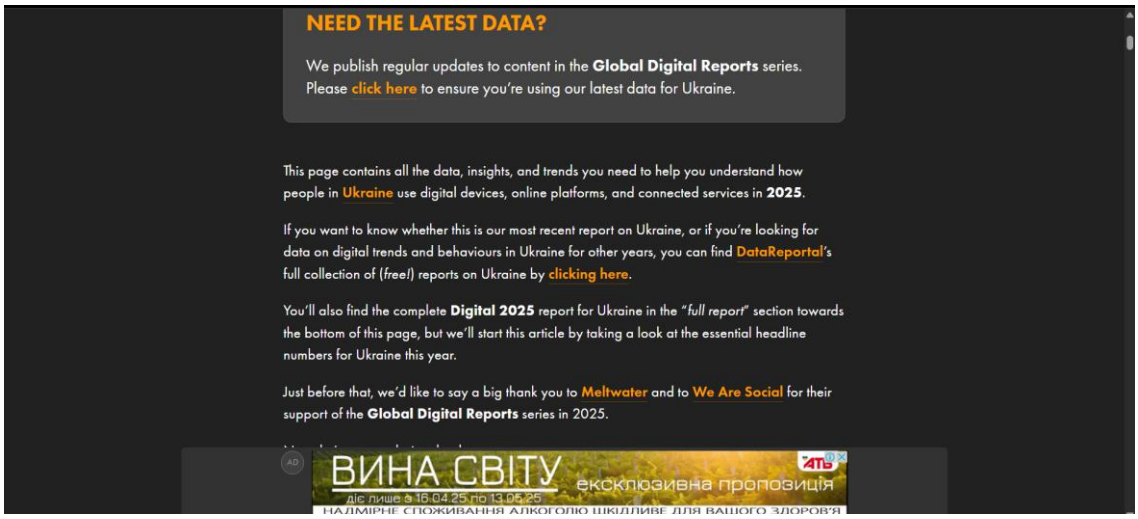

Рисунок 1.5 – Digital 2025: Ukraine

Тенденція до повсюдного користування смартфонами відкриває унікальні можливості для створення додатків допомоги в екстрених ситуаціях, адже пристрій завжди під рукою. Мобільні застосунки мають низку переваг:

- оперативний доступ до інформації та функцій у будь-який момент;
- можливість інтеграції GPS, камер, сповіщень, що підвищує ефективність реагування на загрози;
- персоналізація функцій під конкретні потреби користувача.

Такі застосунки можуть забезпечити базовий просторовий захист жінок.

Водночас ключовим викликом залишається конфіденційність. Уразливість жінок у критичних ситуаціях робить витік даних (місцезнаходження, повідомлення, профілі) потенційно небезпечним.

Закон України про захист персональних даних вимагає::

- безпечне зберігання даних (наприклад, використання шифрування);
- доступ, обмежений уповноваженим персоналом.

Ключові вимоги до безпеки й конфіденційності:

- маскування додатка — вигляд як калькулятор, подвійний інтерфейс, нейтральна іконка, прихований режим;
- захист даних — локальне зберігання з шифруванням, мінімальний збір даних, контроль користувачки над інформацією;
- контроль геолокації — обмежений доступ, можливість відключення, нагадування про стан відстеження.

Баланс між функціональністю й захистом даних є критично важливим. Інтерфейс не має ускладнювати доступ до основних функцій у надзвичайних ситуаціях.

Особливої уваги потребує моделювання сценаріїв, коли агресор має фізичний або цифровий контроль над пристроєм.

Тому, тема кваліфікаційної роботи набуває особливої значущості. Її актуальність зумовлена як нагальною соціальною потребою, так і даними міжнародної статистики, а також наявними технічними обмеженнями сучасних рішень у сфері цифрового захисту жінок.

1.3 Цілі та завдання кваліфікаційної роботи

Метою кваліфікаційної роботи є розробка мобільного додатку, що надає жінкам ресурси для самозахисту та комунікації, спрямовані на підвищення особистої безпеки та обізнаності щодо захисту від насильства.

Для досягнення поставленої мети необхідно виконати такі завдання:

- провести аналіз наявних рішень у сфері мобільних додатків для безпеки;
- визначити функціональні вимоги до додатку;
- розробити структуру та інтерфейс мобільного додатку;
- реалізувати функціонал реєстрації/авторизації з захистом даних;
- додати модулі: навчальні матеріали, плани, систему сповіщень;
- забезпечити зворотний зв’язок (форма підтримки, FAQ);
- провести тестування додатку на функціональність і безпеку;

2 РОЗРОБКА ФУНКЦІОНАЛЬНОЇ СХЕМИ І АЛГОРИТМУ

2.1 Розробка та докладний опис функціональної схеми програми

Мобільний додаток, спрямований на підвищення безпеки жінок та забезпечення їх інструментами самозахисту, має чітко сформовану функціональну схему. Вона орієнтована на простоту використання, інформативність і зручну взаємодію між користувачем і системою. Основна мета функціональної схеми – продемонструвати логічний поділ системи на модулі та зв’язки між ними, що забезпечує ефективність і надійність у виконанні ключових операцій.

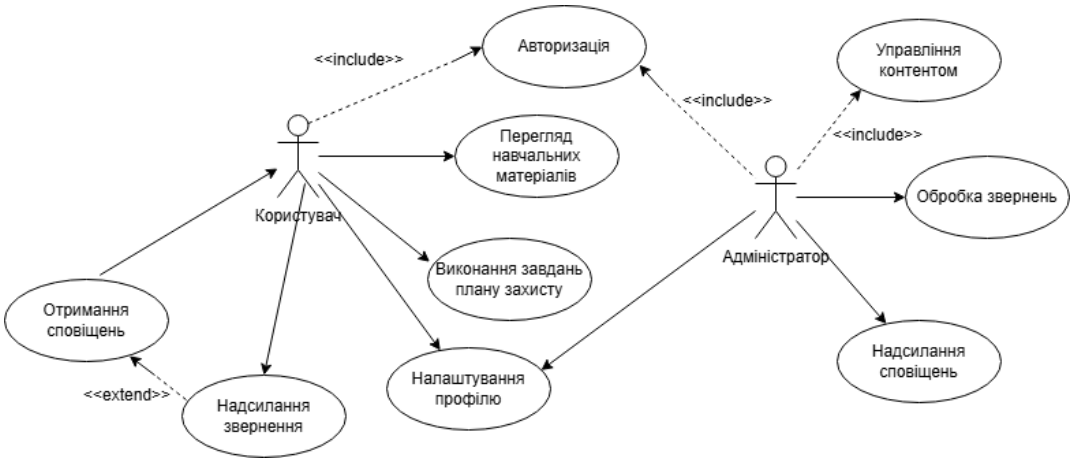

Рисунок 2.1 - Use case diagram

У контексті розробки додатку для підтримки жінок у кризових ситуаціях створено діаграму варіантів використання (див. на рисунок 2.1), яка відображає ключові взаємодії основних акторів з функціональними можливостями системи.

Основними акторами системи є Користувач та Адміністратор. Користувач – представниця цільової аудиторії, що використовує додаток для доступу до інформації, підтримки, самозахисту та правової обізнаності. Адміністратор, у свою чергу, відповідає за технічну і змістову підтримку системи, управління даними, обробку звернень.

Взаємодія користувачки з додатком починається з реєстрації або входу (використання типу <<include>> для всіх інших дій). Після автентифікації відкривається доступ до можливостей: перегляд навчальних матеріалів (відео, інструкції, статті з самозахисту) та ознайомлення з правовими статтями — про законодавство, права жінок і корисні контакти.

Важливою функцією є виконання завдань плану захисту, що допомагає користувачці систематизувати підготовку до небезпечних ситуацій. У разі загроз вона може надіслати звернення через спеціальну форму (варіант <<extend>> — отримання відповіді від адміністратора чи підтримки).

Користувачка також отримує сповіщення про новий контент, нагадування або попередження, може налаштовувати профіль: змінювати особисті дані, мову інтерфейсу та параметри безпеки.

Адміністратор керує контентом: додає, редагує, видаляє матеріали. У цей сценарій включено функцію надсилання сповіщень користувачкам. Також адміністратор обробляє звернення, що надходять від користувачок.

Типові елементи функціональної діаграми:

Модуль авторизації та реєстрації:

- форма реєстрації з перевіркою даних;
- форма входу (логін і пароль);
- автентифікації за токенами;
- відновлення паролю.

Модуль навчальних матеріалів (уроки та статті):

- перелік відеоуроків із самозахисту;
- статті з рекомендаціями та методиками;
- заходи для посилення компетенцій;
- функції фільтрації та пошуку.

Модуль підтримки та зворотного зв'язку:

- форма звернення до адміністрації;
- сторінка поширених запитань;
- автоматичний пошук відповідей за ключовими словами.

Модуль сповіщень:

- сповіщення про нові ресурси, події або важливу інформацію;
- налаштування частоти сповіщень;
- підтримка push-сповіщень.

Модуль налаштування профілю:

- зміна особистої інформації (ім'я, електронна пошта, пароль);
- зміна мови інтерфейсу та теми додатку;
- активувати/деактивувати функції (наприклад, сповіщення або чат).

Внутрішній сервер і база даних:

- обробка запитів користувачів на сервері, перевірка авторизації;
- забезпечення конфіденційності даних у базі даних;
- шифрування даних для їх захисту;
- API для взаємодії клієнтської частини з сервером.

Взаємодія між компонентами: Після входу в систему користувачка отримує доступ до персоналізованого профілю, навчальних матеріалів, сповіщень, плану захисту. Вона може взаємодіяти з іншими компонентами системи, надсилати звернення, налаштовувати профіль. Усі операції здійснюються через захищене з'єднання з сервером.

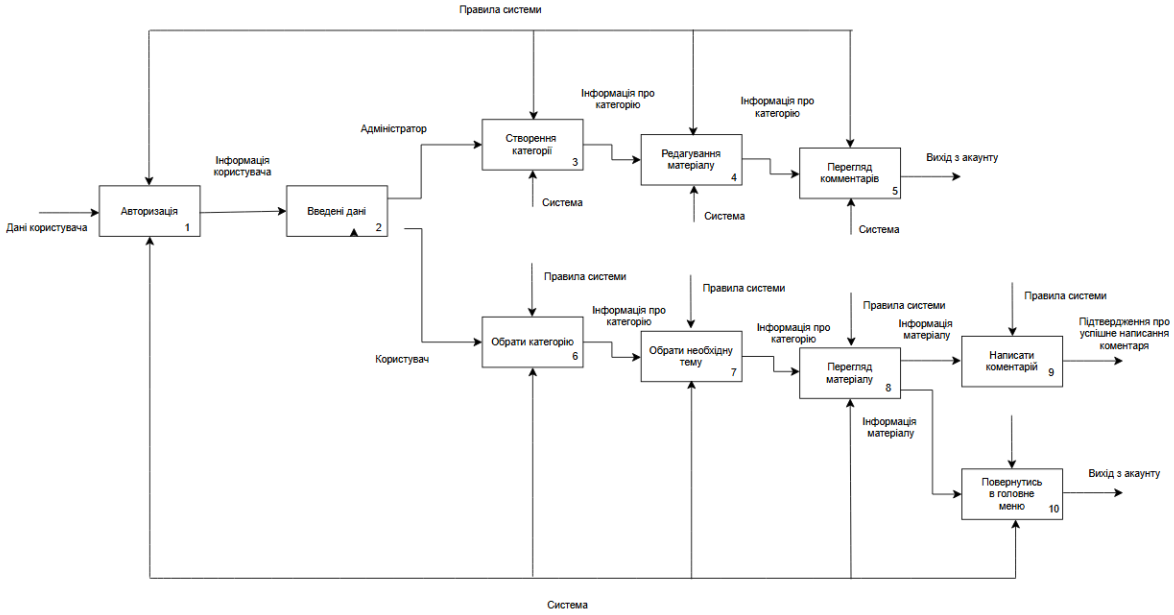

Рисунок 2.2 - Функціональна схема

Функціональна схема(див. рис. 2.2) описує взаємодію користувача та адміністратора із системою мобільного додатку або веб-платформи. Схема охоплює процеси:

- авторизації,
- керування категоріями та матеріалами,
- перегляду і створення коментарів,
- навігації користувача,
- виходу з акаунту.

Кожен процес позначений номером і виконує конкретну дію з чітко визначеними вхідними та вихідними потоками даних.

Опис процесу:

- авторизація
    1) вхід: дані користувача (логін, пароль);
    2) вихід: інформація користувача або відмова;
    3) учасники: користувач / адміністратор;
    4) призначення: забезпечує вхід до системи лише авторизованим особам.
- введені дані
    1) вхід: дані користувача, передані після авторизації;
    2) вихід: передача ролі (користувач / адміністратор) до відповідного модуля;
    3) призначення: розмежовує функціонал за типом користувача.
- створення категорії
    1) доступ: лише адміністратор;
    2) вхід: введення нової категорії;
    3) вихід: інформація про категорію;
    4) призначення: створення структурованого контенту.
- редагування матеріалу
    1) доступ: адміністратор;
    2) вхід: інформація про категорію;

3) вихід: оновлений контент;
4) призначення: актуалізація навчального чи довідкового матеріалу.

- перегляд коментарів
  1) доступ: адміністратор;
  2) вхід: інформація про матеріал;
  3) вихід: переглянуті коментарі;
  4) призначення: модерація або аналіз зворотного зв’язку.
- обрати категорію
  1) доступ: користувач;
  2) вхід: введені дані користувача;
  3) вихід: інформація про категорію;
  4) призначення: дозволяє перейти до певної теми.
- обрати необхідну тему
  1) вхід: категорія;
  2) вихід: інформація про матеріал;
  3) призначення: навігація всередині вибраної категорії.
- перегляд матеріалу
  1) вхід: вибрана тема;
  2) вихід: зміст уроку або статті;
  3) призначення: надання корисної інформації користувачу.
- написати коментар
  1) вхід: вибраний матеріал, правила системи;
  2) вихід: підтвердження про успішне написання;
  3) призначення: надання користувачем відгуку або питання.
- повернутись в головне меню
  1) вхід: бажання користувача повернутис;
  2) вихід: вихід з акаунту або перенаправлення на головну;
  3) призначення: навігаційна функція, завершення сесії.

Таблиця 2.1 - Основні принципи схеми:

Принцип	Реалізація
Розмежування ролей	Користувач і Адміністратор мають різні маршрути взаємодії
Безпека	Авторизація обов'язкова для доступу до системи
Модульність	Система розбита на функціональні блоки
Зворотній зв’язок	Можливість написати коментар і його подальший перегляд
Редагування контенту	Адміністратор оновлює матеріали та додає категорії

2.2 Розробка та докладний опис алгоритму роботи програми

Цей розділ містить покроковий огляд ключових алгоритмів, що використовуються в мобільному додатку. Кожен з алгоритмів є невід'ємною частиною критично важливих функціональних аспектів системи, від реєстрації та автентифікації користувача, доступу до навчальних матеріалів, відстеження прогресу навчання, взаємодії через форму зворотного зв'язку, управління налаштуваннями профілю та системи сповіщень.

Алгоритм авторизації та реєстрації (див. рис. 2.3)користувача засновується на підтвердженні особи шляхом введення облікових даних для входу або створення нового профілю , після чого видається токен доступу. Цей токен гарантує безпечну взаємодію з сервером під час сеансу.Відразу після запуску програми користувачка отримує вибір: увійти до наявного облікового запису або створити новий. При реєстрації потрібно вказати адресу електронної пошти та пароль, що проходить перевірку на відповідність вимогам складності. Якщо дані введено у правильному форматі, вони шифруються та передаються на сервер, де зберігаються в базі даних, після чого

формується токен доступу. Цей токен зберігається у захищеному локальному сховищі пристрою, а користувача перенаправляють до основного інтерфейсу програми.

Вхід до вже існуючого профілю вимагає введення облікових даних, які також перевіряються та шифруються перед відправкою на сервер. Сервер перевіряє наявність відповідних записів у базі даних та, за умови успішного співпадіння, надсилає новий токен доступу. Якщо дані невірні, користувач отримує відповідне сповіщення. При успішній авторизації програма зберігає новий токен та відкриває головний екран. У випадку неправильного логіну чи паролю, система інформує про помилку, а при відсутності з’єднання з сервером передбачено кешування дій або повторна спроба. Коли термін дії токена вичерпується, користувачеві необхідно пройти авторизацію знову.

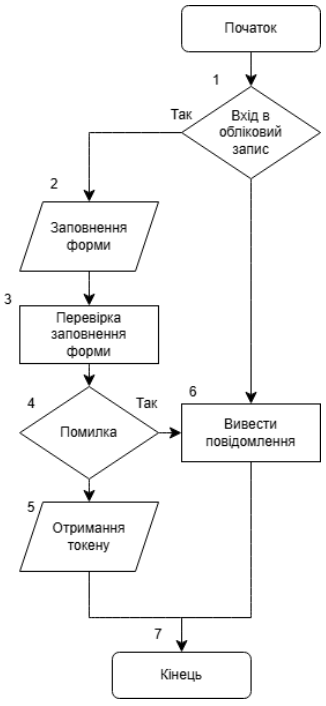

Рисунок 2.3 – Алгоритм авторизації/реєстрації користувача

Алгоритм доступу до навчальних матеріалів і відстеження прогресу(див. рис. 2.4) розкриває механізм, як користувач отримує доступ до освітнього контенту, а також гарантується збереження й синхронізація даних про його дії у мобільному додатку. Після успішної авторизації, що підтверджується

сервером шляхом перевірки токена, завантажується головне меню, де користувач має змогу обрати категорію навчання. Під час перегляду доступних розділів, наприклад, самооборони, домедичної допомоги чи юридичної інформації, користувач надсилає запит до API. API у відповідь повертає перелік тем, що відносяться до обраної категорії. Після вибору певної теми відбувається завантаження навчальних матеріалів, які можуть включати відео, текстові ресурси або комбінацію обох.

У процесі взаємодії з матеріалами ,тобто перегляду відео або читання тексту — програма локально фіксує час активності користувача. Після закінчення сесії або закриття теми відбувається передача даних на сервер. Ці дані містять індетифікатор користувача, назву теми, відсоток перегляду або виконання, а також тривалість взаємодії. Сервер зберігає ці відомості в базі даних. Після цього у профілі користувача з’являється доступна інформація про прогрес у кожній темі: ступінь виконання у відсотках та відповідний статус – від «Не розпочато» до «Завершено».

Система аналітики, аналізуючи зібрані дані, здатна генерувати рекомендації щодо наступних тем або нагадувати про незавершені заняття. У випадках, коли користувач працює офлайн, прогрес зберігається локально, а після відновлення інтернет-з’єднання відбувається автоматична синхронізується з сервером. Додатково, додаток підтримує можливість ручного оновлення прогресу.

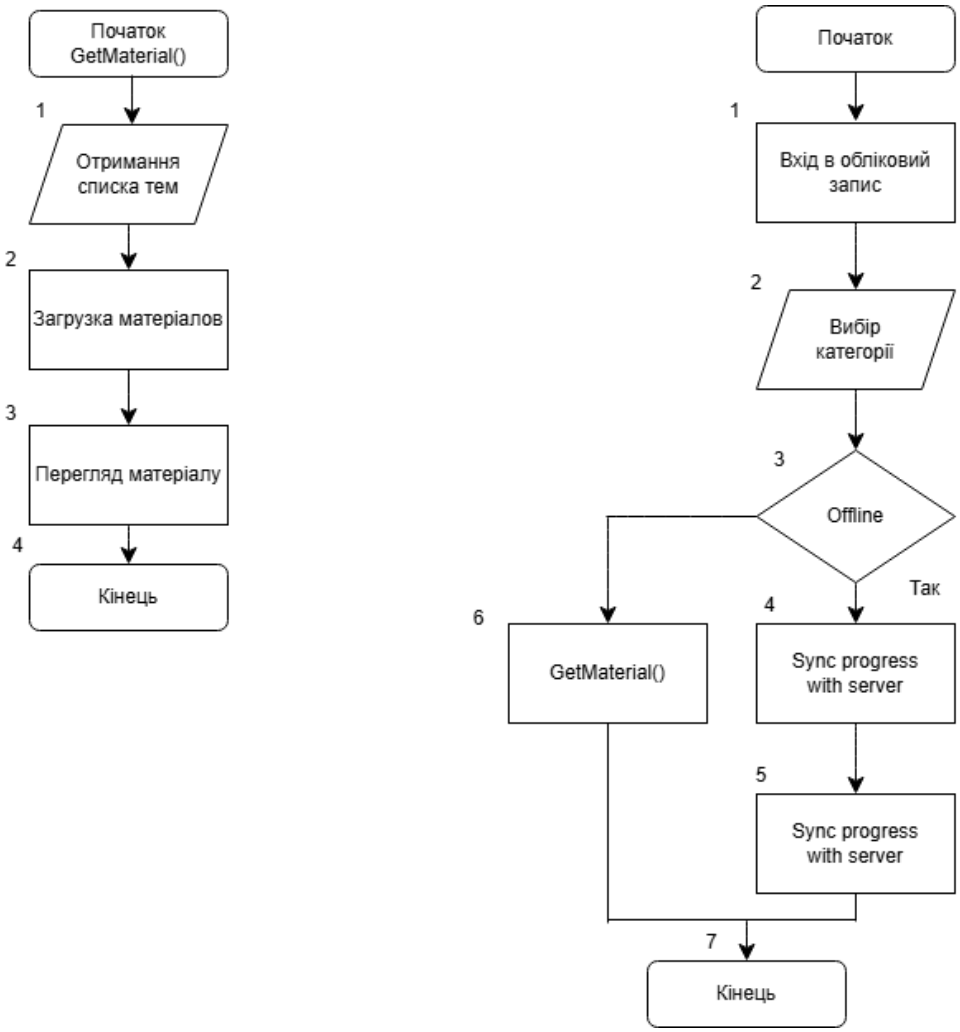

Рисунок 2.4 – Алгоритм доступу до навчальних матеріалів та відстеження прогресу

Алгоритм обробки запитів з форми зворотного зв’язку (див. рис. 2.5) описує, як саме відбувається взаємодія користувача з функціоналом додатку, а також як система фіксує, обробляє та передає звернення до адміністрації або служби підтримки. Користувач розпочинає процес, відкриваючи розділ «Зворотній зв’язок» через головне меню, після чого завантажується форма з полями для введення теми, змісту повідомлення та , за бажанням, типу запиту — наприклад, технічна проблема, побажання чи інше.

Після заповнення відповідних полів відбувається валідація введених даних на стороні клієнта. Далі, натискаючи кнопку «Надіслати», формується POST-запит до сервера.У запит включаються: ідентифікатор користувача (отриманий з токена або профілю), тема та текст звернення, а також позначка

часу надсилання. На сервері перевіряється автентичності користувача, після чого повідомлення зберігається у відповідній таблиці бази даних. Для кожного звернення генерується унікальний ідентифікатор.

Користувач бачить повідомлення про успішне відправлення. Якщо передбачено, він може відстежувати стан свого запиту через розділ «Мої звернення». Адміністратор або співробітник підтримки переглядає отримані повідомлення через спеціальний інтерфейс, відповідає на звернення або змінює його статус (наприклад, «У роботі» чи «Закрито»). У разі оновлення статусу або надходження відповіді користувач отримує відповідне сповіщення.

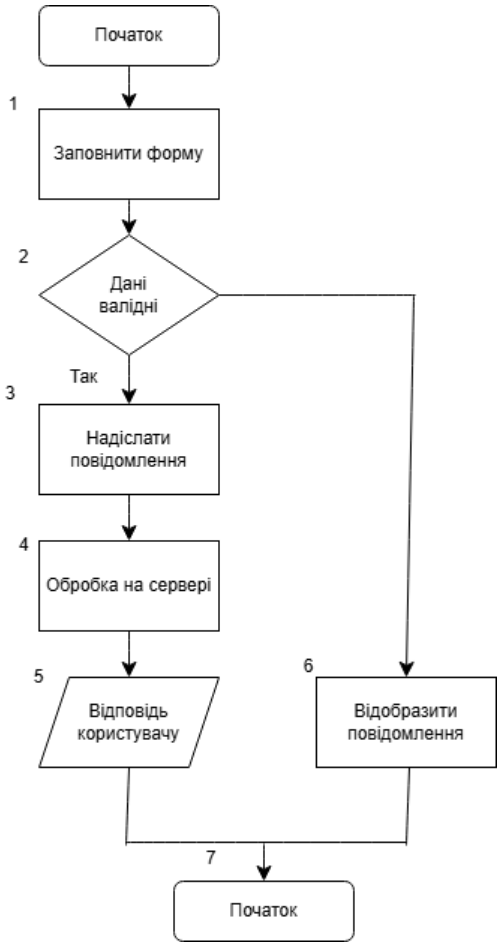

Рисунок 2.5 – Алгоритм обробки запитів з форми зворотного зв’язку

2.3 Розробка інтерфейсу програми

Інтерфейс мобільного додатка розроблено з урахуванням принципів зручності, інтуїтивної навігації та візуальної привабливості. Основна увага приділяється забезпеченню швидкого доступу до ключових функцій програми, особливо до навчального контенту. Головний екран є центральним вузлом взаємодії користувача з додатком, об'єднуючи основні розділи та інструменти.

Основні елементи головного екрана:

- верхній AppBar (заголовок)
    1) відображає назву поточного розділу або вітальне повідомлення користувачу;
    2) може містити іконки для переходу до профілю, сповіщень або швидких налаштувань.
- меню навчальних категорій (відображається як список)
    1) представлено у вигляді вертикального списку або плиток з іконками;
    2) кожна категорія поставляється з коротким описом. натискання на них, відкриває список пов'язаних тем.
- розділ «Рекомендовані теми»
    1) пропонує динамічні пропозиції тем на основі попередньої активності користувача;
    2) дозволяє горизонтально прокручувати відео чи теми.
- індикатор прогресу навчання
    1) показує відсоток завершених тем або загальний прогрес користувача;
    2) може бути реалізований у вигляді кругової діаграми або горизонтального прогрес-бара.
- нижня панель навігації (нижня панель навігації)
    1) основні розділи: головна, навчання, прогрес, зворотній зв’язок та профіль.

- панель/банери сповіщень (необов’язково)
    1) використовується для оголошення оновлень, нових тем, подій чи нагадувань.

Для демонстрації інтерфейсу було створено низку макетів із використанням онлайн-платформи Canva. Вони ілюструють вигляд основних екранів, зокрема:

- авторизація / реєстрація (екрани входу та створення акаунту);
- навчальний екран (категорії, теми, перегляд матеріалів);
- прогрес (звіти та індикатори виконання);
- зворотній зв’язок (форма звернення до адміністрації);
- сповіщення та профіль (налаштування, повідомлення користувачу).

Макети зосереджені на забезпеченні логічної структури, адаптивності для різних розмірів екранів та простоти в користуванні. Їхня візуальна привабливість і функціональність сприяють ефективній роботі з навчальним матеріалом.

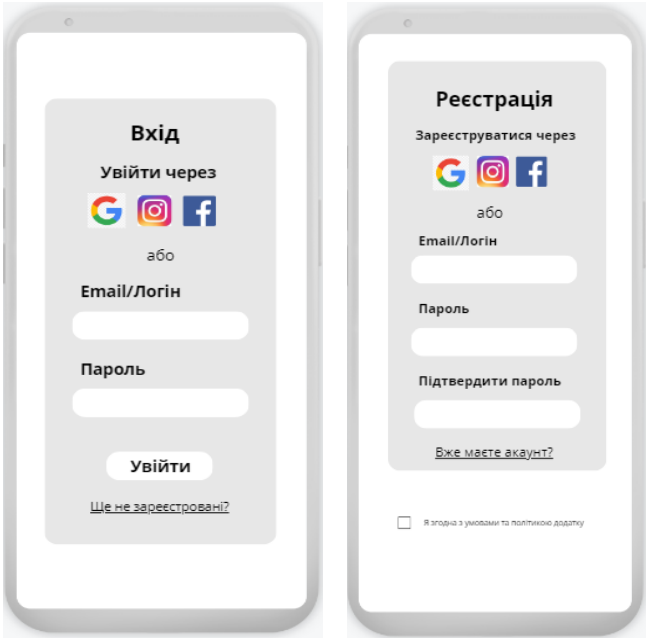

Рисунок 2.6 - Авторизація/Реєстрація

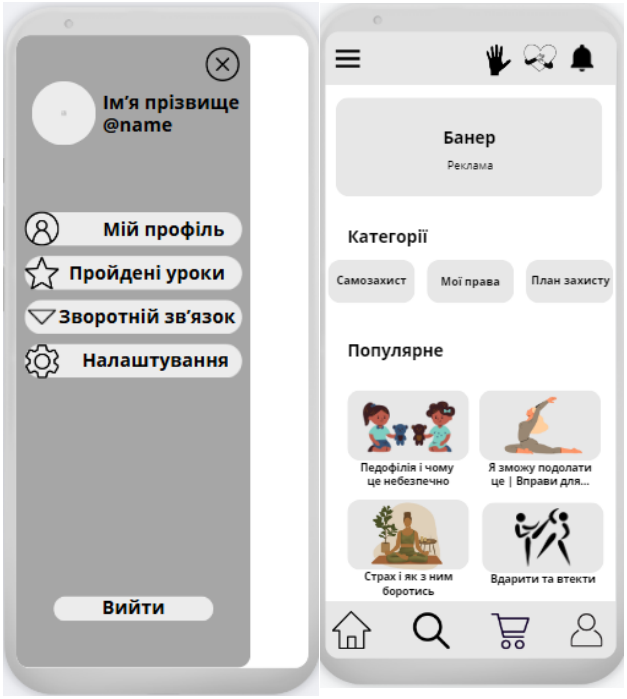

Рисунок 2.7 - Сторінка користувача

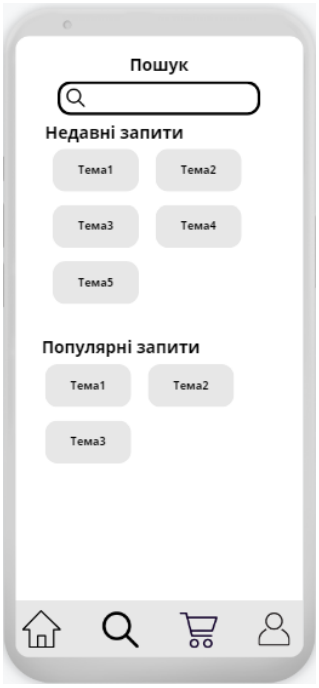

Рисунок 2.8 - Пошук

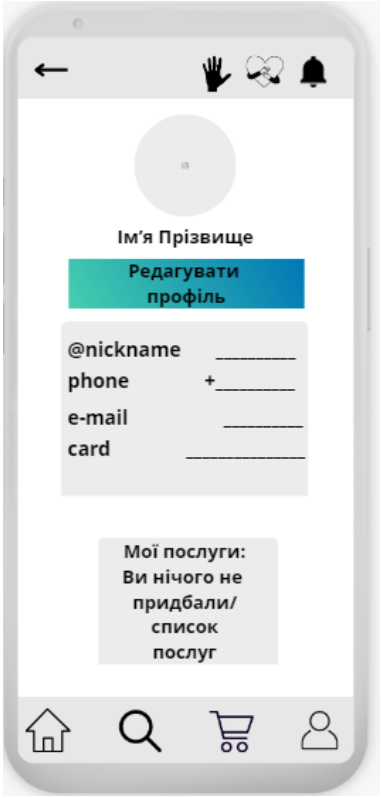

Рисунок 2.9 - Профіль користувача

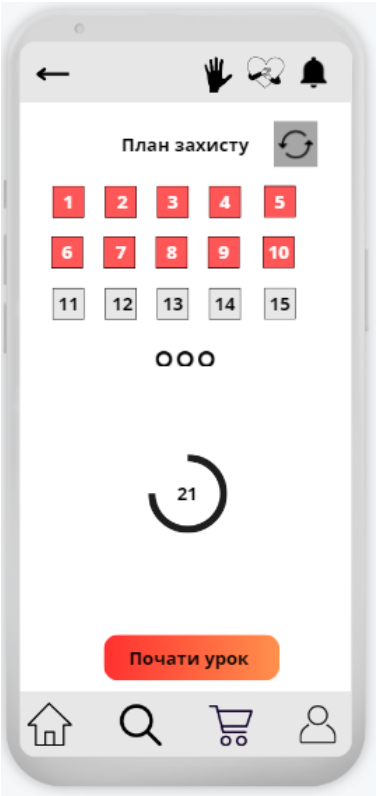

Рисунок 2.10 - Сторінка з планом захисту

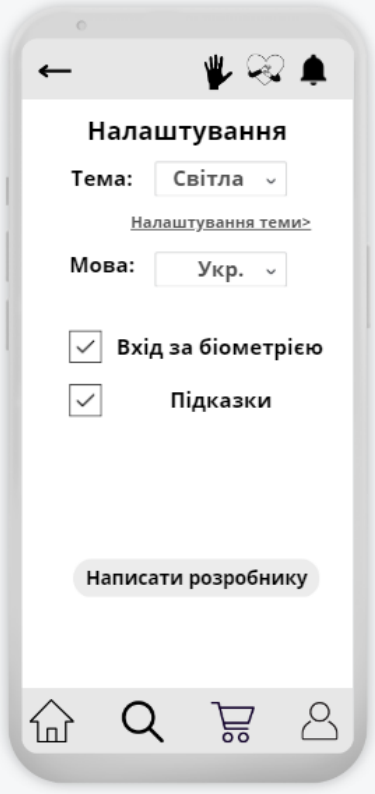

Рисунок 2.11 - Налаштування

3 РОЗРОБКА БАЗИ ДАНИХ І ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз обраної середи програмування

У процесі розробки програмного забезпечення було обрано мову програмування C#, , фреймворк .NET MAUI для створення кросплатформеного інтерфейсу, інтегроване середовище Visual Studio, а також засоби роботи з базами даних, зокрема Entity Framework та SQLite.

Обґрунтування вибору мови програмування (C#)

Мова C# є сучасною, об’єктно-орієнтованою та строго типізованою мовою, що активно підтримується компанією Microsoft. Вона забезпечує високу продуктивність, підтримку багатьох парадигм програмування та потужну інтеграцію з фреймворками .NET.

Переваги C# включають:

- повну інтеграцію з платформою .NET;
- велику кількість бібліотек та фреймворків;
- підтримку асинхронного програмування (async/await);
- значну кількість документації та активну спільноту;
- інтегроване середовище розробки (Visual Studio).

Основним інструментом розробки є Visual Studio Community Edition — безкоштовне, офіційне середовище для створення .NET-додатків. Воно забезпечує потужний набір інструментів для розробки, налагодження, тестування та контролю версій.

Серед переваг Visual Studio виділяються:

- інтегровані засоби для створення графічного інтерфейсу;
- вбудовані засоби для відлагодження (debugger), аналізу продуктивності та профілювання;
- підтримка Git, розширень, шаблонів та хмарних сервісів;
- використання бібліотек/фреймворків.

.NET MAUI (Multi-platform App UI) — сучасний фреймворк, що дозволяє створювати єдину кодову базу для Android, iOS, Windows та macOS. Це робить MAUI ідеальним вибором для мобільних додатків, оскільки:

- дозволяє спільне використання UI та логіки між платформами;
- підтримує XAML для створення інтерфейсів;
- має доступ до API пристроїв (камера, датчики, push-нотифікації);
- інтегрується з Visual Studio та .NET.

Entity Framework – ORM-фреймворк, що дає змогу працювати з базами даних за допомогою об'єктної моделі, не пишучи вручну SQL-запити. Це спрощує доступ до даних, пришвидшує розробку і зменшує ризик помилок.

SQLite – легка, вбудована база даних, що оптимально підходить для десктопних додатків, не вимагає окремого сервера та підтримується Entity Framework. SQLite дозволяє зберігати всю інформацію локально в одному файлі.

Вибрані технології було обрано з огляду на їхню зручність, гнучкість, безкоштовну доступність та високий рівень підтримки. Основні причини такого вибору:

- доступність: усі використані засоби (C#, Visual Studio Community, SQLite) є безкоштовними та легко встановлюються;
- популярність: .NET та C# мають високі рейтинги серед мов та платформ розробки (відповідно до рейтингів Stack Overflow, TIOBE Index);
- підтримка та документація: Microsoft регулярно оновлює та підтримує інструменти, а також надає розгорнуту офіційну документацію;
- гнучкість: платформа .NET і MAUI дозволяють легко масштабувати проєкт, створювати кросплатформні рішення та інтегрувати зовнішні сервіси;
- зручність інтеграції: всі компоненти легко поєднуються в межах єдиної екосистеми, що дозволяє скоротити час розробки;

- гнучкість: вибрані інструменти дозволяють масштабувати систему, реалізувати як локальне, так і віддалене зберігання даних.

3.2 Розробка бази даних

Схема зв’язків між таблицями(див. на рис 3.12) бази даних віддзеркалює структуру взаємодії ключових сутностей інформаційної системи, включаючи користувачів, навчальні матеріали, плани захисту, зворотний зв’язком і статті. Центральне місце у цій моделі посідає таблиця користувачів, що має зв’язки з низкою інших таблиць через відношення «один до багатьох».

Зокрема, один користувач може мати безліч записів про свій навчальний прогрес,які зберігаються в таблиці LessonProgress. У цій таблиці міститься зовнішній ключ на ідентифікатор користувача, а також інформація про пройдений матеріал й рівень успішності. Аналогічна структура реалізована у зв’язку з таблицею ProtectionPlanProgress, де фіксується індивідуальний прогрес кожного користувача у виконанні завдань захисного плану.

Крім того користувачі можуть залишати численні звернення через механізм зворотного зв’язку, що зберігаються у відповідній таблиці Feedback разом із темою, текстом повідомленням та міткою часу. Система передбачає розсилку сповіщень, які також прив’язуються до конкретного користувача й мають статус перегляду.

До того ж, самі уроки (Lesson) пов’язані з записами про їх вивчення, що дозволяє визначати, як саме користувачі взаємодіяли з кожним навчальним матеріалом. В свою чергу, плани захисту (ProtectionPlan) містять набір щоденних завдань, представлених у таблиці ProtectionTask, де кожне завдання має власний порядковий номер дня. Кожне з цих завдань, у свою чергу, може бути пройдене багатьма користувачами, інформація про що зберігається у таблиці ProtectionPlanProgress.

Ще передбачено категоризацію статей: кожна стаття належить до певної тематичної категорії, що дозволяє групувати інформаційні матеріали за

релевантними напрямами (наприклад, «Фізичне насилля»). Такий зв’язок реалізований через відношення між таблицями Category та Article.

Всі вищезазначені зв’язки реалізовані як відношення типу «один до багатьох» (1:N), що забезпечує логічну узгодженість і масштабованість структури даних у контексті подальшого розвитку системи.

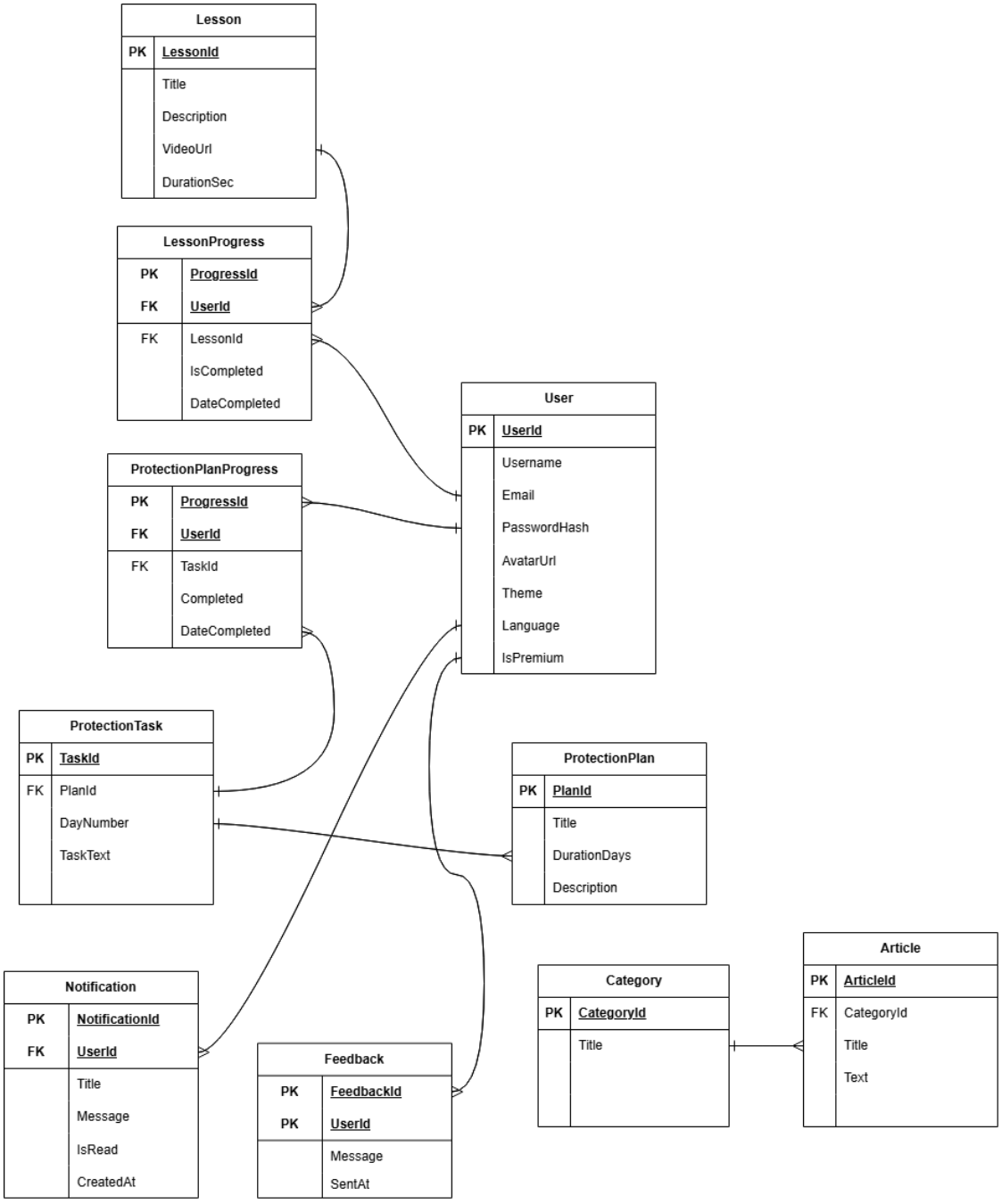

Рисунок 3.12- ER-діаграма

Нормалізація (до 3НФ):

- перша нормальна форма (1НФ). Вимога 1НФ: усі атрибути мають бути атомарними (недільними), а в таблицях не повинно бути повторюваних груп. User: всі поля (ім’я, email, пароль тощо) — атомарні. LessonProgress, ProtectionPlanProgress тощо також мають тільки скалярні значення. Article зберігає текст статті в одному полі, а не кілька колонок для абзаців;
- друга нормальна форма (2НФ). Вимога 2НФ: таблиця має бути в 1НФ, а всі неключові атрибути повинні залежати від усього первинного ключа — а не від його частини. Таблиці з простими ключами (User, Lesson, ProtectionPlan тощо) автоматично у 2НФ. У таблицях зі складеним ключем, наприклад ProtectionPlanProgress (PK = {UserId, TaskId}), усі інші поля (Completed, DateCompleted) залежать від обох частин ключа, а не лише від UserId або лише від TaskId. Аналогічно в LessonProgress (PK = {UserId, LessonId}) — всі поля (IsCompleted, DateCompleted) залежать від обох компонент;
- третя нормальна форма (3НФ). Вимога 2НФ: таблиця має бути в 1НФ, а всі неключові атрибути повинні залежати від усього первинного ключа — а не від його частини. Таблиці з простими ключами (User, Lesson, ProtectionPlan тощо) автоматично у 2НФ. У таблицях зі складеним ключем, наприклад ProtectionPlanProgress (PK = {UserId, TaskId}), усі інші поля (Completed, DateCompleted) залежать від обох частин ключа, а не лише від UserId або лише від TaskId. Аналогічно в LessonProgress (PK = {UserId, LessonId}) — всі поля (IsCompleted, DateCompleted) залежать від обох компонент.

Схематично дивитись на рисунок 3.13.

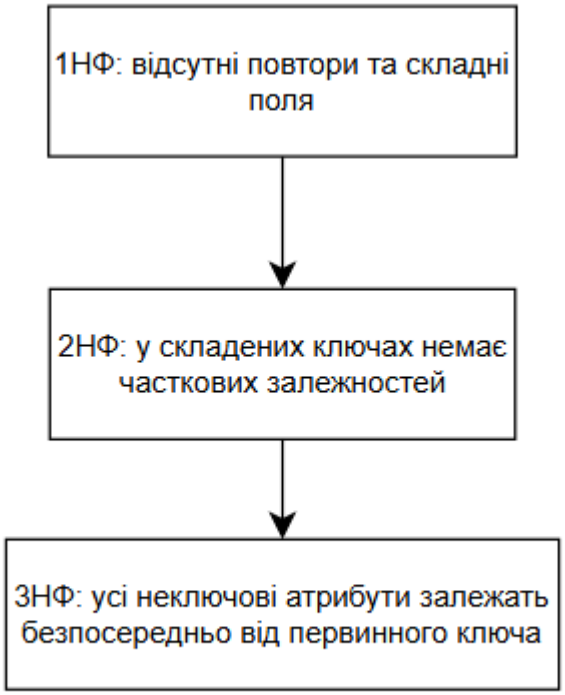

Рисунок 3.13 - Нормалізація

3.3 Програмна реалізація основних функцій

Розроблена програмна система побудована згідно з принципами багаторівневої архітектури орієнтованої на патерни MVC (Model-View-Controller) або MVP (Model-View-Presenter). Такий підхід гарантує поділ обов’язків між представленням інтерфейсу, логікою обробки даних і контролем поведінки, що полегшує супровід коду та його повторне використання.

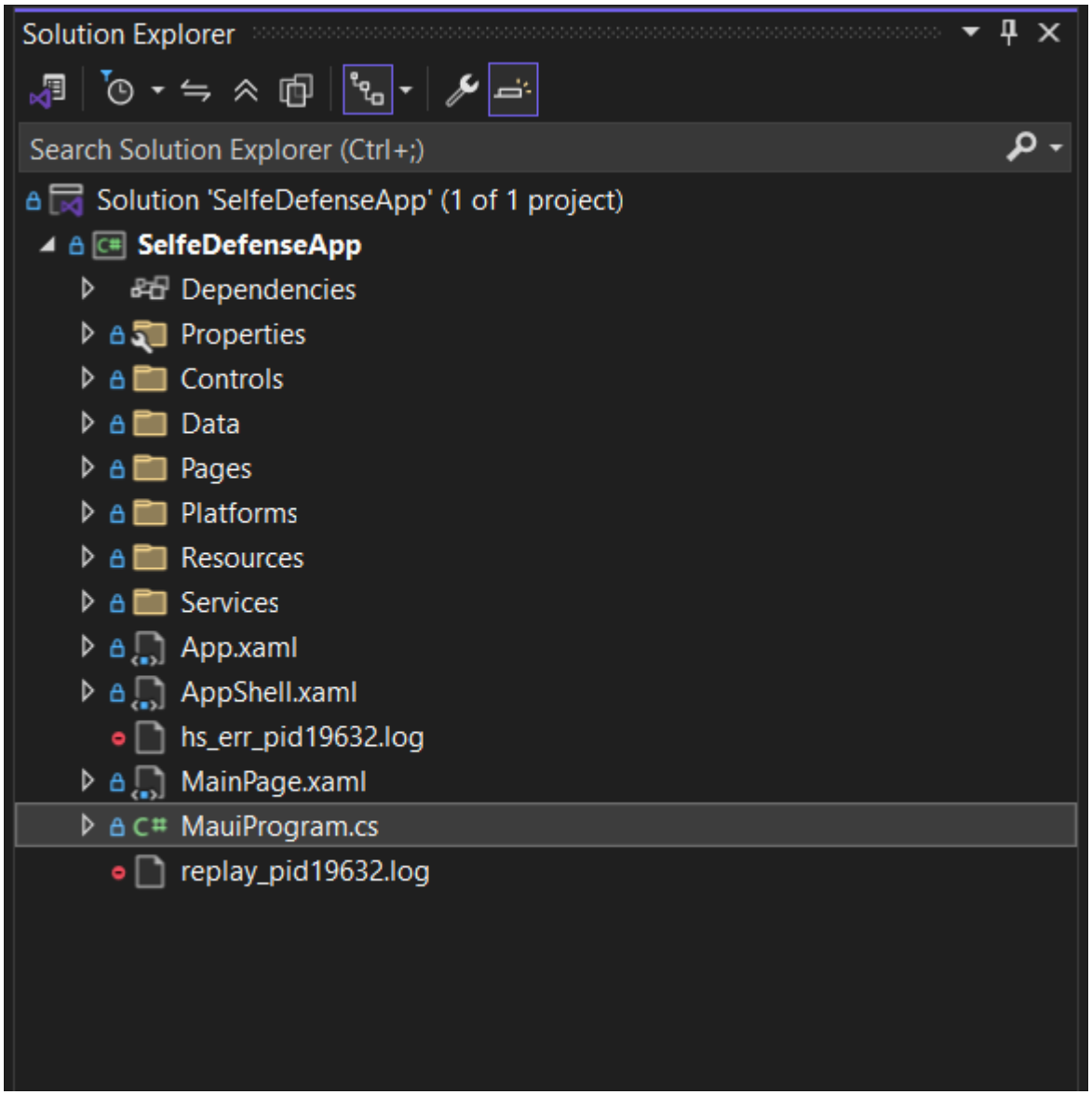

Рисунок 3.14 - Структура папок проєкту

Розглянемо програмну реалізацію (повний код — у Додатку А), починаючи зі сторінки Авторизації/Реєстрації. Сторінка LoginPage (рис. 3.15) забезпечує вхід і створення облікового запису, побудована на ContentPage. Вона містить поля для введення, кнопки «Вхід» і «Реєстрація», а також повідомлення про результати дій. Введені дані обробляє UserRepository через ін’єкцію залежностей. При запуску перевіряється наявність адміністратора — за потреби створюється обліковий запис. Після перевірки даних кнопка входу перенаправляє користувача згідно з його роллю. Сесія зберігається через

SessionService. Під час реєстрації створюється новий користувач із роллю «User», при зайнятому імені виводиться повідомлення про помилку.

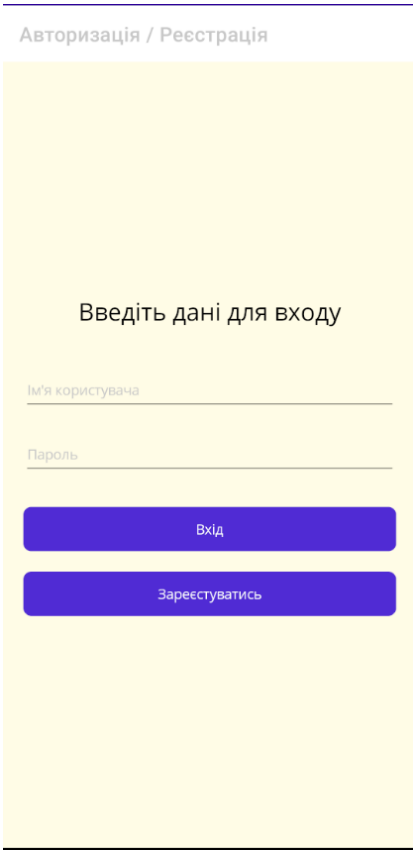

Рисунок 3.15 – Фінальна сторінка реєстрації/авторизаціїї

Головна сторінка користувача — MainMenuPage (рис. 3.16) — основний інтерфейс для роботи з планами самозахисту й тематичними матеріалами. Вона адаптується до теми й мови за допомогою ThemeService і LocalizationService. У верхній частині — навігаційна панель із кнопками доступу до сповіщень, профілю, налаштувань, плану й зворотного зв’язку. Основний вміст — кнопка переходу до плану самозахисту та інтерактивні категорії з бази даних. Уся структура побудована на сітці для зручності користування.

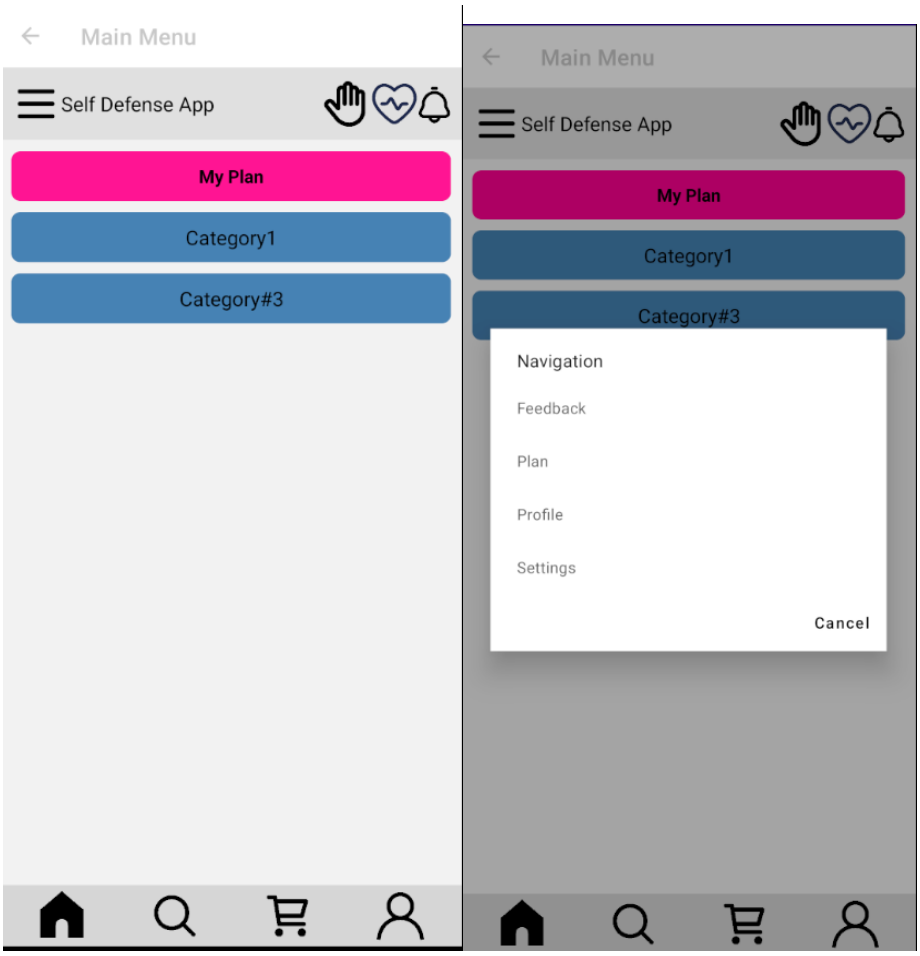

Рисунок 3.16 – Фінальна сторінка головного меню користувача

Головна сторінка адміністратора (AdminMainMenuPage) орієнтована на керування вмістом платформи. Вона реагує на зміну мови та теми, оновлюючи зовнішній вигляд динамічно. Після завантаження перевіряється наявність непрочитаних сповіщень з відповідною індикацією. Основну частину інтерфейсу займають адміністративні функції: редагування планів, створення та керування категоріями. Для кожної категорії надаються кнопки перегляду й редагування, які відкривають відповідні сторінки. Інформація подається у впорядкованому вигляді, що спрощує адміністрування. Додаткове меню забезпечує доступ до профілю, налаштувань, статистики та зворотного зв’язку.

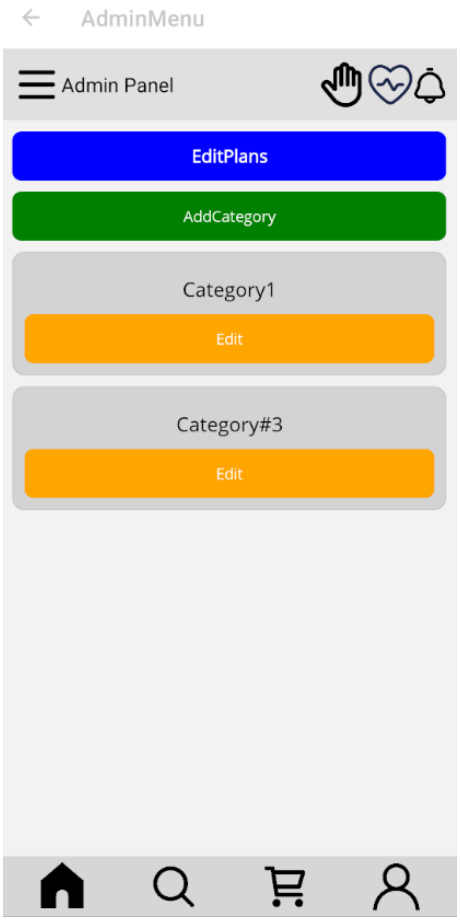

Рисунок 3.17 – Фінальна сторінка головного меню адміна

Сторінка ProtectionPlanPage (див. рис. 3.18) надає користувачу можливість переглядати всі доступні плани самозахисту. Після завантаження даних із бази, кожен план виводиться у вигляді інтерактивної кнопки. Натискання ініціює перехід до сторінки з уроками відповідного плану, передаючи його ідентифікатор та назву.

Сторінка AdminPlansPage (див. рис. 3.19) виконує функції адміністрування навчальних планів. Вона містить кнопку для додавання нового плану та список існуючих, де для кожного передбачено дві дії: перегляд уроків і редагування. Інтерфейс реалізований у вигляді вертикального прокручуваного контейнера, що забезпечує зручне представлення даних.

Сторінка уроку (див. рис. 3.21) містить функціонал позначення уроку як завершеного через відповідний прапорець. Його стан зчитується та

оновлюється у базі даних при зміні. Реалізовано навігацію між уроками, якщо вони доступні в межах поточного плану. Крім того, користувач має змогу залишати коментарі — передбачено текстове поле для введення, кнопку надсилання та область для перегляду наявних відгуків.

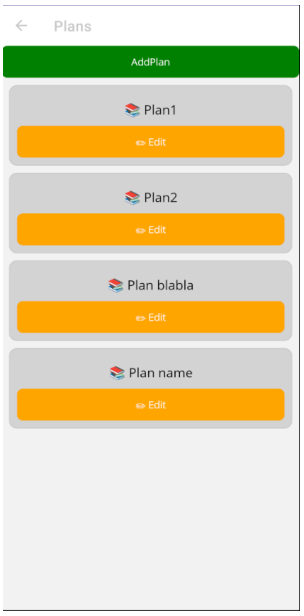

Рисунок 3.18 – Фінальна сторінка планів у адміністратора

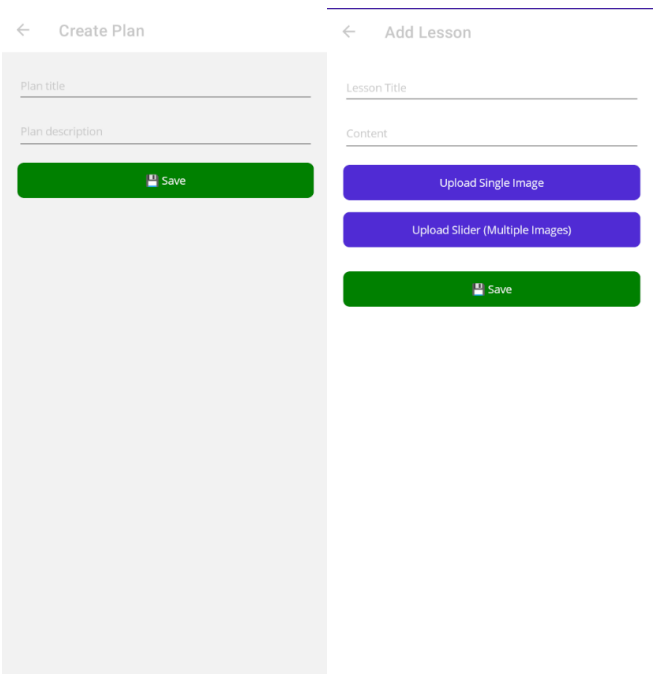

Рисунок 3.19 – Додавання плану та уроків

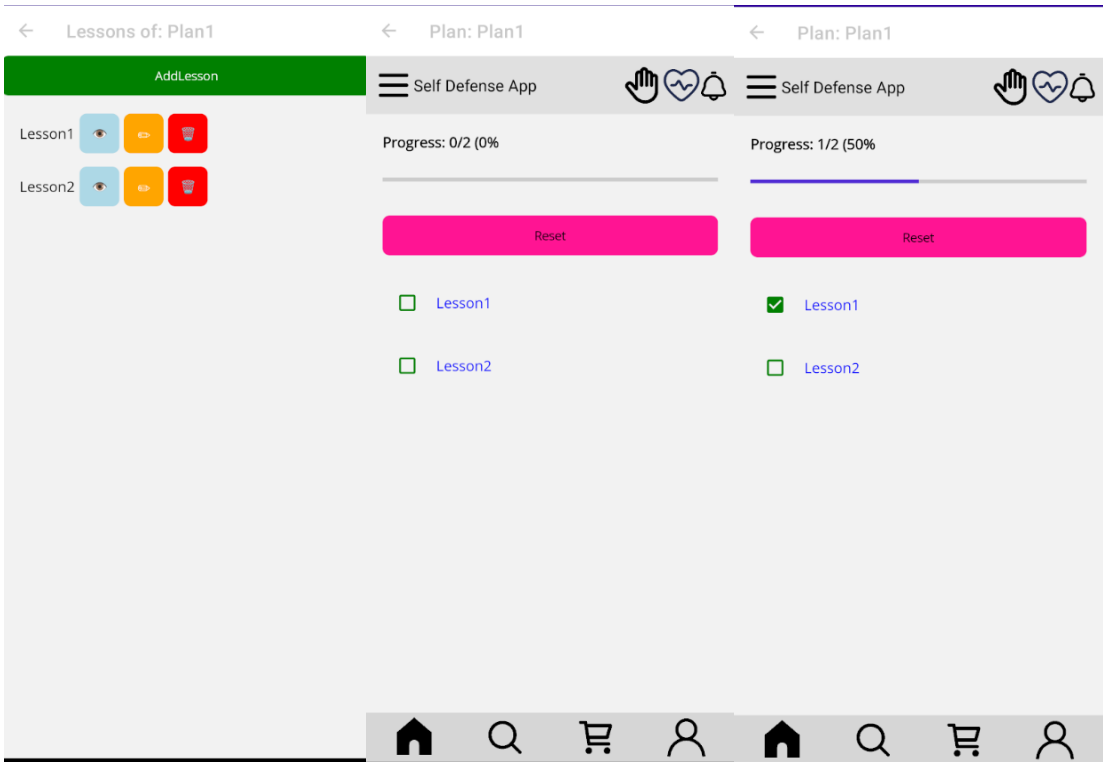

Рисунок 3.20 – Перегляд уроків у користувача та адміністратора

Клас AdminLessonEditPage забезпечує інтерфейс для створення або редагування уроків у межах навчального плану (див. рис. 3.20, зліва). Сторінка дозволяє вводити заголовок, текст, додавати зображення з попереднім переглядом або у вигляді каруселі, а також видаляти окремі з них до збереження. Нові уроки створюються, а наявні — оновлюються, зберігаючи дані у локальній базі.

Клас UserLessonListPage відповідає за відображення списку уроків, прив’язаних до конкретного навчального плану (див. рис. 3.20, центр і справа). Відображається загальний прогрес у числовому та графічному форматі. Уроки подано у вигляді списку з чекбоксами для позначення виконання. Натискання на назву відкриває повний зміст уроку, а кнопка "Скинути" обнуляє прогрес користувача. Усі зміни автоматично синхронізуються з базою даних.

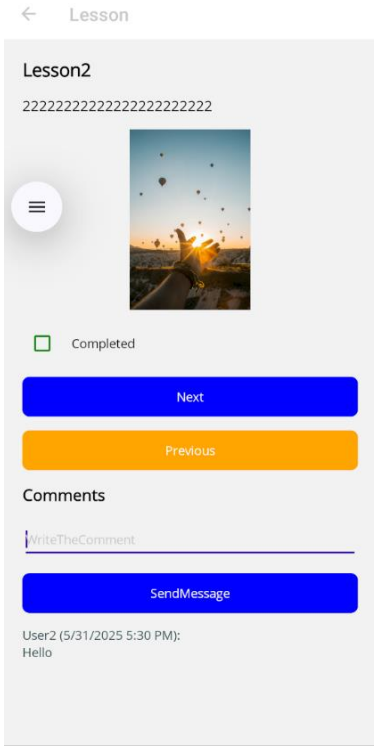

Рисунок 3.21 – Перегляд уроку та коментарі під контентом

На сторінці уроку (див. рис. 3.22) реалізовано інтерактивний функціонал: користувач може відзначити урок як завершений за допомогою прапорця, стан якого зчитується та оновлюється через базу даних. Після збереження змін здійснюється повернення до попередньої сторінки.

У нижній частині сторінки — зона коментування: поле введення, кнопка надсилання та список коментарів, який оновлюється автоматично після додавання нового запису.

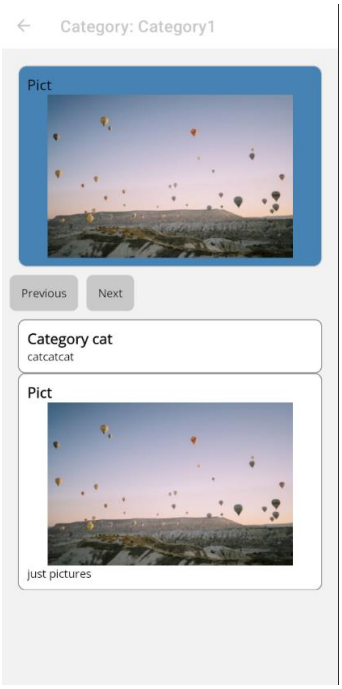

Рисунок 3.22 – Перегляд категорій

Логіка перегляду категорій(див. рис. 3.23) й створення/редагування цієї сторінки від лиця адміністратора аналогічна сторінці плану.

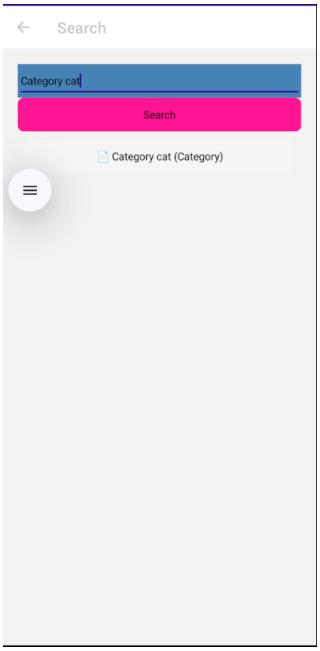

Рисунок 3.23 – Сторінка пошуку

Клас SearchPage реалізує інтерфейс пошуку: містить поле для введення запиту, кнопку запуску та область для результатів. Після натискання кнопки

вхідні дані перевіряються, вивід очищується, а запит обробляється локально через базу даних. Результати — уроки, тексти та зображення — фільтруються за збігами у заголовку або змісті та виводяться у вигляді кнопок. Натискання відкриває відповідну сторінку: список уроків або категорію матеріалів.

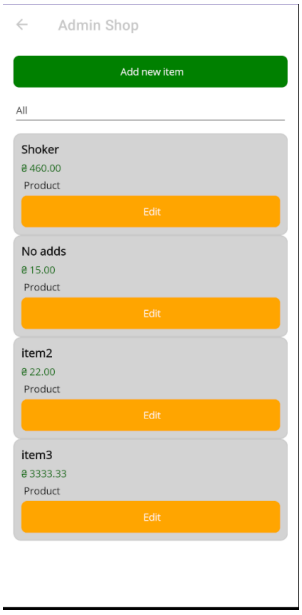

Рисунок 3.24 – Сторінка магазин(адмін)

Сторінка «AdminShopPage»(див. рис. 3.25) втілює інтерфейс керування товарами,даючи можливість переглядати,сортувати по категоріях та додавати нові позиції. «AdminEditItemPage»(див. рис. 3.26) призначена для створення та коригування товарів, де вказуються назва,опис,вартість,категорія,видимість та зображення.Додатково реалізовано керування категоріями:їхнє створення, перейменування та видалення. Вся інформація зберігається за допомогою DatabaseService, а локалізація та темаоформленнязастосовуються автоматично.

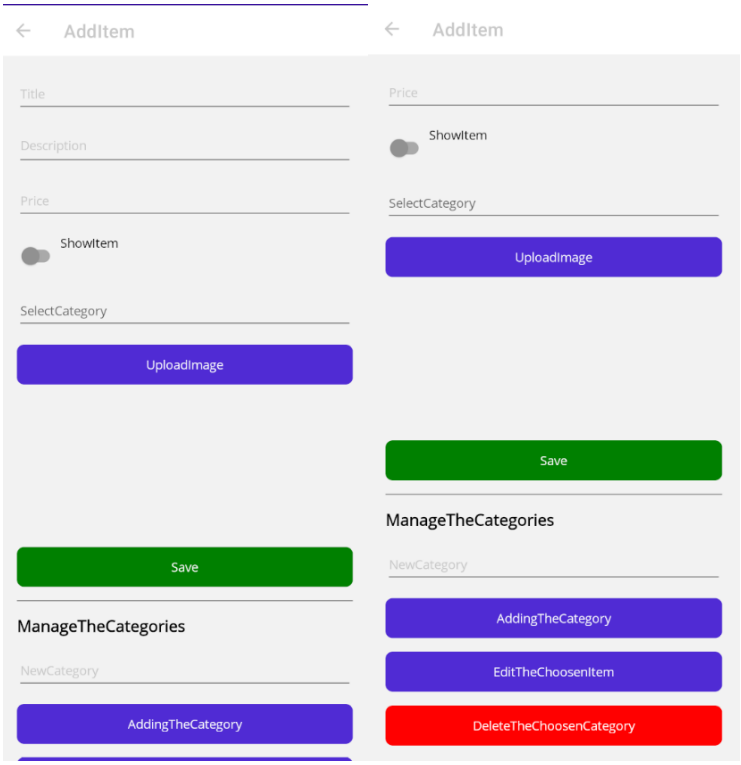

Рисунок 3.25 – Додавання товару

ShopPage(див. рис. 3.27) — сторінка магазину з фільтрацією товарів за категоріями, переглядом товарів і додаванням до кошика.

CartPage(див. рис. 3.27) — сторінка кошика з переглядом, видаленням товарів, очищенням і оформленням замовлення.

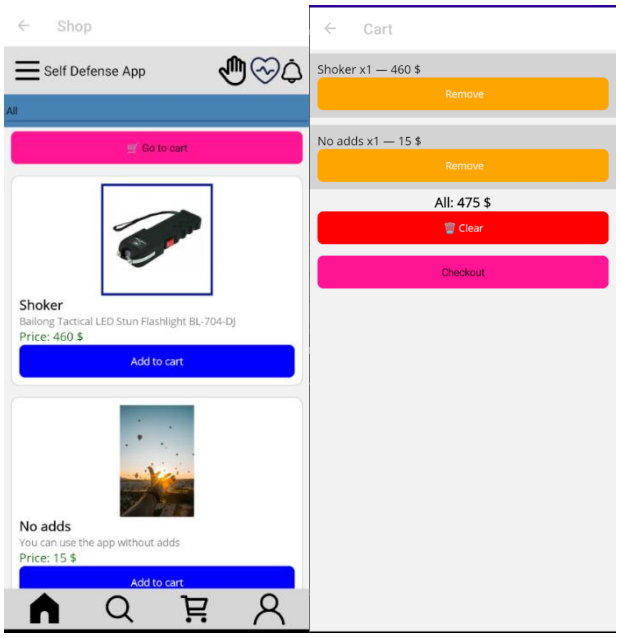

Рисунок 3.26 – Сторінка магазину та кошика(користувач)

Клас «ProfilePage» відповідає за відображення та редагування профілю користувача, взаємодіючи з базою через «UserRepository» і «DatabaseService». Сторінка має верхню та нижню навігаційні панелі і центральну частину з полями для імені, посади, зміни імені і пароля, кнопками оновлення та виходу, а також повідомленнями про результати. Дані завантажуються асинхронно зі сесії або бази. Оновлення проходить перевірку автентифікації за старим паролем, мінімальної довжини і унікальності імені. Після успіху дані зберігаються в сховищі, сесії та оновлюються в інтерфейсі.

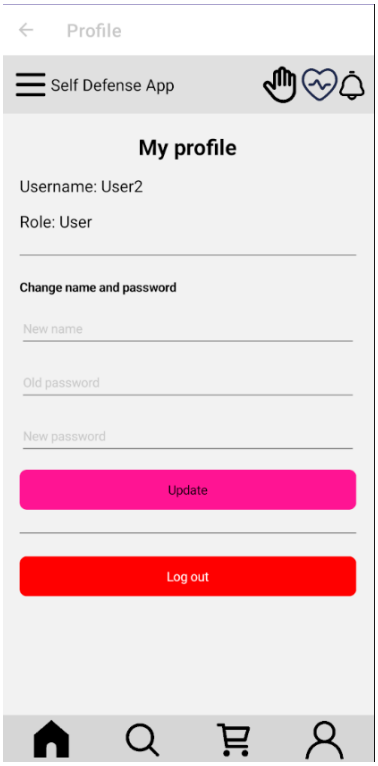

Рисунок 3.27 – Профіль користувача/адміна

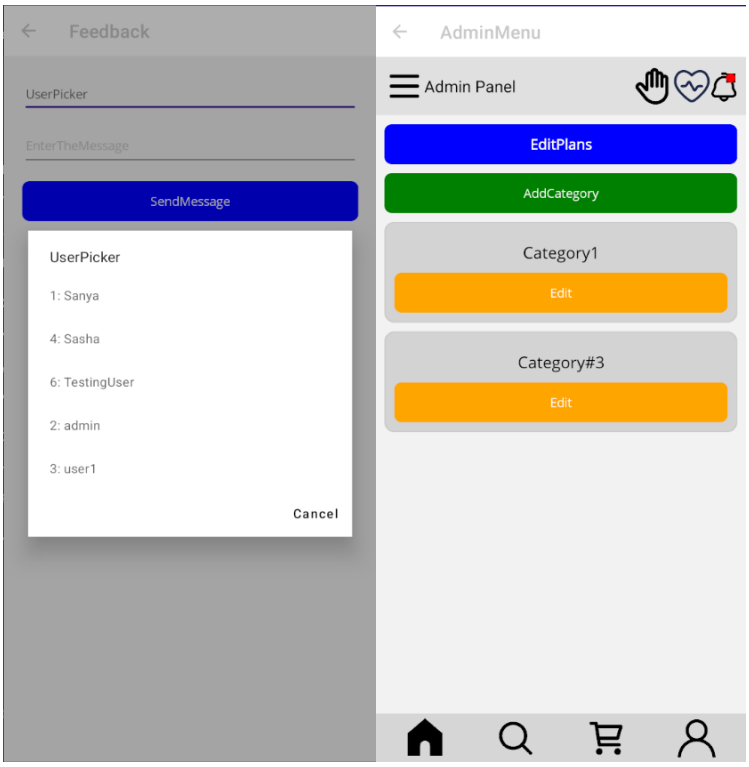

Рисунок 3.28 – Відправка та отримання повідомлення

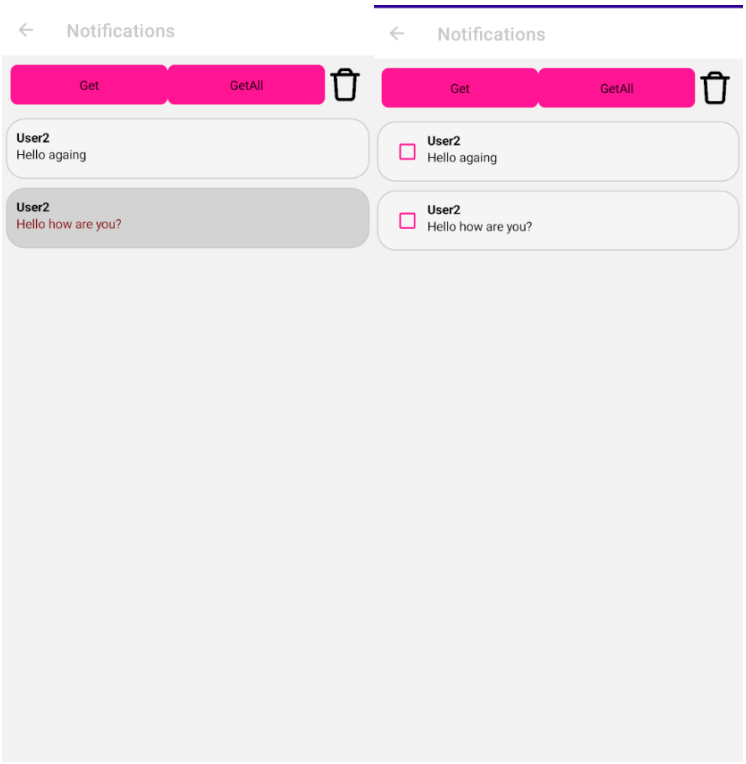

Рисунок 3.29 – Перегляд та видалення повідомлень

Сторінка MessagePage(див. рис. 3.29) дозволяє надсилати повідомлення вибраному користувачу, враховуючи локалізацію, тему й активного

користувача. NotificationsPage(див. рис. 3.30) відображає сповіщення, дозволяє їх вибирати, видаляти та відповідати (для адміністратора), автоматично позначаючи непрочитані.

Сторінка «SettingsPage»(див. рис. 3.31) відповідає за налаштування зовнішнього вигляду застосунку, дозволяючи користувачам обирати тему та мову інтерфейсу. Обрані параметри зберігаються у пам'яті пристрою та одразу застосовуються завдяки інтеграції з відповідними сервісами.

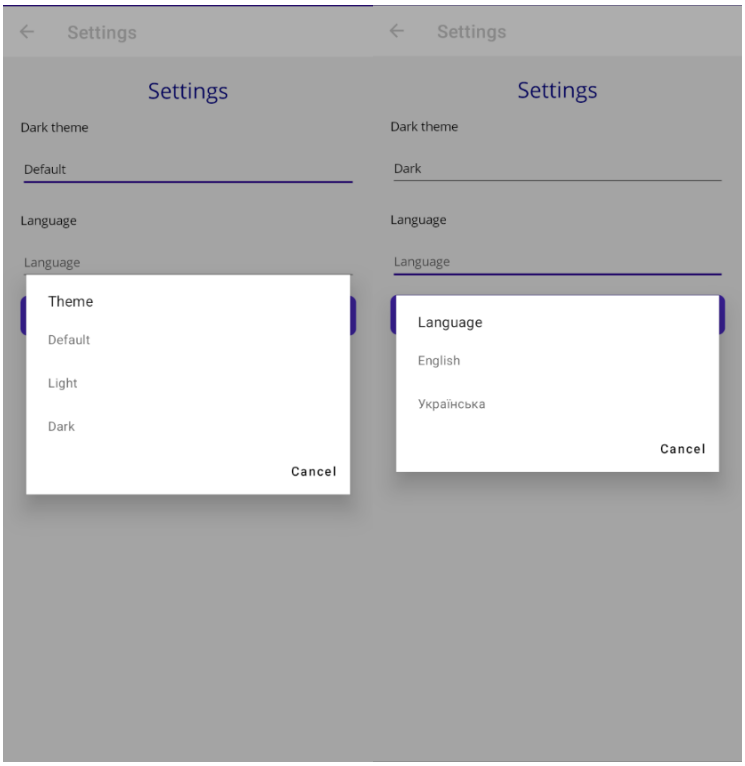

Рисунок 3.30 – Вибір теми та мови

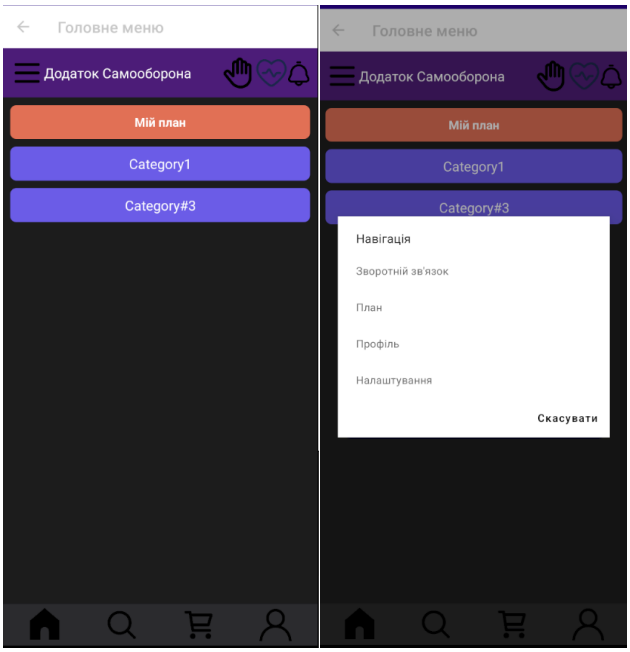

Рисунок 3.31 – Результат вибору теми(«Dark») та мови(«Українська»)

Сторінка «StatisticPage» виводить на екран статистичну інформацію з бази даних.Тут можна побачити дані про кількість елементів, категорій, а також розподіл елементів по категоріях.Додатково реалізовано навігацію, використовуючи верхню та нижню панелі.

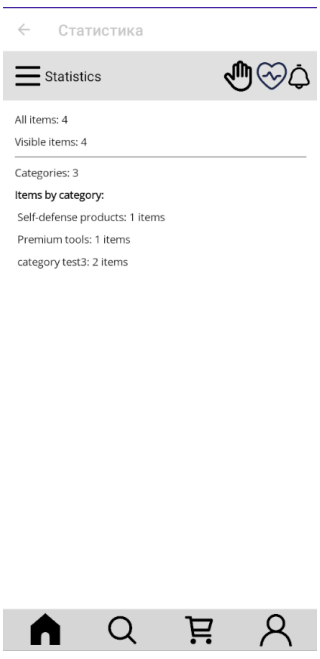

Рисунок 3.32 – Статистика

Сторінки з поширюваними питаннями(див. рис. 3.34). При натискані одного із розділів користувач переходить на сторінку з відповідним питанням,де написана порада або інструкція дій.

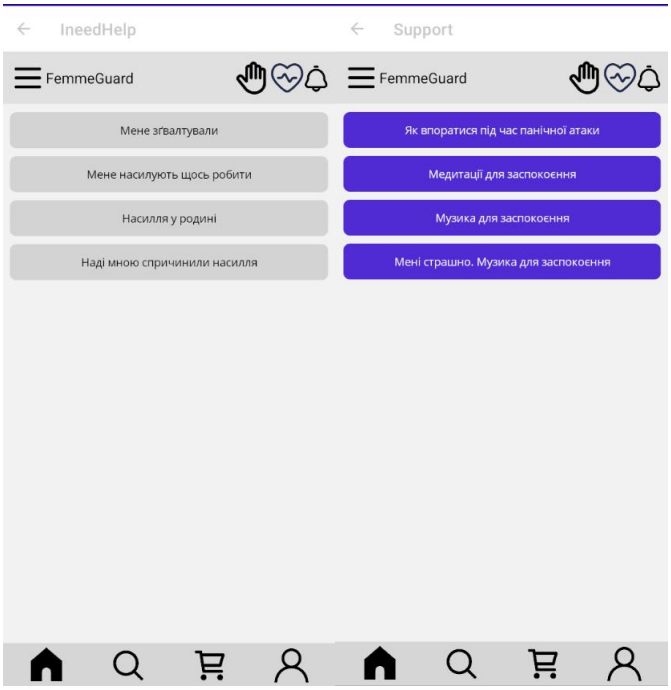

Рисунок 3.33 – Сторінки з поширюваними питаннями

3.4 Методика роботи користувача

Мобільний додаток розроблено з урахуванням принципів зручності, інтуїтивної навігації та швидкого доступу до ключових функцій, які спрямовані на покращення особистої безпеки жінок. Робота користувача з додатком стартує з головної сторінки, яка є навігаційним центром для переходу до всіх інших модулів.

Головна сторінка містить актуальні матеріали та огляд функціональних можливостей додатку. Тут користувачка бачить ключові розділи, зокрема доступ до навчальних матеріалів, новин, тренувальних планів або важливих оголошень. Відображення вмісту адаптовано до рівня доступу користувача та може змінюватися залежно від оновлень чи активності.

На сторінці пошуку реалізовано можливість швидкого знаходження матеріалів, що стосуються самозахисту, правової підтримки або технік безпеки. Пошук виконується за ключовими словами, з можливістю фільтрації за категоріями. Інтерфейс пошуку побудований з урахуванням зручності введення тексту та негайного отримання релевантних результатів.

Сторінка профілю дозволяє користувачці переглянути особисті дані, налаштування облікового запису, а також змінити певну інформацію, таку як ім’я або пароль. З профілю також доступний вихід з облікового запису та перехід до розділів зворотного зв’язку.

У налаштуваннях користувачка має змогу обрати мову інтерфейсу, змінити тему оформлення, а також керувати параметрами сповіщень. Усі зміни застосовуються відразу після підтвердження, а сам інтерфейс побудовано таким чином, щоб навіть недосвідчений користувач міг легко здійснити налаштування.

Сторінка повідомлень відображає як системні сповіщення, так і персональні повідомлення від адміністрації додатку. Непрочитані повідомлення виділяються візуально. Також реалізовано функцію перегляду деталей кожного повідомлення після натискання на нього.

У магазині користувачка може переглядати доступні товари, що стосуються тематики самозахисту, включно з фізичними товарами або тренувальними пакетами. Додаток забезпечує перегляд опису товару, зображень, ціни та можливість додавання до кошика чи оформлення замовлення.

ВИСНОВКИ

У ході кваліфікаційної роботи було створено мобільний додаток, що забезпечує жінок засобами самозахисту та взаємодії, задля покращення особистої безпеки та розуміння питань захисту від насильства. Здійснено аналіз наявних рішень, визначено функціональні потреби,спроектовано структуру та інтерфейс додатку, реалізовано механізми реєстрації та авторизації з урахуванням надійності даних. Включено модулі навчання, тренувань, сповіщень та зворотного зв'язку. Проведене тестування показало правильність роботи функціональності та відповідність вимогам безпеки.

Підсумком роботи став сучасний мобільний додаток із комфортним інтерфейсом та широкими можливостями, що сприяє збільшенню рівня самозахисту жінок. У майбутньому доцільно розширити функціональність та інтегрувати з додатковими сервісами для збільшення ефективності та зручності використання.

ПЕРЕЛІК ПОСИЛАНЬ

1. WhiteRibbonUkraine – платформа протидії насильству [Електронний ресурс]. – Режим доступу: https://whiteribbonukraine.org/ – Назва з екрана.

2. Мобільний додаток «Крила» - допомога жінкам, які потерпають від домашнього насильства [Електронний ресурс] // Уланівська сільска рада. – 2024. – Режим доступу: https://ulaniv-rada.gov.ua/news/1710492137/ – Назва з екрана.

3. Застосунки для безпеки жінок: що слід завантажити на свій телефон [Електронний ресурс] // Вікна – 2024. – Режим доступу: https://vikna.tv/dlia-tebe/bezpeka/zastosunky-dlya-bezpeky-zhinok// – Назва з екрана.

4. Всесвітня організація охорони здоров'я (ВООЗ): Гарсія-Морено К., Абрахамс Н., Девріс К., Паллітто К., Штекль Г., Воттс Ш. Глобальні та регіональні оцінки насильства щодо жінок: поширеність і наслідки для здоров'я фізичного та сексуального насильства з боку інтимного партнера та сторонніх осіб. – Женева: Всесвітня організація охорони здоров'я, 2013. – 51 с. – ISBN 978-92-4-156462-5.

5. Організація з безпеки і співробітництва в Європі (ОБСЄ): ОБСЄ. Дослідження щодо добробуту та безпеки жінок: Україна – Звіт про результати. – Варшава: Організація з безпеки і співробітництва в Європі, 2019.

6. Зростання випадків домашнього насильства під час соціально-економічної кризи та надзвичайних ситуацій UN Women. Три роки повномасштабної війни в Україні відкидають десятиліття прогресу у правах жінок, безпеці та економічних можливостях [Електронний ресурс] // UN Women – 2025. – Режим доступу: https://www.unwomen.org/en/news-stories/press-release/2025/02/three-years-of-full-scale-war-in-ukraine-roll-back-decades-of-progress-for-womens-rights-safety-and-economic-opportunities – Назва з екрана.

7. Низький рівень звернень до правоохоронних органів UNFPA. Емоційна вартість боротьби з гендерно зумовленим насильством на передовій в Україні [Електронний ресурс] // UNFPA – 2024. – Режим доступу: https://www.unfpa.org/news/emotional-cost-combating-gender-based-violence-front-line-ukraine – Назва з екрана.

8. Недостатня ефективність існуючих захисних механізмів UNFPA. Голоси з України: Огляд гендерно зумовленого насильства [Електронний ресурс] // UNFPA Ukraine – 2024. – Режим доступу: https://ukraine.unfpa.org/en/publications/voices-ukraine-brief-2024 – Назва з екрана.

9. Digital 2025: Ukraine — DataReportal – Global Digital Insights [Електронний ресурс]. – Режим доступу: https://datareportal.com/reports/digital-2025-ukraine

Додаток А – Код програми

Pages/

LoginPage.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.Maui.Controls;
using SelfeDefenseApp.Services;
using Microsoft.Maui.Storage;
using SelfeDefenseApp.Pages.Admin;

namespace SelfeDefenseApp.Pages
{

    public class LoginPage : ContentPage
    {
        private readonly UserRepository _userRepository;
        private Entry _usernameEntry;
        private Entry _passwordEntry;
        private Label _messageLabel;

        public LoginPage(UserRepository userRepository)
        {
            _userRepository = userRepository;
            Title = "Авторизація / Реєстрація";
            BackgroundColor = ThemeService.BackgroundColor;

            _usernameEntry = new Entry { Placeholder = "Ім'я користувача", Margin =
new Thickness(0, 10, 0, 10) };
            _passwordEntry = new Entry { Placeholder = "Пароль", IsPassword = true,
Margin = new Thickness(0, 0, 0, 20) };
            _messageLabel = new Label { TextColor = Colors.Red, IsVisible = false,
HorizontalOptions = LayoutOptions.Center };

            var loginButton = new Button { Text = "Вхід", Margin = new Thickness(0,
0, 0, 10) };
            var registerButton = new Button { Text = "Зареєструватись" };

            loginButton.Clicked += OnLoginClicked;
            registerButton.Clicked += OnRegisterClicked;

            Content = new VerticalStackLayout
            {
                Padding = 20,
                Spacing = 10,
                VerticalOptions = LayoutOptions.Center,
                Children =
            {
                new Label { Text = "Введіть дані для входу", FontSize = 24,
HorizontalOptions = LayoutOptions.Center, Margin = new Thickness(0, 0, 0, 20) },
                _usernameEntry,
                _passwordEntry,
                loginButton,
                registerButton,
                _messageLabel
```

```
            }
            };

            EnsureAdminExists();
        }

        private async void OnLoginClicked(object sender, EventArgs e)
        {
            try
            {
                string username = _usernameEntry?.Text ?? "";
                string password = _passwordEntry?.Text ?? "";

                if (string.IsNullOrWhiteSpace(username) ||
string.IsNullOrWhiteSpace(password))
                {
                    ShowMessage("Введіть ім'я користувача та пароль", false);
                    return;
                }

                if (_userRepository.ValidateUser(username, password))
                {
                    var user = _userRepository.GetUserByUsername(username);
                    if (user != null)
                    {
                        SessionService.UserId = user.Value.Id;
                        SessionService.Username = user.Value.Username;
                        SessionService.Role = _userRepository.GetUserRole(username)
?? "User";

                        ShowMessage("The entrance is successful!", true);

                        if (SessionService.Role == "Admin")
                            await Navigation.PushAsync(new AdminMainMenuPage());
                        else
                            await Navigation.PushAsync(new MainMenuPage());
                    }
                }
                else
                {
                    ShowMessage("Login error. Incorrect data.", false);
                }
            }
            catch (Exception ex)
            {
                ShowMessage($"Error: {ex.Message}", false);
            }
        }

        private void OnRegisterClicked(object sender, EventArgs e)
        {
            try
            {
                string username = _usernameEntry?.Text ?? "";
                string password = _passwordEntry?.Text ?? "";

                if (string.IsNullOrWhiteSpace(username) ||
string.IsNullOrWhiteSpace(password))
                {
                    ShowMessage("Введіть ім'я користувача та пароль", false);
                    return;
                }

                if (_userRepository.RegisterUser(username, password, "User"))
                {
```

```
                    ShowMessage("Реєстрація пройшла успішно! Тепер авторизуйтесь.",
true);
                }
                else
                {
                    ShowMessage("Помилка при реєстрації. Можливо, назва вже
зайнята.", false);
                }
            }
            catch (Exception ex)
            {
                ShowMessage($"Помилка: {ex.Message}", false);
            }
        }
    }


    }
```

MainMenuPage.cs:

```
using SelfeDefenseApp.Data;
using SelfeDefenseApp.Pages.General;
using SelfeDefenseApp.Services;
using System;
using System.Linq;
using Microsoft.Maui.Controls;

namespace SelfeDefenseApp.Pages
{
    public class MainMenuPage : ContentPage
    {
        private readonly DatabaseService _databaseService;

        public MainMenuPage()
        {
            _databaseService = new DatabaseService();
            Title = LocalizationService.Get("MainMenuTitle");
            BackgroundColor = ThemeService.BackgroundColor;

            var topBar = SharedUI.CreateTopBar(
                LocalizationService.Get("AppTitle"),
                async () => await Navigation.PushAsync(new NotificationsPage()),
                OnMenuClicked,
                Navigation
            );

            var categoriesStack = new VerticalStackLayout
            {
                Padding = 10,
                Spacing = 10
            };

            var planButton = new Button
            {
                Text = LocalizationService.Get("Myplan"),
                BackgroundColor = ThemeService.AccentColor,
                TextColor = ThemeService.TextColor,
                FontSize = 16,
                FontAttributes = FontAttributes.Bold,
                FontFamily = ThemeService.FontFamily
            };
            planButton.Clicked += async (s, e) =>
                await Navigation.PushAsync(new ProtectionPlanPage());

            categoriesStack.Children.Add(planButton);
```

```
            var categories = _databaseService.GetCategories();
            foreach (var category in categories)
            {
                var categoryButton = new Button
                {
                    Text = category.Title,
                    BackgroundColor = ThemeService.SecondaryColor,
                    TextColor = ThemeService.TextColor,
                    FontSize = 18,
                    FontFamily = ThemeService.FontFamily
                };

                categoryButton.Clicked += (s, e) =>
                    Navigation.PushAsync(new CategoryPage(category.Id,
category.Title));

                categoriesStack.Children.Add(categoryButton);
            }

            var scrollView = new ScrollView { Content = categoriesStack };

            var bottomNav = SharedUI.CreateBottomNav(Navigation);

            var mainGrid = new Grid
            {
                RowDefinitions =
                {
                    new RowDefinition { Height = GridLength.Auto },
                    new RowDefinition { Height = GridLength.Star },
                    new RowDefinition { Height = GridLength.Auto }
                }
            };

            mainGrid.Children.Add(topBar); Grid.SetRow(topBar, 0);
            mainGrid.Children.Add(scrollView); Grid.SetRow(scrollView, 1);
            mainGrid.Children.Add(bottomNav); Grid.SetRow(bottomNav, 2);

            Content = mainGrid;
            SharedUI.SetPageStyle(this);
        }


    }
}
```

AdminMainMenuPage.cs:

```
using SelfeDefenseApp.Data;
using SelfeDefenseApp.Pages;
using SelfeDefenseApp.Pages.General;
using SelfeDefenseApp.Services;
using Microsoft.Maui.Controls;
using System;
using System.Linq;

namespace SelfeDefenseApp.Pages.Admin
{
    public class AdminMainMenuPage : ContentPage
    {
        private readonly DatabaseService _databaseService;

        public AdminMainMenuPage()
        {
            LocalizationService.LanguageChanged += (_, _) =>
LocalizationService.ApplyLocalization(this);
```

```
            ThemeService.ThemeChanged += (_, _) => ThemeService.ApplyTheme(this);

            ThemeService.ApplyTheme(this);


            _databaseService = new DatabaseService();
            Title = LocalizationService.Get("AdminMenu");


            bool hasUnread =
_databaseService.HasUnreadNotifications(SessionService.UserId);


            var topBar = SharedUI.CreateTopBar("Admin Panel",
        async () => await Navigation.PushAsync(new NotificationsPage()),
        OnMenuClicked,
        Navigation);

            SharedUI.UpdateNotificationDot(hasUnread);


            var categoriesStack = new VerticalStackLayout { Padding = 10, Spacing =
10 };

            var plansButton = new Button
            {
                Text = LocalizationService.Get("EditPlans"),
                BackgroundColor = Colors.Blue,
                FontAttributes = FontAttributes.Bold,
                FontSize = 16
            };
            plansButton.Clicked += async (s, e) => await Navigation.PushAsync(new
AdminPlansPage());

            var addCategoryButton = new Button
            {
                Text = LocalizationService.Get("AddCategory"),
                BackgroundColor = Colors.Green,

            };
            addCategoryButton.Clicked += (s, e) => Navigation.PushAsync(new
AdminCreateCategoryPage());

            categoriesStack.Children.Add(plansButton);
            categoriesStack.Children.Add(addCategoryButton);

            var categories = _databaseService.GetCategories();
            foreach (var category in categories)
            {
                var editButton = new Button { Text =
LocalizationService.Get("Edit"), BackgroundColor = Colors.Orange,   };
                editButton.Clicked += (s, e) => Navigation.PushAsync(new
AdminCategoryPage(category.Id));

                var categoryButton = new Button
                {
                    Text = category.Title,
                    BackgroundColor = Colors.LightGray,
                    TextColor = Colors.Black,
                    FontSize = 18
                };
                categoryButton.Clicked += (s, e) => Navigation.PushAsync(new
CategoryPage(category.Id, category.Title));

                categoriesStack.Children.Add(new Frame
```

```
                {
                    Content = new VerticalStackLayout { Children = { categoryButton,
editButton } },
                    BackgroundColor = Colors.LightGray,
                    CornerRadius = 10,
                    Padding = 10
                });
            }


            var scroll = new ScrollView { Content = categoriesStack };


            var bottomNav = SharedUI.CreateBottomNav(Navigation);

            var mainGrid = new Grid
            {
                RowDefinitions =
                {
                    new RowDefinition { Height = GridLength.Auto },
                    new RowDefinition { Height = GridLength.Star },
                    new RowDefinition { Height = GridLength.Auto }
                }
            };

            mainGrid.Children.Add(topBar); Grid.SetRow(topBar, 0);
            mainGrid.Children.Add(scroll); Grid.SetRow(scroll, 1);
            mainGrid.Children.Add(bottomNav); Grid.SetRow(bottomNav, 2);

            Content = mainGrid;
        }
    }
}
```

AdminPlanEditPage.cs:

```
using SelfeDefenseApp.Data;
using SelfeDefenseApp.Services;
using System;

namespace SelfeDefenseApp.Pages.Admin
{
    public class AdminPlanEditPage : ContentPage
    {
        private readonly DatabaseService _databaseService;
        private readonly int _planId;
        private readonly Entry _titleEntry;
        private readonly Editor _descriptionEditor;
        private readonly Button _saveButton;

        public AdminPlanEditPage(int planId = 0, string title = "", string
description = "")
        {
            LocalizationService.LanguageChanged += (_, _) =>
LocalizationService.ApplyLocalization(this);
            ThemeService.ThemeChanged += (_, _) => ThemeService.ApplyTheme(this);

            ThemeService.ApplyTheme(this);

            _databaseService = new DatabaseService();
            _planId = planId;

            Title = _planId == 0 ? "Create Plan" : "Edit Plan";


            _titleEntry = new Entry { Text = title, Placeholder = "Plan title" };
```

```
            _descriptionEditor = new Editor { Text = description, Placeholder =
"Plan description", AutoSize = EditorAutoSizeOption.TextChanges };
            _saveButton = new Button { Text = "💾 Save", BackgroundColor =
Colors.Green,   };

            _saveButton.Clicked += SaveButton_Clicked;

            Content = new ScrollView
            {
                Content = new VerticalStackLayout
                {
                    Padding = 20,
                    Spacing = 15,
                    Children = { _titleEntry, _descriptionEditor, _saveButton }
                }
            };
        }



    }
}
```

ProtectionPlanPage.cs:

```
using SelfeDefenseApp.Data;
using SelfeDefenseApp.Services;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SelfeDefenseApp.Pages
{
    public class ProtectionPlanPage : ContentPage
    {
        private readonly DatabaseService _databaseService;
        int userId = SessionService.UserId;

        public ProtectionPlanPage()
        {
            _databaseService = new DatabaseService();
            Title = LocalizationService.Get("Plan");
            BackgroundColor = ThemeService.BackgroundColor;

            var plansStack = new VerticalStackLayout { Padding = 10, Spacing = 10 };

            var plans = _databaseService.GetAllPlans();

            foreach (var plan in plans)
            {
                var openPlanButton = new Button
                {
                    Text = plan.Title,
                    BackgroundColor = ThemeService.SecondaryColor,
                    TextColor = ThemeService.TextColor,
                    FontSize = 18,
                    FontFamily = ThemeService.FontFamily
                };

                openPlanButton.Clicked += (s, e) =>
                {
                    int userId = SessionService.UserId;
                    Navigation.PushAsync(new UserLessonListPage(plan.Id, userId,
plan.Title));
```

```
            };

            plansStack.Children.Add(openPlanButton);
        }

        Content = new ScrollView { Content = plansStack };
    }
}
}
```

AdminLessonEditPage.cs:

```
using SelfeDefenseApp.Data;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SelfeDefenseApp.Pages.Admin
{
    public class AdminLessonEditPage : ContentPage
    {
        private readonly DatabaseService _databaseService;
        private readonly int _planId;
        private readonly int _lessonId;

        private Entry _titleEntry;
        private Editor _contentEditor;
        private Button _uploadImageButton;
        private Button _uploadSliderButton;
        private StackLayout _previewStack;
        private List<string> _imagePaths = new();

        public AdminLessonEditPage(int planId, int lessonId = 0, string title = "",
string content = "", string mediaUrl = "")
        {
            _planId = planId;
            _lessonId = lessonId;
            _databaseService = new DatabaseService();

            Title = lessonId == 0 ? "Add Lesson" : "Edit Lesson";
            BackgroundColor = Colors.White;

            _titleEntry = new Entry { Text = title, Placeholder = "Lesson Title" };
            _contentEditor = new Editor { Text = content, Placeholder = "Content",
AutoSize = EditorAutoSizeOption.TextChanges };

            _uploadImageButton = new Button { Text = "Upload Single Image" };
            _uploadImageButton.Clicked += UploadImageButton_Clicked;

            _uploadSliderButton = new Button { Text = "Upload Slider (Multiple
Images)" };
            _uploadSliderButton.Clicked += UploadSliderButton_Clicked;

            _previewStack = new StackLayout { Spacing = 10 };

            if (!string.IsNullOrWhiteSpace(mediaUrl))
            {
                _imagePaths = mediaUrl.Split(';',
StringSplitOptions.RemoveEmptyEntries).ToList();
                ShowPreview();
            }
```

```
            var saveButton = new Button { Text = "🖫 Save", BackgroundColor =
Colors.Green, TextColor = Colors.White };
            saveButton.Clicked += SaveButton_Clicked;

            Content = new ScrollView
            {
                Content = new VerticalStackLayout
                {
                    Padding = 20,
                    Spacing = 15,
                    Children =
                    {
                        _titleEntry,
                        _contentEditor,
                        _uploadImageButton,
                        _uploadSliderButton,
                        _previewStack,
                        saveButton
                    }
                }
            };
        }


    }
}
```

UserLessonPage.cs:

```
using SelfeDefenseApp.Data;
using SelfeDefenseApp.Pages.General;
using SelfeDefenseApp.Services;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SelfeDefenseApp.Pages
{
    public class UserLessonPage : ContentPage
    {
        private readonly DatabaseService _databaseService;
        private readonly int _lessonId;
        private readonly int _userId = SessionService.UserId;

        public UserLessonPage(int lessonId)
        {
            _lessonId = lessonId;
            _databaseService = new DatabaseService();

            LocalizationService.LanguageChanged += (_, _) =>
LocalizationService.ApplyLocalization(this);
            ThemeService.ThemeChanged += (_, _) => ThemeService.ApplyTheme(this);

            ThemeService.ApplyTheme(this);

            _lessonId = lessonId;
            _databaseService = new DatabaseService();

            Title = LocalizationService.Get("Lesson");

            var topBar = SharedUI.CreateTopBar("FemmeGuard",
    async () => await Navigation.PushAsync(new NotificationsPage()),
    OnMenuClicked,
```

```
    Navigation);
            var bottomNav = SharedUI.CreateBottomNav(Navigation);


            var lesson = _databaseService.GetLessonById(lessonId);

            var titleLabel = new Label
            {
                Text = lesson.Title,
                FontSize = 20,
                FontAttributes = FontAttributes.Bold,
                TextColor = Colors.Black
            };

            var contentLabel = new Label
            {
                Text = lesson.Content,
                FontSize = 16,
                TextColor = Colors.Black
            };

            var layout = new VerticalStackLayout
            {
                Padding = 20,
                Spacing = 15,
                Children = { titleLabel, contentLabel }
            };


            var isCompleted = _databaseService.IsLessonCompleted(_userId,
_lessonId);

            var completeCheckbox = new CheckBox
            {
                IsChecked = isCompleted,
                Color = Colors.Green
            };

            var checkboxLabel = new Label
            {
                Text = LocalizationService.Get("Completed"),
                VerticalOptions = LayoutOptions.Center
            };

            completeCheckbox.CheckedChanged += async (s, e) =>
            {
                if (e.Value)
                    _databaseService.MarkLessonCompleted(_userId, _lessonId);
                else
                    _databaseService.UnmarkLessonCompleted(_userId, _lessonId);
                await Navigation.PopAsync();
            };


            layout.Children.Add(new HorizontalStackLayout
            {
                Children = { completeCheckbox, checkboxLabel },
                Spacing = 10
            });


            int? nextId = _databaseService.GetNextLessonId(_lessonId);
```

```
            int? prevId = _databaseService.GetPreviousLessonId(_lessonId);


            layout.Children.Add(new Label
            {
                Text = LocalizationService.Get("Comments"),
                FontAttributes = FontAttributes.Bold,
                FontSize = 18,
                TextColor = Colors.Black
            });

            var commentEntry = new Entry { Placeholder =
LocalizationService.Get("WriteTheComment") };
            var commentButton = new Button
            {
                Text = LocalizationService.Get("SendMessage"),
                BackgroundColor = Colors.Blue,

            };

            var commentsStack = new VerticalStackLayout { Spacing = 10 };
            layout.Children.Add(commentEntry);
            layout.Children.Add(commentButton);
            layout.Children.Add(commentsStack);
            LoadComments();

            var grid = new Grid
            {
                RowDefinitions =
    {
        new RowDefinition { Height = GridLength.Auto },
        new RowDefinition { Height = GridLength.Star },
        new RowDefinition { Height = GridLength.Auto }
    }
            };

            grid.Children.Add(topBar); Grid.SetRow(topBar, 0);
            grid.Children.Add(layout); Grid.SetRow(layout, 1);
            grid.Children.Add(bottomNav); Grid.SetRow(bottomNav, 2);

            Content = grid;



            Content = new ScrollView { Content = layout };
        }

    }

}
```

AdminCategoryPage.cs:

```
using SelfeDefenseApp.Data;
using SelfeDefenseApp.Services;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SelfeDefenseApp.Pages.Admin
{
    public class AdminCategoryPage : ContentPage
    {
        private readonly ContentService _contentService;
```

```
        private readonly DatabaseService _databaseService;
        private readonly int _categoryId;

        private Entry _titleEntry;
        private VerticalStackLayout _contentLayout;

        public AdminCategoryPage(int categoryId)
        {
            LocalizationService.LanguageChanged += (_, _) =>
LocalizationService.ApplyLocalization(this);
            ThemeService.ThemeChanged += (_, _) => ThemeService.ApplyTheme(this);

            ThemeService.ApplyTheme(this);


            _categoryId = categoryId;
            _databaseService = new DatabaseService();
            _contentService = new ContentService(DatabaseService.GetDatabasePath());

            Title = LocalizationService.Get("EditCategory");


            _titleEntry = new Entry { Placeholder =
LocalizationService.Get("CategoryTitle") };

            var saveTitleButton = new Button { Text =
LocalizationService.Get("SaveTitle"), BackgroundColor = Colors.Gold,   };
            saveTitleButton.Clicked += async (s, e) => await
SaveTitleButton_Clicked();

            var deleteButton = new Button { Text =
LocalizationService.Get("DeleteCategory"), BackgroundColor = Colors.Red,   };
            deleteButton.Clicked += async (s, e) =>
            {
                bool confirm = await
DisplayAlert(LocalizationService.Get("Confirm"),
LocalizationService.Get("AreYouSureYouWantToDeleteThisCategory"),
LocalizationService.Get("Yes"), LocalizationService.Get("No"));
                if (confirm)
                {
                    _databaseService.DeleteCategory(_categoryId);
                    await Navigation.PopAsync();
                }
            };

            var addImageButton = new Button { Text = "Add image", BackgroundColor =
Colors.Blue,   };
            addImageButton.Clicked += (s, e) => Navigation.PushAsync(new
AdminVideoPage(DatabaseService.GetDatabasePath(), _categoryId));

            var addTextButton = new Button { Text = "Add text", BackgroundColor =
Colors.Green,   };
            addTextButton.Clicked += (s, e) => Navigation.PushAsync(new
AdminTextPage(DatabaseService.GetDatabasePath(), _categoryId));

            _contentLayout = new VerticalStackLayout { Padding = 10, Spacing = 15 };

            Content = new ScrollView
            {
                Content = new VerticalStackLayout
                {
                    Padding = 20,
                    Spacing = 15,
                    Children =
                    {
```

```
                        _titleEntry,
                        saveTitleButton,
                        deleteButton,
                        addImageButton,
                        addTextButton,
                        _contentLayout
                    }
                }
            };

            _ = LoadCategoryAsync();
        }

        private async Task LoadCategoryAsync()
        {
            var currentCategory = _databaseService.GetCategoryById(_categoryId);
            if (currentCategory == null)
            {
                await DisplayAlert(LocalizationService.Get("Error"),
LocalizationService.Get("TheCategoryIsNotFound"), "OK");
                await Navigation.PopAsync();
                return;
            }

            _titleEntry.Text = currentCategory.Title;
            LoadContent();
        }

        private async Task SaveTitleButton_Clicked()
        {
            var newTitle = _titleEntry.Text;
            if (!string.IsNullOrWhiteSpace(newTitle))
            {
                _databaseService.UpdateCategoryTitle(_categoryId, newTitle);
                await DisplayAlert(LocalizationService.Get("Saved"),
LocalizationService.Get("CategoryTitleUpdated"), "OK");
            }
            else
            {
                await DisplayAlert(LocalizationService.Get("Error"),
LocalizationService.Get("EnterTheTitle"), "OK");
            }
        }
    }


}
```

CategoryPage.cs:

```
using Microsoft.Maui.Controls;
using Microsoft.Maui.Controls.Xaml;
using SelfeDefenseApp.Controls;
using SelfeDefenseApp.Data;
using SelfeDefenseApp.Pages.Admin;
using SelfeDefenseApp.Pages.General;
using SelfeDefenseApp.Services;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SelfeDefenseApp.Pages
{
    public class CategoryPage : ContentPage
```

```
    {
        private readonly int _categoryId;
        private readonly string _categoryTitle;
        private readonly DatabaseService _databaseService;
        private readonly bool _isAdmin;
        private int _currentPage = 1;
        private const int ItemsPerPage = 5;
        private readonly ContentService _contentService;
        private VerticalStackLayout _contentLayout;
        private VerticalStackLayout _videosStack;
        private Button _prevButton;
        private Button _nextButton;

        public CategoryPage(int categoryId, string categoryTitle, bool isAdmin =
false)
        {
            _categoryId = categoryId;
            _categoryTitle = categoryTitle;
            _databaseService = new DatabaseService();
            _isAdmin = isAdmin;
            _contentService = new ContentService(DatabaseService.GetDatabasePath());
            Title = LocalizationService.Get("CategoryTitle") + ": " +
_categoryTitle;
            BackgroundColor = ThemeService.BackgroundColor;

            var topBar = SharedUI.CreateTopBar(
                LocalizationService.Get("AppTitle"),
                async () => await Navigation.PushAsync(new NotificationsPage()),
                OnMenuClicked,
                Navigation);
            var bottomNav = SharedUI.CreateBottomNav(Navigation);


            _videosStack = new VerticalStackLayout { Padding = 10, Spacing = 10 };
            _prevButton = new Button { Text = LocalizationService.Get("Previous"),
IsEnabled = false };
            _nextButton = new Button { Text = LocalizationService.Get("Next") };
            _prevButton.Clicked += (s, e) => ChangePage(-1);
            _nextButton.Clicked += (s, e) => ChangePage(1);

            var navigationButtons = new HorizontalStackLayout { Children = {
_prevButton, _nextButton }, Spacing = 10 };

            var mainLayout = new VerticalStackLayout { Padding = 10, Children = {
_videosStack, navigationButtons } };
            _contentLayout = new VerticalStackLayout { Padding = 10 };
            mainLayout.Children.Add(_contentLayout);

            var grid = new Grid
            {
                RowDefinitions =
    {
        new RowDefinition { Height = GridLength.Auto },
        new RowDefinition { Height = GridLength.Star },
        new RowDefinition { Height = GridLength.Auto }
    }
            };

            grid.Children.Add(topBar); Grid.SetRow(topBar, 0);
            grid.Children.Add(mainLayout); Grid.SetRow(mainLayout, 1);
            grid.Children.Add(bottomNav); Grid.SetRow(bottomNav, 2);

            Content = grid;
```

```
            Content = new ScrollView { Content = mainLayout };

            LoadVideos();
            LoadContent();
        }
    }

}
```

SearchPage.cs:

```
using SelfeDefenseApp.Data;
using SelfeDefenseApp.Services;
using Microsoft.Maui.Controls;
using System;
using System.Linq;
using SelfeDefenseApp.Pages.General;

namespace SelfeDefenseApp.Pages
{
    public class SearchPage : ContentPage
    {
        private readonly Entry _searchEntry;
        private readonly StackLayout _resultsLayout;
        private readonly DatabaseService _databaseService;

        public SearchPage()
        {
            Title = LocalizationService.Get("SearchTitle");
            BackgroundColor = ThemeService.BackgroundColor;

            _databaseService = new DatabaseService();

            _searchEntry = new Entry
            {
                Placeholder = LocalizationService.Get("SearchPlaceholder"),
                BackgroundColor = ThemeService.SecondaryColor,
                TextColor = ThemeService.TextColor,
                FontFamily = ThemeService.FontFamily
            };

            var searchButton = new Button
            {
                Text = LocalizationService.Get("Search"),
                BackgroundColor = ThemeService.AccentColor,
                TextColor = ThemeService.TextColor,
                FontFamily = ThemeService.FontFamily
            };
            searchButton.Clicked += SearchButton_Clicked;

            _resultsLayout = new StackLayout { Padding = 10, Spacing = 10 };

            Content = new ScrollView
            {
                Content = new VerticalStackLayout
                {
                    Padding = 20,
                    Children = { _searchEntry, searchButton, _resultsLayout }
                }
            };

            SharedUI.SetPageStyle(this);
        }


    }
```

```
}
```

ShopPage.cs:

```
using Microsoft.Maui.Controls;
using SelfeDefenseApp.Data;
using SelfeDefenseApp.Pages.General;
using SelfeDefenseApp.Services;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http;

namespace SelfeDefenseApp.Pages
{
    public class ShopPage : ContentPage
    {
        private readonly DatabaseService _db;
        private readonly StackLayout _productList;
        private Picker _categoryFilter;

        public ShopPage()
        {
            _db = new DatabaseService();
            Title = LocalizationService.Get("ShopTitle");
            BackgroundColor = ThemeService.BackgroundColor;


            var topBar = SharedUI.CreateTopBar(
    LocalizationService.Get("AppTitle"),
    async () => await Navigation.PushAsync(new NotificationsPage()),
    OnMenuClicked,
    Navigation);


            _categoryFilter = new Picker
            {
                Title = LocalizationService.Get("CategoryTitle"),
                TextColor = ThemeService.TextColor,
                FontFamily = ThemeService.FontFamily,
                BackgroundColor = ThemeService.SecondaryColor
            };
            _categoryFilter.Items.Add(LocalizationService.Get("All"));


            foreach (var (id, title) in _db.GetShopCategories())
                _categoryFilter.Items.Add(title);

            _categoryFilter.SelectedIndexChanged += (s, e) => LoadProducts();

            var goToCartButton = new Button
            {
                Text = $"🛒 {LocalizationService.Get("GoToCart")}",
                BackgroundColor = ThemeService.AccentColor,
                TextColor = ThemeService.TextColor,
                FontFamily = ThemeService.FontFamily,
                Margin = new Thickness(10, 5)
            };
            goToCartButton.Clicked += async (s, e) => await Navigation.PushAsync(new
CartPage());


            _productList = new StackLayout { Spacing = 15, Padding = 10 };
            _productList.BackgroundColor = ThemeService.BackgroundColor;
```

```
            var bottomNav = SharedUI.CreateBottomNav(Navigation);

            var grid = new Grid
            {
                RowDefinitions =
            {
                new RowDefinition { Height = GridLength.Auto },
                new RowDefinition { Height = GridLength.Star },
                new RowDefinition { Height = GridLength.Auto }
            }
            };

            Content = new VerticalStackLayout
            {
                Children =
                {
                    _categoryFilter,
                    goToCartButton,
                    new ScrollView { Content = _productList }
                }
            };

            grid.Children.Add(topBar); Grid.SetRow(topBar, 0);
            grid.Children.Add(Content); Grid.SetRow(Content, 1);
            grid.Children.Add(bottomNav); Grid.SetRow(bottomNav, 2);

            Content = grid;

            SharedUI.SetPageStyle(this);


            _categoryFilter.SelectedIndex = 0;
        }


        }



    }
}
```

CartPage.cs:

```
using Microsoft.Maui.Controls;
using SelfeDefenseApp.Data;
using SelfeDefenseApp.Pages.General;
using SelfeDefenseApp.Services;
using System;
using System.Linq;

namespace SelfeDefenseApp.Pages
{
    public class CartPage : ContentPage
    {
        private readonly DatabaseService _db = new();
        private readonly StackLayout _cartLayout = new();

        public CartPage()
        {
            Title = LocalizationService.Get("CartTitle");
            BackgroundColor = ThemeService.BackgroundColor;

            var topBar = SharedUI.CreateTopBar(
```

```
                LocalizationService.Get("AppTitle"),
                async () => await Navigation.PushAsync(new NotificationsPage()),
                OnMenuClicked,
                Navigation);

            var scroll = new ScrollView { Content = _cartLayout };

            var checkoutButton = new Button
            {
                Text = LocalizationService.Get("Checkout"),
                BackgroundColor = ThemeService.AccentColor,
                TextColor = ThemeService.TextColor,
                Margin = new Thickness(10, 5),
                FontFamily = ThemeService.FontFamily
            };

            checkoutButton.Clicked += CheckoutButton_Clicked;

            var clearButton = new Button
            {
                Text = "🗑 Clear",
                BackgroundColor = Colors.Red,
                TextColor = Colors.White,
                Margin = new Thickness(10, 0, 10, 10)
            };
            clearButton.Clicked += async (_, _) =>
            {
                CartService.ClearCart();
                LoadCart();
                await DisplayAlert("Cart", "Everything is cleared", "OK");
            };
            var bottomNav = SharedUI.CreateBottomNav(Navigation);

            var grid = new Grid
            {
                RowDefinitions =
        {
            new RowDefinition { Height = GridLength.Auto },
            new RowDefinition { Height = GridLength.Star },
            new RowDefinition { Height = GridLength.Auto }
        }
            };

            grid.Children.Add(topBar); Grid.SetRow(topBar, 0);
            grid.Children.Add(new VerticalStackLayout { Children = { scroll,
checkoutButton } }); Grid.SetRow(scroll, 1);
            grid.Children.Add(bottomNav); Grid.SetRow(bottomNav, 2);


            Content = new VerticalStackLayout
            {
                Children = { scroll, clearButton, checkoutButton }
            };

            LoadCart();
        }



    }
}
```

AdminShopPage.cs:

```
using Microsoft.Maui.Controls;
```

```
using SelfeDefenseApp.Data;
using System;
using System.Linq;
using System.Collections.Generic;

namespace SelfeDefenseApp.Pages.Admin
{
    public class AdminShopPage : ContentPage
    {
        private readonly DatabaseService _db = new();
        private readonly Picker _categoryPicker = new();
        private readonly VerticalStackLayout _itemStack = new();

        public AdminShopPage()
        {
            Title = " Admin Shop";
            BackgroundColor = Colors.White;

            var addButton = new Button
            {
                Text = " Add new item",
                BackgroundColor = Colors.Green,

            };
            addButton.Clicked += async (s, e) => await Navigation.PushAsync(new
AdminEditItemPage());

            _categoryPicker.Title = "Filter by category";
            _categoryPicker.Items.Add("All");
            var categoryMap = new Dictionary<string, int>();
            foreach (var (id, title) in _db.GetShopCategories())
            {
                _categoryPicker.Items.Add(title);
                categoryMap[title] = id;
            }
            _categoryPicker.SelectedIndexChanged += (s, e) =>
            {
                LoadItems(_categoryPicker.SelectedIndex > 0 ?
categoryMap[_categoryPicker.SelectedItem.ToString()] : null);
            };



            var scroll = new ScrollView { Content = _itemStack };

            Content = new VerticalStackLayout
            {
                Padding = 15,
                Spacing = 10,
                Children =
                {
                    addButton,
                    _categoryPicker,
                    scroll
                }
            };

            _categoryPicker.SelectedIndex = 0;
        }



    }
}
```

AdminEditItemPage.cs:

```
using Microsoft.Maui.Controls;
using Microsoft.Maui.Storage;
using SelfeDefenseApp.Data;
using SelfeDefenseApp.Models;
using SelfeDefenseApp.Services;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;

namespace SelfeDefenseApp.Pages.Admin
{
    public class AdminEditItemPage : ContentPage
    {
        private readonly DatabaseService _db = new();
        private readonly ShopItem _item;

        private Entry _titleEntry;
        private Editor _descriptionEditor;
        private Entry _priceEntry;
        private Picker _categoryPicker;
        private Switch _visibleSwitch;
        private Image _previewImage;
        private string _imagePath;

        private Dictionary<string, int> categoryMap = new();

        public AdminEditItemPage(ShopItem item = null)
        {
            LocalizationService.LanguageChanged += (_, _) =>
LocalizationService.ApplyLocalization(this);
            ThemeService.ThemeChanged += (_, _) => ThemeService.ApplyTheme(this);

            ThemeService.ApplyTheme(this);
```

```
        _item = item;
        Title = item == null ? LocalizationService.Get("AddItem") :
LocalizationService.Get("EditItem");

        _titleEntry = new Entry { Placeholder = LocalizationService.Get("Title"), Text =
item?.Title ?? "" };
        _descriptionEditor = new Editor { Placeholder =
LocalizationService.Get("Description"), Text = item?.Description ?? "", AutoSize =
EditorAutoSizeOption.TextChanges };
        _priceEntry = new Entry { Placeholder = LocalizationService.Get("Price"),
Keyboard = Keyboard.Numeric, Text = item?.Price.ToString("F2") ?? "" };

        _visibleSwitch = new Switch { IsToggled = item?.IsVisible == 1 };
        var visibleLabel = new Label { Text = LocalizationService.Get("ShowItem") };

        _previewImage = new Image { HeightRequest = 150, Aspect = Aspect.AspectFit
};
        if (!string.IsNullOrEmpty(item?.ImagePath))
        {
            _imagePath = item.ImagePath;
            _previewImage.Source = ImageSource.FromFile(_imagePath);
        }


        _categoryPicker = new Picker { Title = LocalizationService.Get("SelectCategory")
};
        RefreshCategories();

        if (item != null)
        {
            var selectedCategory = _db.GetShopCategories().FirstOrDefault(c => c.Id ==
item.CategoryId);
            if (selectedCategory != default)
                _categoryPicker.SelectedItem = selectedCategory.Title;
```

```
        }

        var saveButton = new Button { Text = LocalizationService.Get("Save"), BackgroundColor = Colors.Green, };
        saveButton.Clicked += async (s, e) =>
        {
            if (string.IsNullOrWhiteSpace(_titleEntry.Text) ||
                string.IsNullOrWhiteSpace(_priceEntry.Text) ||
                _categoryPicker.SelectedItem == null ||
                string.IsNullOrEmpty(_imagePath))
            {
                await DisplayAlert(LocalizationService.Get("Error"), "Fill all fields and select image.", LocalizationService.Get("OK"));
                return;
            }


        var categoryManagementLabel = new Label
        {
            Text = LocalizationService.Get("ManageTheCategories"),
            FontAttributes = FontAttributes.Bold,
            FontSize = 18
        };

        var newCategoryEntry = new Entry { Placeholder = LocalizationService.Get("NewCategory") };
        var addCategoryButton = new Button { Text = LocalizationService.Get("AddingTheCategory") };
                string selectedTitle = selectedItem.ToString();
        };
        };

        Content = new ScrollView { Content = layout };
    }
```

```
    }
}
```

ProfilePage.cs:

```
using SelfeDefenseApp.Data;
using SelfeDefenseApp.Pages.General;
using SelfeDefenseApp.Services;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SelfeDefenseApp.Pages
{
    public class ProfilePage : ContentPage
    {
        private readonly UserRepository _userRepository;
        private Label _usernameLabel;
        private Label _roleLabel;
        private Entry _oldPasswordEntry;
        private Entry _newPasswordEntry;
        private Entry _newUsernameEntry;
        private Label _messageLabel;

        public ProfilePage()
        {
            _userRepository = new UserRepository(new DatabaseService());
            Title = LocalizationService.Get("Profile");
            BackgroundColor = ThemeService.BackgroundColor;

            InitializeUI();
            LoadProfile();
        }

        private void InitializeUI()
```

```
{
    var topBar = SharedUI.CreateTopBar(
        LocalizationService.Get("AppTitle"),
        async () => await Navigation.PushAsync(new NotificationsPage()),
        OnMenuClicked,
        Navigation
    );

    _usernameLabel = new Label
    {
        FontSize = 18,
        TextColor = ThemeService.TextColor,
        FontFamily = ThemeService.FontFamily
    };

    _roleLabel = new Label
    {
        FontSize = 18,
        TextColor = ThemeService.TextColor,
        FontFamily = ThemeService.FontFamily
    };

    _newUsernameEntry = new Entry
    {
        Placeholder = LocalizationService.Get("NewName"),
        FontFamily = ThemeService.FontFamily
    };

    _oldPasswordEntry = new Entry
    {
        Placeholder = LocalizationService.Get("OldPassword"),
        IsPassword = true,
        FontFamily = ThemeService.FontFamily
    };
```

```
_newPasswordEntry = new Entry
{
    Placeholder = LocalizationService.Get("NewPassword"),
    IsPassword = true,
    FontFamily = ThemeService.FontFamily
};

var updateButton = new Button
{
    Text = LocalizationService.Get("Update"),
    BackgroundColor = ThemeService.AccentColor,
    TextColor = ThemeService.TextColor,
    FontFamily = ThemeService.FontFamily
};
updateButton.Clicked += OnUpdateClicked;

var logoutButton = new Button
{
    Text = LocalizationService.Get("Logout"),
    BackgroundColor = Colors.Red,
    FontFamily = ThemeService.FontFamily
};
logoutButton.Clicked += OnLogoutClicked;

_messageLabel = new Label
{
    TextColor = Colors.Red,
    IsVisible = false,
    HorizontalOptions = LayoutOptions.Center,
    FontFamily = ThemeService.FontFamily
};

var scrollContent = new ScrollView
{
    Content = new VerticalStackLayout
```

```
            {
                Padding = 20,
                Spacing = 15,
                Children =
                {
                    new Label {
                        Text = LocalizationService.Get("MyProfile"),
                        FontSize = 24,
                        HorizontalOptions = LayoutOptions.Center,
                        TextColor = ThemeService.TextColor,
                        FontFamily = ThemeService.FontFamily,
                        FontAttributes = FontAttributes.Bold
                    },
                    _usernameLabel,
                    _roleLabel,
                    new BoxView { HeightRequest = 1, BackgroundColor = Colors.Gray, Margin = new Thickness(0, 10) },
                    new Label {
                        Text = LocalizationService.Get("ChangeNamePassword"),
                        FontAttributes = FontAttributes.Bold,
                        FontFamily = ThemeService.FontFamily,
                        TextColor = ThemeService.TextColor
                    },
                    _newUsernameEntry,
                    _oldPasswordEntry,
                    _newPasswordEntry,
                    updateButton,
                    new BoxView { HeightRequest = 1, BackgroundColor = Colors.Gray, Margin = new Thickness(0, 10) },
                    logoutButton,
                    _messageLabel
                }
            }
        };
```

```
            var bottomNav = SharedUI.CreateBottomNav(Navigation);

            var grid = new Grid
            {
                RowDefinitions =
                {
                    new RowDefinition { Height = GridLength.Auto },
                    new RowDefinition { Height = GridLength.Star },
                    new RowDefinition { Height = GridLength.Auto }
                }
            };

            grid.Children.Add(topBar); Grid.SetRow(topBar, 0);
            grid.Children.Add(scrollContent); Grid.SetRow(scrollContent, 1);
            grid.Children.Add(bottomNav); Grid.SetRow(bottomNav, 2);

            Content = grid;
            SharedUI.SetPageStyle(this);
        }

    }
}
```

MessagePage.cs:

```
using SelfeDefenseApp.Data;
using SelfeDefenseApp.Services;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SelfeDefenseApp.Pages
{
    public class MessagePage : ContentPage
    {
        private readonly DatabaseService _db = new();
        private readonly int _senderId = SessionService.UserId;
        private readonly string _defaultRecipient;
        private Picker _userPicker;
        private Editor _messageEditor;
        private Button _sendButton;
        private VerticalStackLayout _messageList;

        public MessagePage(string recipientUsername = null)
        {
```

```
            LocalizationService.LanguageChanged += (_, _) =>
LocalizationService.ApplyLocalization(this);
            ThemeService.ThemeChanged += (_, _) => ThemeService.ApplyTheme(this);

            ThemeService.ApplyTheme(this);

            _defaultRecipient = recipientUsername;
            Title = LocalizationService.Get("Feedback");


            _userPicker = new Picker { Title = LocalizationService.Get("UserPicker")
};

            var allUsers = _db.GetAllUsers().Where(u => u.Id != _senderId).ToList();
            foreach (var (id, name) in allUsers)
            {
                _userPicker.Items.Add($"{id}: {name}");
            }

            if (!string.IsNullOrWhiteSpace(_defaultRecipient))
            {
                var match = allUsers.FirstOrDefault(u => u.Username ==
_defaultRecipient);
                if (match.Id > 0)
                    _userPicker.SelectedItem = $"{match.Id}: {match.Username}";
            }

            _messageEditor = new Editor
            {
                Placeholder = LocalizationService.Get("EnterTheMessage"),
                AutoSize = EditorAutoSizeOption.TextChanges
            };

            _sendButton = new Button
            {
                Text = LocalizationService.Get("SendMessage"),
                BackgroundColor = Colors.Blue,

            };
            _sendButton.Clicked += SendButton_Clicked;

            _messageList = new VerticalStackLayout { Spacing = 10 };
            var scroll = new ScrollView { Content = _messageList };

            Content = new VerticalStackLayout
            {
                Padding = 20,
                Spacing = 15,
                Children = {
                _userPicker,
                _messageEditor,
                _sendButton,
                scroll
            }
            };

        }

        private void SendButton_Clicked(object sender, EventArgs e)
        {
            if (_userPicker.SelectedItem == null ||
string.IsNullOrWhiteSpace(_messageEditor.Text))
            {
                DisplayAlert("Error", "Choose user and write the SMS", "OK");
                return;
```

```
            }

            var selected = _userPicker.SelectedItem.ToString();
            int receiverId = int.Parse(selected.Split(':')[0]);

            _db.SendMessage(_senderId, receiverId, _messageEditor.Text.Trim());
            _messageEditor.Text = "";
        }


    }


}
```

NotificationsPage.cs:

```
using Microsoft.Data.Sqlite;
using SelfeDefenseApp.Data;
using SelfeDefenseApp.Services;
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Maui.Controls;
using SelfeDefenseApp.Pages.General;

namespace SelfeDefenseApp.Pages
{
    public class NotificationsPage : ContentPage
    {
        private readonly DatabaseService _db;
        private readonly int _userId = SessionService.UserId;
        private readonly bool _isAdmin = SessionService.IsAdmin;
        private readonly List<int> _selectedIds = new();
        private readonly VerticalStackLayout _notificationStack = new();
        private bool _selecting = false;

        public NotificationsPage()
        {
            _db = new DatabaseService();
            Title = LocalizationService.Get("Notifications");
            BackgroundColor = ThemeService.BackgroundColor;

            var selectBtn = new Button
            {
                Text = LocalizationService.Get("Get"),
                BackgroundColor = ThemeService.AccentColor,
                TextColor = ThemeService.TextColor,
                FontFamily = ThemeService.FontFamily
            };

            var selectAllBtn = new Button
            {
                Text = LocalizationService.Get("GetAll"),
                BackgroundColor = ThemeService.AccentColor,
                TextColor = ThemeService.TextColor,
                FontFamily = ThemeService.FontFamily
            };

            var deleteBtn = new ImageButton
            {
                Source = "trash_icon.svg",
                BackgroundColor = Colors.Transparent,
                HorizontalOptions = LayoutOptions.End,
                HeightRequest = 28,
                WidthRequest = 28
```

```
            };

            selectBtn.Clicked += (_, _) =>
            {
                _selecting = !_selecting;
                LoadNotifications();
            };

            selectAllBtn.Clicked += (_, _) =>
            {
                _selectedIds.Clear();
                _selectedIds.AddRange(_db.GetNotifications(_userId).Select(n =>
n.Id));
                LoadNotifications();
            };

            deleteBtn.Clicked += async (_, _) =>
            {
                if (_selectedIds.Count > 0)
                {
                    _db.DeleteNotifications(_selectedIds);
                    _selectedIds.Clear();
                    await DisplayAlert(LocalizationService.Get("Success"),
LocalizationService.Get("MessagesDeleted"), "OK");
                    LoadNotifications();
                }
            };

            var header = new Grid
            {
                ColumnDefinitions =
                {
                    new ColumnDefinition(GridLength.Star),
                    new ColumnDefinition(GridLength.Star),
                    new ColumnDefinition(GridLength.Auto)
                },
                Padding = new Thickness(10)
            };

            header.Children.Add(selectBtn); Grid.SetColumn(selectBtn, 0);
            header.Children.Add(selectAllBtn); Grid.SetColumn(selectAllBtn, 1);
            header.Children.Add(deleteBtn); Grid.SetColumn(deleteBtn, 2);

            var scroll = new ScrollView { Content = _notificationStack };

            Content = new VerticalStackLayout
            {
                Children = { header, scroll }
            };

            SharedUI.SetPageStyle(this);

            LoadNotifications();
        }

        private void LoadNotifications()
        {
            _notificationStack.Children.Clear();

            foreach (var (id, sender, message, isMessage, isRead) in
_db.GetNotifications(_userId))
            {
                var grid = new Grid
                {
                    ColumnDefinitions =
```

```
        {
            new ColumnDefinition { Width = GridLength.Auto },
            new ColumnDefinition(GridLength.Star),
            new ColumnDefinition { Width = GridLength.Auto }
        },
        Padding = new Thickness(10)
    };

    if (_selecting)
    {
        var checkBox = new CheckBox
        {
            IsChecked = _selectedIds.Contains(id),
            Color = ThemeService.AccentColor
        };

        checkBox.CheckedChanged += (_, e) =>
        {
            if (e.Value) _selectedIds.Add(id);
            else _selectedIds.Remove(id);
        };

        grid.Children.Add(checkBox);
        Grid.SetColumn(checkBox, 0);
    }

    var stack = new VerticalStackLayout();
    stack.Children.Add(new Label
    {
        Text = sender,
        FontAttributes = FontAttributes.Bold,
        TextColor = ThemeService.TextColor,
        FontFamily = ThemeService.FontFamily
    });
    stack.Children.Add(new Label
    {
        Text = message,
        TextColor = isRead ? ThemeService.TextColor : Colors.DarkRed,
        FontFamily = ThemeService.FontFamily
    });

    grid.Children.Add(stack);
    Grid.SetColumn(stack, 1);

    if (_isAdmin && isMessage)
    {
        var replyBtn = new ImageButton
        {
            Source = "mail_icon.svg",
            BackgroundColor = Colors.Transparent,
            HeightRequest = 24,
            WidthRequest = 24
        };

        replyBtn.Clicked += async (_, _) =>
        {
            _db.MarkNotificationAsRead(id);
            await Navigation.PushAsync(new MessagePage(sender));
        };

        grid.Children.Add(replyBtn);
        Grid.SetColumn(replyBtn, 2);
    }

    _notificationStack.Children.Add(new Frame
```

```
                {
                    CornerRadius = 20,
                    Padding = 0,
                    Margin = new Thickness(5),
                    BackgroundColor = isRead ? ThemeService.CardBackgroundColor :
ThemeService.HighlightColor,
                    Content = grid
                });

                if (!isRead)
                    _db.MarkNotificationAsRead(id);
            }
        }
    }
}
```

SettingsPage.cs:

```
using SelfeDefenseApp.Services;
using System;
using Microsoft.Maui.Storage;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SelfeDefenseApp.Pages
{
    public class SettingsPage : ContentPage
    {
        private readonly Dictionary<string, string> languageMap = new()
        {
            { "English", "en" },
            { "Українська", "uk" }
        };

        public SettingsPage()
        {
            Title = LocalizationService.Get("Settings");
            BackgroundColor = ThemeService.BackgroundColor;

            var themePicker = new Picker { Title = LocalizationService.Get("Theme")
};
            themePicker.Items.Add("Default");
            themePicker.Items.Add("Light");
            themePicker.Items.Add("Dark");

            var savedTheme = Preferences.Get("AppTheme", "Default");
            themePicker.SelectedItem = savedTheme;

            var languagePicker = new Picker { Title =
LocalizationService.Get("Language") };
            foreach (var langName in languageMap.Keys)
                languagePicker.Items.Add(langName);

            var savedLangCode = Preferences.Get("AppLanguage", "en");
            languagePicker.SelectedItem = languageMap.FirstOrDefault(x => x.Value ==
savedLangCode).Key;

            var applyButton = new Button { Text = LocalizationService.Get("Apply")
};
            applyButton.Clicked += async (s, e) =>
            {
                var selectedTheme = themePicker.SelectedItem?.ToString();
                var selectedLang = languagePicker.SelectedItem?.ToString();
```

```
                switch (selectedTheme)
                {
                    case "Dark":
                        ThemeService.ApplyDarkTheme();
                        break;
                    case "Light":
                        ThemeService.ApplyLightTheme();
                        break;
                    default:
                        ThemeService.ApplyDefaultTheme();
                        break;
                }

                var selectedLangCode = languageMap.ContainsKey(selectedLang) ?
languageMap[selectedLang] : "en";
                LocalizationService.SetLanguage(selectedLangCode);

                Preferences.Set("AppTheme", selectedTheme ?? "Default");
                Preferences.Set("AppLanguage", selectedLang ?? "en");

                await Application.Current.MainPage.Navigation.PopToRootAsync();
            };


            var backButton = new Button { Text = LocalizationService.Get("Back"),
Margin = new Thickness(0, 30, 0, 0), HorizontalOptions = LayoutOptions.Center };
            backButton.Clicked += async (s, e) => await Navigation.PopAsync();

            Content = new VerticalStackLayout
            {
                Padding = 20,
                Spacing = 15,
                Children =
                {
                    new Label { Text = LocalizationService.Get("Settings"), FontSize
= 24, HorizontalOptions = LayoutOptions.Center, TextColor = Colors.DarkBlue },
                    new Label { Text = LocalizationService.Get("Dark theme"),
TextColor = Colors.Black },
                    themePicker,
                    new Label { Text =  LocalizationService.Get("Language"),
TextColor = Colors.Black, Margin = new Thickness(0, 10, 0, 0) },
                    languagePicker,
                    applyButton,
                    backButton
                }
            };
        }
    }
}
```

StatisticPage.cs:

```
using SelfeDefenseApp.Data;
using SelfeDefenseApp.Pages.General;
using SelfeDefenseApp.Services;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SelfeDefenseApp.Pages.Admin
{
    class StatisticPage : ContentPage
    {
        private readonly DatabaseService _db = new();
```

```
        public StatisticPage()
        {
            Title = "Статистика";
            BackgroundColor = Colors.White;

            var topBar = SharedUI.CreateTopBar("Statistics",
                async () => await Navigation.PushAsync(new NotificationsPage()),
                OnMenuClicked,
                Navigation);


SharedUI.UpdateNotificationDot(_db.HasUnreadNotifications(SessionService.UserId));
            var bottomNav = SharedUI.CreateBottomNav(Navigation);

            var layout = new VerticalStackLayout
            {
                Padding = 15,
                Spacing = 10
            };

            layout.Children.Add(new Label { Text = $"All items:
{_db.GetTotalItems()}" });
            layout.Children.Add(new Label { Text = $"Visible items:
{_db.GetVisibleItems()}" });


            layout.Children.Add(new BoxView { HeightRequest = 1, Color = Colors.Gray
});


            layout.Children.Add(new Label { Text = $"Categories:
{_db.GetTotalCategories()}" });
            layout.Children.Add(new Label { Text = "Items by category:",
FontAttributes = FontAttributes.Bold });
            foreach (var (title, count) in _db.GetItemCountByCategory())
            {
                layout.Children.Add(new Label { Text = $" {title}: {count} items"
});
            }

            var scroll = new ScrollView { Content = layout };

            var grid = new Grid
            {
                RowDefinitions =
                {
                    new RowDefinition { Height = GridLength.Auto },
                    new RowDefinition { Height = GridLength.Star },
                    new RowDefinition { Height = GridLength.Auto }
                }
            };

            grid.Children.Add(topBar); Grid.SetRow(topBar, 0);
            grid.Children.Add(scroll); Grid.SetRow(scroll, 1);
            grid.Children.Add(bottomNav); Grid.SetRow(bottomNav, 2);

            Content = grid;
        }


    }
}
```

Services/

UserRepository.cs:

```
using Microsoft.Data.Sqlite;
using SelfeDefenseApp.Data;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;

namespace SelfeDefenseApp.Services
{
   public class UserRepository
   {
      private readonly string _dbPath;

      public UserRepository(DatabaseService dbService)
      {
         _dbPath = dbService.DbPath;
         EnsureUserTableStructure();
      }

      public bool ValidateUser(string username, string password)
      {
         if (string.IsNullOrWhiteSpace(username) || string.IsNullOrWhiteSpace(password))
            return false;

         using var db = new SqliteConnection($"Data Source={_dbPath}");
         db.Open();

         using var cmd = db.CreateCommand();
         cmd.CommandText = "SELECT PasswordHash FROM Users WHERE Username
= @username";
         cmd.Parameters.AddWithValue("@username", username);
```

```
            using var reader = cmd.ExecuteReader();
            if (reader.Read())
            {
                string storedHash = reader.GetString(0);
                string inputHash = HashPassword(password);

                if (storedHash == inputHash)
                    return true;

                if (storedHash == password)
                {
                    reader.Close();

                    using var updateCmd = db.CreateCommand();
                    updateCmd.CommandText = "UPDATE Users SET PasswordHash = @hash
WHERE Username = @username";
                    updateCmd.Parameters.AddWithValue("@hash", inputHash);
                    updateCmd.Parameters.AddWithValue("@username", username);
                    updateCmd.ExecuteNonQuery();

                    return true;
                }
            }

            return false;
        }

        public bool UpdateUser(string oldUsername, string? newUsername, string?
newPassword)
        {
            if (string.IsNullOrWhiteSpace(oldUsername))
                return false;

            using var db = new SqliteConnection($"Data Source={_dbPath}");
            db.Open();
```

```
            using var transaction = db.BeginTransaction();

            try
            {
                var updates = new List<string>();
                var parameters = new Dictionary<string, object>();

                if (!string.IsNullOrWhiteSpace(newUsername) && newUsername != oldUsername)
                {
                    if (DoesUserExist(newUsername))
                    {
                        return false;
                    }
                    updates.Add("Username = @newName");
                    parameters["@newName"] = newUsername;
                }

                if (!string.IsNullOrWhiteSpace(newPassword))
                {
                    updates.Add("PasswordHash = @newPass");
                    parameters["@newPass"] = HashPassword(newPassword);
                }

                if (updates.Count == 0)
                {
                    transaction.Rollback();
                    return false;
                }

                string updateClause = string.Join(", ", updates);

                using var cmd = db.CreateCommand();
                cmd.Transaction = transaction;
```

```
            cmd.CommandText = $"UPDATE Users SET {updateClause} WHERE
Username = @oldName";
            cmd.Parameters.AddWithValue("@oldName", oldUsername);

            foreach (var param in parameters)
                cmd.Parameters.AddWithValue(param.Key, param.Value);

            int rowsAffected = cmd.ExecuteNonQuery();

            if (rowsAffected > 0)
            {
                transaction.Commit();
                return true;
            }
            else
            {
                transaction.Rollback();
                return false;
            }
        }
        catch (Exception)
        {
            transaction.Rollback();
            return false;
        }
    }

    private string HashPassword(string password)
    {
        using var sha256 = SHA256.Create();
        byte[] bytes = Encoding.UTF8.GetBytes(password);
        byte[] hash = sha256.ComputeHash(bytes);
        return Convert.ToBase64String(hash);
    }
```

```
        }
    }
```

LocalizationService.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SelfeDefenseApp.Services
{
    public static class LocalizationService
    {

        public static event EventHandler LanguageChanged;

        private static string _currentLanguage = "en";
        public static string CurrentLanguage => _currentLanguage;


        private static Dictionary<string, Dictionary<string, string>> _translations
= new Dictionary<string, Dictionary<string, string>>
    {
        { "en", new Dictionary<string, string>
{
    { "MainMenuTitle", "Main Menu" },
    { "AppTitle", "Self Defense App" },
    { "Myplan", "My Plan" },
    { "Navigation", "Navigation" },
    { "Cancel", "Cancel" },
//……

}
},
{ "uk", new Dictionary<string, string>
{
    { "MainMenuTitle", "Головне меню" },
    { "AppTitle", "Додаток Самооборона" },
    { "Myplan", "Мій план" },
    { "Navigation", "Навігація" },
    { "Cancel", "Скасувати" },
    { "Feedback", "Зворотній зв'язок" },
//……


}
}

    };

        public static void SetLanguage(string lang)
        {
            if (_translations.ContainsKey(lang))
                _currentLanguage = lang;
            else
                _currentLanguage = "en";

            LanguageChanged?.Invoke(null, EventArgs.Empty);
        }

        public static string Get(string key)
        {
```

```
        if (_translations.TryGetValue(_currentLanguage, out var langDict))
        {
            if (langDict.TryGetValue(key, out var translation))
                return translation;
        }
        return key;


}
  }
```