ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ ВИВЧЕННЯ ПРАВИЛ ЕТИКЕТУ МОВОЮ C#**»

Виконав: студент 4 курсу, групи ПД–43
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності/спеціалізації)

Король Р.Є.
(прізвище та ініціали)

Керівник Аверічев І.М
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Київ –2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти -«Бакалавр»
Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____Негоденко О.В.
"____" _____2023 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТА

КОРОЛЯ РОМАНА ЄВГЕНІЙОВИЧА
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка мобільного додатку для вивчення правил етикету мовою C#».
Керівник роботи: Аверічев І.М., к.е.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «24» лютого 2023 року №26.
2. Строк подання студентом роботи «01» червня 2023 року
3. Вхідні дані до роботи
Загальновідомі правила етикету;
Науково-технічна література пов'язана з розробкою програмного забезпечення для мобільних пристроїв;

4. Зміст розрахунково-пояснювальної записки(перелік питань, які потрібно розробити).
4.1 Огляд існуючих мобільних додатків з вивчення етикету.
4.2 Засоби розробки програмного продукту..
4.3 Опис програмної реалізації.
4.4 Тестування та оцінка якості системи.
4.5 Робота Користувача З Інтерфейсом

5. Перелік демонстраційного матеріалу (назва основних слайдів)

1. Існуючі мобільні додатки з вивчення етикету
2. Блок-схема архітектури мобільного додатку
3. Інтерфейс мобільного додатку

6. Дата видачі завдання «25»лютого 2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Пошук джерел науково-технічного спрямування	02.03-07.03	Виконано
2	Аналіз задачі та її розбір	08.03-13.03	Виконано
3	Підбір системних вимог	14.03-16.03	Виконано
4	Опрацювання системної архітектури	17.03-31.03	Виконано
5	Розробка підсистемних структур	01.04-15.04	Виконано
6	Реалізація системи програмно	16.04-30.04	Виконано
7	Оформлення звіту	01.05-15.05	Виконано
8	Попередній захист роботи	15.05	Виконано
9	Здача роботи	01.06	Виконано

Студент __ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи 56 с., 16 рис., 9 джерел.

Об'єкт дослідження – вивчення правил етикету різних сфер життя.

Предмет дослідження – мобільна платформа для структурування та вивчення правил етикету.

Мета роботи – розробити мобільний додаток, який допоможе користувачам вивчати та оволодівати правилами соціального етикету в різних ситуаціях. Додаток буде надавати корисну інформацію про різноманітні мовленнєві інструменти етикетної поведінки та приклади їх вживання у реальному житті для практичного навчання.

Методи дослідження – аналіз літератури та джерел, що стосуються соціального етикету; аналіз наявних мобільних додатків з навчальною тематикою та етикетом.

Під час роботи були проаналізовані існуючі мобільні додатки з навчальною тематикою, а також додатки, що присвячені етикету та правилам соціальної поведінки.

Загальною проблемою цих продуктів є брак наявності цільового мобільного додатку, спрямованого на вивчення та покращення навичок етикету в різних сферах життя.

Особливістю додатку є цілеспрямований навчальний процес з використанням інтерактивних елементів для кращого засвоєння правил етикету. За основу було взято технологію Xamarin на базі фрейморка .NET з серверної сторони та XAML-розмітку з клієнтської сторони для основного функціоналу.

В якості серверу бази даних було взято SQLite та бібліотеку sqlite-net-pcl для взаємодії з нею.

Отже, розроблено мобільний додаток для платформи Android, який містить навчальний контент, опис, приклади та пояснення правил етикету для вжитку в різних сферах життя.

У якості вихідних даних є ієрархічна структура блоків та навчальних карток, розділених по категоріям.

Даний додаток може бути використано у різних сферах життя для взаємодії з людьми, від буденного побутового етикету до ділових та міжнародних відносин.

Галузь використання – освіта, особистісний розвиток.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

XAML – розширювана мова розмітки програм

ASP.NET - активні серверні сторінки для .NET

SDK – комплект розробки програмного забезпечення

API - інтерфейс прикладного програмування

ПЗ – програмне забезпечення

СУБД – Система Управління Базами Даних

ВСТУП

Обґрунтування вибору теми та її актуальність: в сучасному суспільстві правила соціального етикету набувають все більшого значення. Правильна поведінка та вміння взаємодіяти з іншими людьми стають важливими як в особистому, так і в професійному житті. Розуміння правил етикету може сприяти покращенню комунікації, будуванню стосунків та досягненню успіху в різних сферах життя.

Мобільні пристрої, такі як смартфони та планшети, стали невід'ємною частиною повсякденного життя. Вони дозволяють людям мати постійний доступ до інформації та навчальних ресурсів.

Щоби дозволити користувачам зручно та ефективно здобувати знання про етикет в будь-якому місці і в будь-який час необхідно розробити мобільний додаток для вивчення правил етикету.

Ступінь вивчення проблеми: На сьогоднішній день на ринку немає багато мобільних додатків, спеціалізованих на вивчення правил етикету. Брак такого додатку ускладнює доступність інформації та навчання етикету для широкого загалу користувачів. Розробка мобільного додатку з цією специфічною тематикою заповнить цю прогалину і надасть зручний і доступний інструмент для вивчення правил етикету.

Багато з наявних на ринку додатків мають обмежені можливості і не забезпечують достатньо інтерактивного та персоналізованого досвіду навчання. Основна проблема полягає у неможливості самостійно створювати власні навчальні блоки для використання додатку вчителям.

Об’єктом дослідження є вивчення правил етикету різних сфер життя.

Предметом роботи є мобільна платформа для структурування та вивчення правил етикету.

Метою роботи є покращення якості вивчення правил етикету за допомогою додатку.

Завданням роботи є розробка мобільного додатку для вивчення правил етикету.

Методика дослідження: Перш за все, потрібно було вивчити літературу та наукові джерела, що стосуються соціального етикету і навчання, а також проаналізувати існуючі мобільні додатки, які пов'язані з навчанням етикету або схожими темами, визначити їх переваги, недоліки та можливість вдосконалення.

З урахуванням вимог, специфіки програмного продукту та забезпечення оптимальної функціональності, було визначено, що найкращим рішенням є реалізація мобільного додатку з клієнт-серверною архітектурою. У такій архітектурі клієнтська сторона відповідає за логіку та роботу програмного продукту, тоді як серверна сторона виконує обчислення та надає додатку необхідні ресурси і дані.

Для клієнтської сторони, інформаційна система має включати в себе ієрархічну бібліотеку правил етику для навчання. Кожен модуль ієрархії матиме опції додавання, редагування та видалення, для більшої персоналізації програмного продукту.

На основі отриманих даних та аналізу необхідно розробити прототип мобільного додатку та забезпечити інтерактивні функції, можливості персоналізації та практичного навчання етикету.

Таким чином, наукова новизна полягає в інтерактивних функціях, що дозволять користувачам відразу застосовувати вивчені правила етикету. Крім того, додаток забезпечить персоналізацію, дозволяючи налаштувати навчання під свої потреби та рівень знань.

Практична значущість результатів: даний додаток може бути використано у різних сферах життя для ефективної взаємодії з людьми, від буденного побутового етикету до ділових та міжнародних відносин.

1 ОГЛЯД ІСНУЮЧИХ МОБІЛЬНИХ ДОДАТКІВ З ВИВЧЕННЯ ЕТИКЕТУ

1.1. Додаток Etiquette

Додаток "Etiquette" є мобільним додатком, розробленим для вивчення основних правил етикету з використанням барвистих зображень та відповідних картинок. Додаток наведений на рис.1.1 та має наступний докладний опис:

Головний екран: Після запуску додатка користувач зустрічається з головним екраном, на якому відображено списком основні правила етикету. Кожне правило супроводжується зображенням або картинкою, яка ілюструє конкретну ситуацію.

1. Категорії етикету: Правила етикету розділені на категорії для зручності користувача. Наприклад, можуть бути категорії, такі як "Зустрічі та прийоми", "Ділова етикет", "Столова етикет" тощо. Користувач може вибрати потрібну категорію та переглянути відповідні правила.
2. Детальна інформація про правила: При виборі конкретного правила користувач може переглянути його детальний опис. Опис містить пояснення правила, вжиток в життєвих ситуаціях.

Рисунок 1.1 – Додаток «Etiquette»

Додаток "Etiquette" є зручним інструментом для вивчення основного етикету з використанням барвистих зображень та відповідних картинок. Основні характеристики додатку включають:

Переваги:

1. Візуальна привабливість: Додаток використовує барвисті зображення та картинки, що робить процес вивчення правил етикету цікавим та привабливим для користувачів. Візуальні елементи допомагають залучити увагу та запам'ятати правила легше.
2. Широкий охоплення етикету: "Etiquette" включає понад 50 елементів етикету, що дозволяє користувачам ознайомитись з широким спектром соціальних норм та правил. Це забезпечує повноту та різноманітність змісту додатку.
3. Зручний спосіб перегляду: Правила етикету відображаються у вигляді карток з зображеннями, що дозволяє користувачам прокручувати список знизу вгору та переглядати весь етикет. Це забезпечує простий та зрозумілий спосіб ознайомлення з правилами.

Недолік:

1. Відсутність можливості доповнення та редагування: Додаток "Etiquette" не надає можливість користувачам додавати або редагувати правила етикету. Весь контент є фіксованим і не може бути змінений або доповнений користувачами.

1.2. Додаток "Діловий етикет"

Користувач взаємодіє з додатком "Діловий етикет" за допомогою простого та зрозумілого інтерфейсу.

На рис 1.2 наведений додаток «Діловий етикет».

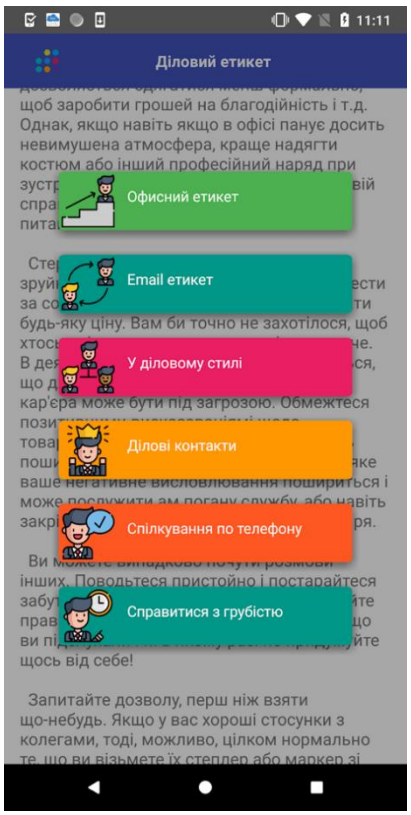


Рисунок 1.2 – Додаток «Діловий етикет»

Основні етапи взаємодії включають наступне:

1. Запуск додатку: Користувач запускає додаток на своєму мобільному пристрої. Після запуску відображається головний екран з доступними категоріями або списком правил ділового етикету.
2. Вибір категорії або правила: Користувач може вибрати конкретну категорію правил, якщо вони доступні, або переглянути повний список правил. Вибір здійснюється шляхом торкання елемента на екрані.
3. Читання порад та рекомендацій: Після вибору категорії або правила користувач може читати поради та рекомендації, які пов'язані з конкретним правилом ділового етикету. Текстовий матеріал може бути представлений у вигляді коротких описів, списків або абзаців.
4. Приклади та пояснення: Додаток може надавати приклади з реального життя, які допомагають краще зрозуміти застосування правил ділового етикету.

Приклади можуть включати сценарії взаємодії з колегами, клієнтами або під час корпоративних заходів.

Основні особливості додатка "Діловий етикет":

1. Коротка книга порад: Додаток містить компактну, але інформативну книгу з практичними порадами та рекомендаціями щодо ділового етикету. Вона надає користувачам доступ до необхідної інформації без зайвого перевантаження.
2. Чіткі і лаконічні пояснення: Кожне правило ділового етикету пояснюється чітко і лаконічно, зрозуміло для будь-якого користувача. Пояснення базуються на конкретних прикладах з реального життя, що допомагає легше усвідомити, як правильно застосовувати ці правила.
3. Прості виконання вказівки: Додаток надає прості та легкі виконання вказівки щодо того, як дотримуватися правил ділового етикету в різних ситуаціях. Це допомагає користувачам швидко засвоїти правила та безпомилково їх застосовувати.
4. Мобільний доступ: Додаток доступний на мобільних пристроях, що дозволяє користувачам мати постійний доступ до порад та рекомендацій ділового етикету незалежно від місця та часу.

Головні недоліки додатку:

1. Вузька спеціалізація тільки для ділової етики: Додаток фокусується виключно на діловому етикеті і не надає інформацію про інші аспекти етикету, такі як соціальний етикет, етикет в особистому житті тощо. Це може бути обмеженням для користувачів, які також зацікавлені в інших сферах етикету.
2. Відсутність можливості для редагування: Додаток не надає можливості користувачам додавати або редагувати правила ділового етикету. Це означає, що користувачі не можуть доповнювати додаток новими правилами або адаптувати його до своїх конкретних потреб чи контексту. Додаток

обмежений вже наявною інформацією, і користувачі не можуть внести зміни до його вмісту.

1.3 Висновки

Загальні висновки щодо неможливості наявних на ринку додатків мати можливість редагування вмісту для вчителів та необхідності створення нового такі:

1. Обмеженість готових додатків: Багато наявних на ринку додатків для навчання мають фіксований вміст, який не може бути редагованим вчителем. Це створює обмеження для вчителів, оскільки вони не можуть відповідати на конкретні потреби своїх учнів або внести зміни до вмісту, щоб пристосувати його до своєї методики навчання.
2. Персоналізація інструкцій: Кожен вчитель має власний стиль навчання та підхід до викладання. Нередагований вміст додатків може не враховувати унікальні потреби вчителів та їхню власну методику навчання. Можливість редагування дозволить вчителям персоналізувати інструкції та вміст, щоб краще відповідати потребам їхніх учнів.
3. Актуалізація та модифікація вмісту: Одна з важливих складових успішного навчання - це актуальний та сучасний вміст. Відсутність можливості редагування вмісту у наявних додатках може призвести до застарілості матеріалів та невідповідності їхнього змісту сучасним стандартам та потребам учнів. Здатність вчителів оновлювати та модифікувати вміст створить можливість підтримувати актуальність та релевантність навчальних матеріалів.
4. Індивідуальне навчання: Кожен учень має свої власні потреби та темп навчання. Можливість редагування вмісту додатків надає вчителям можливість адаптувати матеріали до індивідуальних потреб кожного учня. Вони можуть включати додаткові приклади, завдання або пояснення, що сприятимуть кращому засвоєнню матеріалу і підвищенню успішності кожного учня.

Враховуючи перераховані недоліки наявних на ринку додатків, стає очевидним, що необхідно створення нового додатку, який надаватиме вчителям можливість редагування вмісту. Це дозволить вчителям адаптувати навчальні матеріали до своїх конкретних потреб, стилю викладання та унікальних вимог їхніх учнів. Такий додаток буде забезпечувати більш гнучку, персоналізовану та актуальну навчальну ситуацію, що сприятиме покращенню навчального процесу та досягненню кращих результатів навчання учнів.

2 ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

2.1. Середовище розробки Visual Studio

Microsoft Visual Studio — серія продуктів фірми Майкрософт, які складають інтегроване середовище розробки програмного забезпечення та низку інших інструментальних засобів. Ці продукти дають можливість розробити як консольні програми, так і програми з графічним інтерфейсом, включно з підтримкою технології Windows Forms, а також веб-сайти, веб-застосунки, веб-служби як у рідному, так і в керованому коді для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile , Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight [4].

Середовище розробки Visual Studio є одним з найпопулярніших та потужних інструментів для розробки мобільних додатків. Ось деякі з його переваг і причин, чому воно підходить для розробки мобільних додатків:

1. Підтримка різних мов програмування: Visual Studio надає підтримку для різних мов програмування, включаючи C#, яка є однією з основних мов для розробки мобільних додатків на платформі Android та iOS.
2. Інтегроване середовище розробки: Visual Studio надає повноцінне інтегроване середовище розробки з вбудованими інструментами для редагування коду, налагодження, керування версіями та іншими корисними функціями. Це робить процес розробки зручним та ефективним.
3. Наявність шаблонів та компонентів: Visual Studio має багатий набір шаблонів та компонентів, які допомагають прискорити процес розробки. Наприклад, він надає шаблони проектів для розробки мобільних додатків на платформах Android та iOS, а також бібліотеки компонентів, які можна використовувати для швидкого створення інтерфейсу користувача та функціональності.
4. Налагоджування та тестування: Visual Studio надає потужні засоби для налагодження та тестування мобільних додатків. Він підтримує розробку на

емуляторах або фізичних пристроях, а також надає можливість перевіряти та тестувати додатки на різних пристроях та конфігураціях.

5. Екосистема та підтримка: Visual Studio базується на широкій екосистемі, яка включає різноманітні розширення, плагіни та інструменти спільноти. Це означає, що розробники мають доступ до безлічі ресурсів, які можуть поліпшити їхню продуктивність та розширити можливості розробки мобільних додатків.
6. Хмарні сервіси та інтеграція: Visual Studio має вбудовану підтримку для хмарних сервісів, таких як Microsoft Azure. Це дозволяє легко інтегрувати мобільні додатки з хмарними сервісами, використовувати аналітику, зберігання даних та інші функції, що сприяють розширенню можливостей додатків.
7. Підтримка мультиплатформеності: Visual Studio дозволяє розробляти мобільні додатки для різних платформ, таких як Android, iOS та Windows. Це дає розробникам можливість створювати додатки, які працюють на різних пристроях та операційних системах, з використанням спільного коду та переносимих компонентів. У загальному, Visual Studio є потужним інструментом для розробки мобільних додатків, який надає розробникам необхідні засоби, шаблони, підтримку та ресурси для ефективної розробки та впровадження додатків на різних мобільних платформах. Завдяки його можливостям, розробники можуть створювати високоякісні та потужні мобільні додатки зручно та ефективно.

2.1.1. Фреймворк .NET

.NET Framework (читається дот-нет) — програмна технологія, запропонована фірмою Microsoft як платформа для створення як звичайних програм, так і веб-застосунків. Багато в чому є продовженням ідей та принципів, закладених у технологію Java. Однією з ідей .NET є сумісність служб, написаних великими мовами. Якщо ця можливість рекламується Microsoft як перевага .NET, платформа Java має таку ж можливість [5]. Фреймворк .NET є одним з найпопулярніших

інструментів для розробки програмного забезпечення, включаючи мобільні додатки. Він розроблений компанією Microsoft і надає розробникам широкий набір інструментів і бібліотек для побудови високоякісних, безпечних та ефективних додатків. Особливості фреймворку .NET для розробки мобільних додатків:

1. Мови програмування: Фреймворк .NET підтримує декілька мов програмування, таких як C#, VB.NET і F#, які можуть бути використані для розробки мобільних додатків. Це дає розробникам можливість вибрати мову, з якою вони найкраще володіють або яка найбільше підходить для їхнього проекту.
2. Кросплатформеність: Фреймворк .NET має кросплатформену підтримку, що дозволяє розробляти мобільні додатки для різних операційних систем, включаючи Android та iOS. Це досягається за допомогою Xamarin, що дозволяє писати спільний код, який може бути використаний на обох платформах, забезпечуючи ефективне використання ресурсів і збільшуючи швидкість розробки.
3. Багатоплатформеність: Фреймворк .NET дозволяє розробникам створювати додатки для різних платформ, включаючи мобільні пристрої, настільні комп'ютери та хмарні сервіси. Це означає, що розробники можуть використовувати ті ж інструменти та мови програмування для створення додатків, які працюють на різних платформах, спрощуючи процес розробки та підтримку додатків.
4. Широкий вибір бібліотек і фреймворків: Фреймворк .NET має велику екосистему бібліотек і фреймворків, які допомагають розробникам прискорити процес розробки та забезпечити більшу функціональність своїх мобільних додатків. Наприклад, Xamarin.Forms надає набір готових елементів керування і інструментів для побудови користувацького інтерфейсу мобільного додатку. Також існує безліч сторонніх бібліотек, які додають додаткові функціональні можливості, такі як робота з базами даних, мережевими запитами, анімаціями та іншими аспектами розробки мобільних додатків.

5. Інструменти для тестування: Фреймворк .NET надає розробникам різні інструменти для тестування мобільних додатків, включаючи модульне тестування, функціональне тестування та UI-тестування. Це дозволяє розробникам перевіряти функціональність та якість своїх додатків перед релізом, забезпечуючи стабільну та надійну роботу програмного забезпечення.
6. Підтримка спільної розробки: Фреймворк .NET надає засоби для спільної розробки мобільних додатків у команді. Розробники можуть використовувати інструменти контролю версій, такі як Git, для спільної роботи над кодом, управління залежностями та розв'язання конфліктів. Це дозволяє командам ефективно працювати над проектом, спільно ділитися змінами та контролювати версію додатку.

Загалом, фреймворк .NET забезпечує розробникам потужні інструменти та ресурси для розробки мобільних додатків. Його особливості, такі як кросплатформеність, багатоплатформеність, широкий вибір бібліотек і фреймворків, інструменти для тестування та спільна розробка, роблять його привабливим вибором для розробників, які прагнуть створювати мобільні додатки з високою продуктивністю, які працюють на різних платформах.

Особливо для розробки мобільних додатків, фреймворк .NET з використанням Xamarin дозволяє писати спільний код, що ефективно використовується на платформах Android та iOS. Це робить процес розробки швидшим і простішим, оскільки розробники можуть перевикористовувати значну частину коду та розробляти функціональність для обох платформ одночасно. Крім того, фреймворк .NET надає доступ до багатьох перевірених рішень та інструментів для розробки мобільних додатків, що полегшує процес розробки та забезпечує високу якість продукту.

Таким чином, фреймворк .NET разом із середовищем розробки Visual Studio створюють потужну комбінацію для розробки мобільних додатків. Вони надають розробникам необхідні інструменти, бібліотеки та функціональні можливості для

створення високоякісних та кросплатформених додатків, сприяючи продуктивності та швидкому впровадженню на ринок.

2.1.2. Плагін Xamarin для Visual Studio

Xamarin — це компанія з розробки програмного забезпечення, що належить Microsoft і розташована в Сан-Франциско, заснована в травні 2011 року інженерами, які створили Mono Xamarin.Android (раніше Mono для Android) і Xamarin.iOS (раніше MonoTouch), які є перехресними -платформні реалізації інфраструктури спільної мови (CLI) і специфікації загальної мови (часто називаються Microsoft .NET) [6].

Плагін Xamarin для Visual Studio - це надбудова до середовища розробки Visual Studio, яка надає розширені можливості для розробки мобільних додатків з використанням фреймворку Xamarin. Цей плагін дозволяє розробникам ефективно створювати кросплатформені додатки для платформ Android, iOS і Windows, використовуючи мову програмування C# та спільний код.

Основні переваги плагіна Xamarin для Visual Studio включають:

1. Інтегроване середовище: Плагін Xamarin інтегрується безпосередньо з Visual Studio, що забезпечує зручний та зрозумілий розробний досвід. Розробники можуть використовувати всі функції та інструменти Visual Studio, такі як налагоджувач, інструменти аналізу коду та побудови інтерфейсу користувача.
2. Кросплатформений розробка: За допомогою плагіна Xamarin, розробники можуть писати спільний код на мові C#, який може бути використаний для розробки додатків на платформах Android, iOS і Windows. Це зменшує зусилля, потрібні для розробки додатків для кожної окремої платформи і дозволяє ефективно використовувати спільну логіку та компоненти.
3. Візуальний дизайнер інтерфейсу: Плагін Xamarin надає візуальний дизайнер інтерфейсу, що дозволяє розробникам створювати і налаштовувати користувацький інтерфейс додатку безпосередньо в середовищі Visual Studio. Це спрощує розробку і покращує процес дизайну додатку.

4. Розширені можливості тестування: Плагін Xamarin надає інструменти для автоматизованого тестування мобільних додатків, що дозволяє виявляти помилки та перевір
5. Облік різних платформ: Плагін Xamarin для Visual Studio забезпечує можливість налаштування і управління проектами для різних мобільних платформ. Розробники можуть легко додавати нові цільові платформи, налаштовувати їх параметри та використовувати специфічні функції для кожної платформи. Це дозволяє розробникам адаптувати свої додатки до особливостей кожної платформи і забезпечує більшу гнучкість у виборі цільової аудиторії.
6. Підтримка сторонніх бібліотек і компонентів: Плагін Xamarin дозволяє розробникам використовувати сторонні бібліотеки і компоненти, що значно розширює функціональні можливості додатків. Велика кількість сторонніх бібліотек доступна для використання в Xamarin, що спрощує розробку додатків з багатим функціоналом.
7. Підтримка хмарних сервісів: Плагін Xamarin дозволяє легко інтегрувати хмарні сервіси, такі як сервіси аутентифікації, бази даних, зберігання файлів та багато іншого. Це дозволяє розробникам створювати додатки, які використовують потужні функції хмарних сервісів, забезпечуючи розширені можливості та зручне управління даними.
8. Підтримка розподілу додатків: Плагін Xamarin для Visual Studio надає розробникам засоби для розповсюдження додатків. Ви можете легко збирати, підписувати та розповсюджувати додатки для різних платформ, використовуючи різні магазини додатків і інструменти розподілу.

В цілому, плагін Xamarin для Visual Studio є потужним інструментом для розробки мобільних додатків на базі фреймворку .NET. Завдяки плагіну Xamarin, це середовище розробки стає ще більш ефективним і зручним для створення кросплатформових додатків.

2.1.3. Бібліотека для інструментів Android SDK

Бібліотека для інструментів Android SDK (Software Development Kit) - це набір програмних компонентів, інструментів і ресурсів, які надаються розробникам для створення додатків під операційну систему Android. Ця бібліотека включає в себе різноманітні API, документацію, зразки коду, емулятори, плагіни та інші корисні ресурси, необхідні для розробки на платформі Android.

Основні особливості і переваги Бібліотеки для інструментів Android SDK:

1. API-інтерфейси: Android SDK надає широкий набір API-інтерфейсів, які дозволяють розробникам взаємодіяти з різними функціями і сервісами операційної системи Android. Ці API включають доступ до функцій графіки, мережі, бази даних, мультимедіа, сенсорів, а також інші можливості пристрою.
2. Інструменти розробки: Android SDK містить набір інструментів, що допомагають розробникам створювати, тестувати і налагоджувати додатки. Серед цих інструментів можна виділити Android Studio - інтегроване середовище розробки (IDE) для Android, яке надає потужні засоби для написання коду, керування проектами, налагодження та профілювання додатків.
3. Емулятори: Android SDK постачається з емуляторами Android, які дозволяють розробникам виконувати та тестувати додатки на віртуальних пристроях Android. Це дає змогу перевірити поведінку додатків на різних версіях Android та на різних розмірах екрану без реальних пристроїв.
4. Документація та зразки коду: Android SDK постачається з докладною документацією, яка описує функції, класи і методи API.
5. Плагіни та розширення: Android SDK також підтримує різноманітні плагіни та розширення, які дозволяють розширити функціональність і можливості розробки. Наприклад, плагін Android Gradle дозволяє розробникам налаштовувати збірку та залежності проекту, плагін Android Support Library надає додаткові компоненти та функціональність для сумісності з різними версіями Android.

6. Підтримка різних версій Android: Android SDK забезпечує підтримку різних версій операційної системи Android, що дозволяє розробникам створювати додатки, які можуть працювати на широкому спектрі пристроїв. Розробники можуть визначати мінімальну і рекомендовану версію Android для своїх додатків, що допомагає забезпечити сумісність і широку доступність.

Загалом, Бібліотека для інструментів Android SDK забезпечує розробникам необхідні інструменти, ресурси та підтримку для створення високоякісних додатків для мобільної платформи Android. Ця бібліотека дозволяє розробникам створювати додатки з різноманітним функціоналом, використовуючи різні можливості пристроїв Android, такі як камера, GPS, сенсори, сповіщення та багато іншого.

Завдяки Android SDK, розробники можуть створювати інноваційні додатки, які взаємодіють зі специфічними функціями пристрою, використовують мультимедійні можливості, забезпечують доступ до Інтернету, використовують локаційні дані та багато іншого. Крім того, Android SDK надає можливості для розробки ігор, розширених реальності, мобільних сервісів, соціальних додатків та інших типів додатків. Бібліотека для інструментів Android SDK є важливим компонентом розробки на платформі Android, що допомагає розробникам зосередитись на створенні потужних, ефективних та високоякісних додатків. Завдяки розширеним можливостям, документації та активній спільноті розробників, Android SDK є незамінним інструментом для розробки мобільних додатків під платформу Android.

2.2 Декларативна мова розмітки XAML

Розширена мова розмітки додатків (Extensible Applicion Markup Language, XAML) - це ядро додатка Silverlight. Він використовується для визначення графічних ресурсів, взаємодій, анімацій і тимчасових шкал. XAML ґрунтується на розширенні мови розмітки (Extensible Markup Language, XML), тому все описується в текстовому форматі з використанням атрибутів для оголошення властивостей, методів і подій.

Файли XAML - це файли XML, які володіють звичайним розширенням імені файлу XAML. Наступний приклад демонструє вміст файлу XAML дуже спрощеного Silverlight [4]. Декларативна мова розмітки XAML (eXtensible Application Markup Language) є потужним інструментом для опису і візуалізації інтерфейсу користувача в додатках під управлінням платформ Windows, включаючи мобільну платформу Xamarin. XAML використовується для розділення логіки додатку від його представлення, дозволяючи розробникам створювати гнучкі та масштабовані інтерфейси, а також відділяти дизайнерську роботу від розробки програмного коду. Замість того, щоб програмувати власне створення та розміщення кожного елемента вручну, розробники можуть використовувати XAML для опису структури і вигляду елементів інтерфейсу, а фреймворк буде відповідальний за рендеринг цих елементів. Одна з найбільших переваг використання XAML полягає у його читабельності та розумінні для розробників та дизайнерів. XAML використовує декларативний підхід, де елементи розмітки описуються як деревовидна структура, що спрощує візуальне розуміння інтерфейсу. Крім того, XAML дозволяє використовувати прив'язки даних та стилі, що дозволяє швидко змінювати вигляд і поведінку елементів без необхідності зміни коду. Іншою важливою особливістю XAML є можливість використання розділеної розробки, де дизайнери можуть працювати з розміткою XAML, встановлюючи стиль і макет, а розробники можуть використовувати цю розмітку для програмування логіки та взаємодії з елементами. Завдяки XAML, розробники мають можливість швидко створювати складні інтерфейси та забезпечувати їх адаптивність до різних розмірів екранів і пристроїв. XAML надає розширені можливості розміщення елементів, використання контейнерів, розташування елементів на сітці, а також підтримку анімації і переходів між сторінками. Окрім того, XAML інтегрується з різними інструментами і технологіями розробки. Використовуючи XAML, розробники можуть легко поєднувати його з мовою програмування C# або іншими мовами платформи .NET, що дозволяє розробляти потужні функціональність та бізнес-логіку додатка. Крім того, XAML підтримує

використання стилів, шаблонів і ресурсів, що дозволяє зручно управляти виглядом і поведінкою елементів інтерфейсу. XAML також має вбудовану підтримку для прив'язки даних, що дозволяє зручно зв'язувати дані додатку з його користувацьким інтерфейсом. Це дозволяє розробникам ефективно оновлювати вміст інтерфейсу, коли дані змінюються, і спрощує роботу зі списками, таблицями і формами. Однією з особливостей XAML є його можливість використовувати повторне використання коду та компонентів. За допомогою вбудованих механізмів стилів, шаблонів і наслідування, розробники можуть створювати перевикористовувані компоненти і зручно управляти їхнім виглядом та поведінкою. Це дозволяє підвищити продуктивність розробки, зменшити дублювання коду і спростити підтримку додатку.

Особливості XAML зробили його популярним інструментом для розробки мобільних додатків, оскільки він спрощує процес створення привабливого та функціонального інтерфейсу. Розробники можуть швидко візуалізувати свої ідеї, ефективно використовувати готові компоненти та стилі, а також забезпечувати швидку реакцію на зміни вимог і дизайну. Завдяки своїм перевагам та гнучкості, XAML став невід'ємною частиною розробки мобільних додатків на платформі Xamarin, дозволяючи розробникам створювати професійні та зручні інтерфейси для різних пристроїв та операційних систем.

2.2. Об’єктно-орієнтована мова програмування C#

C# (вимовляється Сі-шарп) — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вільтамутом та Пітером Гольде під егідою Microsoft Research (належить Microsoft) [8].

C# (C-Sharp) - це об'єктно-орієнтована мова програмування, розроблена компанією Microsoft. Вона є однією з основних мов програмування для розробки додатків для платформ .NET Framework та .NET Core. C# поєднує в собі сильні сторони C++ та простоту використання мови програмування Java.

Основні особливості мови C# включають:

1. Структурованість та об'єктно-орієнтований підхід: C# дозволяє розробникам створювати структури даних та класи, які містять як дані, так і пов'язані з ними функції (методи). Це сприяє організації коду у логічні блоки та полегшує підтримку та розширення додатків.
2. Сильна типізація: C# вимагає чіткого вказання типів даних для змінних та параметрів. Це допомагає виявити помилки під час компіляції та полегшує розуміння програмного коду.
3. Підтримка збору сміття: C# має вбудований механізм збору сміття, який автоматично вивільняє пам'ять, використану об'єктами, які вже не потрібні в програмі. Це полегшує роботу з пам'яттю та запобігає виникненню помилок, пов'язаних з некоректним управлінням пам'яттю.
4. Винятковий обробник: C# надає механізми для обробки виняткових ситуацій, що дозволяють програмі виявляти та відповідно реагувати на помилки під час виконання. Це сприяє забезпеченню стабільності та надійності програм.
5. Підтримка багатопоточності: C# дозволяє розробникам створювати багатопоточні програми, що виконуються паралельно та одночасно обробляють різні завдання. Це особливо важливо для розробки додатків, які мають велику кількість операцій, які можуть виконуватись асинхронно. Багатопоточність дозволяє поліпшити продуктивність, швидкість та відзивчивість додатку, а також забезпечує ефективне використання ресурсів обчислювальної системи.
6. Підтримка LINQ (Language-Integrated Query): C# включає мову запитів, що інтегрована в мову програмування. LINQ дозволяє зручно та ефективно працювати з колекціями об'єктів, виконуючи різні операції фільтрації, сортування, групування та обчислення. Це спрощує роботу з даними та полегшує розробку складних запитів.
7. Широкі можливості розробки: C# підтримує розробку різноманітних типів додатків, включаючи мобільні додатки для платформ Android та iOS, десктопні додатки для Windows, веб-додатки та служби. Завдяки цьому,

розробники можуть використовувати C# для створення різноманітних додатків та розширення їх функціональності.

C# є однією з основних мов програмування для розробки мобільних додатків на платформі Xamarin, що базується на фреймворку .NET. Дана комбінація надає розробникам потужні та гнучкі інструменти для створення переносних додатків, які можуть працювати на різних платформах, включаючи Android та iOS.

Особливості C# для розробки мобільних додатків:

1. Кросс-платформеність: Завдяки використанню платформи Xamarin, розробники можуть написати код на C#, який може бути використаний як для Android, так і для iOS. Це дозволяє економити час і зусилля, оскільки не потрібно розробляти окремий код для кожної платформи.
2. Висока продуктивність: Мова C# є компільованою мовою, що дозволяє отримувати високу продуктивність виконання додатків. Вона має потужні інструменти оптимізації та підтримує використання низькорівневих операцій, що сприяє ефективній роботі додатків на мобільних пристроях.
3. Розширена база бібліотек та фреймворків: C# має широкий вибір бібліотек та фреймворків, що дозволяють розробникам використовувати готові рішення для різних завдань. Наприклад, Xamarin.Forms надає набір готових елементів керування (controls) та можливостей для швидкої розробки інтерфейсу користувача.
4. Єдина мова для всього проекту: Завдяки використанню C# як основної мови програмування для розробки мобільних додатків, розробники можуть використовувати одну мову для всього проекту. Це полегшує розробку, тестування та випуск коду.

Загалом, мова програмування C# є потужним інструментом для розробки мобільних додатків. Вона поєднує в собі кросс-платформеність, продуктивність та широкі можливості розробки, що дозволяє розробникам створювати якісні та ефективні додатки для різних платформ мобільних пристроїв. Використання C# разом з фреймворком Xamarin дозволяє розробникам зосередитися на розробці функціональності додатку, забезпечуючи його швидкість, стабільність та високу

якість. Багатопоточність, підтримка збору сміття та механізми обробки виняткових ситуацій роблять розробку мобільних додатків на C# ефективною та безпечною. Підтримка LINQ дозволяє зручно та ефективно працювати з даними, спрощуючи розробку складних запитів та маніпуляцій з колекціями. За допомогою C# розробники можуть створювати інноваційні та функціональні мобільні додатки, які задовольняють вимоги користувачів і працюють на різних платформах, розширюючи свій охоплення та досягаючи більшої аудиторії.

2.3. Компоненти .xaml.cs

Компоненти .xaml.cs є важливою частиною розробки мобільних додатків з використанням мови програмування C# та платформи Xamarin. Коли розробляється інтерфейс користувача для мобільного додатку за допомогою XAML (eXtensible Application Markup Language), кожен елемент інтерфейсу може мати відповідний файл .xaml.cs, який містить логіку і взаємодію з цим елементом.

У файлі .xaml.cs визначається клас, який відповідає елементу інтерфейсу, і він наслідується від певного класу базового коду. Цей базовий клас надає розширені функціональні можливості для роботи з елементами інтерфейсу.

У компонентах .xaml.cs можна визначати обробники подій, які реагують на дії користувача або зміни стану елементів інтерфейсу. Наприклад, можна визначити обробник натискання кнопки або зміни значення введеного тексту. Ці обробники подій дозволяють відповідати на взаємодію користувача з додатком та виконувати певні дії відповідно до цих подій.

Крім обробки подій, компоненти .xaml.cs також використовуються для встановлення значень властивостей елементів інтерфейсу, маніпуляції з даними, взаємодії з іншими компонентами додатку та виконання інших функцій, необхідних для роботи додатку.

Одна з основних переваг використання компонентів .xaml.cs полягає в розділенні логіки додатку та його візуального представлення. XAML відповідає за опис інтерфейсу користувача, тоді як C# відповідає за логіку та функціональність

додатку. Це дозволяє зробити код більш структурованим, читабельним і підтримуваним. Крім того, використання компонентів .xaml.cs спрощує роботу з елементами інтерфейсу, оскільки надає зручний доступ до їх властивостей та методів.

Компоненти .xaml.cs також забезпечують можливість роботи з даними. Можна зв'язувати дані з різних джерел, таких як база даних, веб-сервіси або локальні ресурси, і використовувати ці дані для заповнення елементів інтерфейсу або для виконання розрахунків та обробки даних. Ще одна перевага компонентів .xaml.cs полягає у використанні асинхронного програмування. Завдяки цьому, можна виконувати довготривалі операції, такі як завантаження даних з Інтернету, без блокування головного потоку інтерфейсу користувача. Це забезпечує звичайну реактивність додатку та забезпечує плавну взаємодію з користувачем.

Однак, використання компонентів .xaml.cs має певні недоліки. Один з них полягає в можливому збільшенні складності коду при роботі зі складними інтерфейсами. Це може виникнути, якщо логіка додатку залежить від багатьох елементів інтерфейсу та обробників подій. Для запобігання цьому, важливо дотримуватись принципу розділення відповідальностей і розбити код на декілька компонентів, які взаємодіють між собою.

Крім того, редагування інтерфейсу, описаного в XAML, може бути обмеженим при використанні компонентів .xaml.cs. Зміни в дизайні можуть вимагати редагування коду, що може бути не зручним для деяких розробників.

Важливо зазначити, що використання компонентів .xaml.cs має багато переваг для розробки мобільних додатків з використанням мови програмування C# та платформи Xamarin. Вони дозволяють розділити логіку додатку та його візуальне представлення, спрощують роботу з елементами інтерфейсу, надають доступ до властивостей та методів елементів інтерфейсу, а також забезпечують можливість роботи з даними та виконання асинхронних операцій.

Проте, важливо бути усвідомленим щодо можливих недоліків, таких як збільшення складності коду при роботі зі складними інтерфейсами та обмежена

можливість редагування дизайну без змін в коді. Ці недоліки можуть впливати на підтримку та розширення додатку у майбутньому. Загалом, компоненти .xaml.cs є потужним інструментом для розробки мобільних додатків, але їх ефективне використання вимагає правильного проектування та організації коду, а також розуміння особливостей роботи з інтерфейсом користувача та обробки подій.

2.4. База даних SQLite

SQLite — полегшена реляційна система керування базами даних. Втілена у вигляді бібліотеки, де реалізовано функціонал зі стандарту SQL-92 [9]. База даних SQLite є одним з найпоширеніших та найпопулярніших варіантів баз даних для мобільних додатків. Вона забезпечує легку, швидку та ефективну роботу зі структурованою і нереляційною базою даних, що зберігається в одному файлі на пристрої.

Одна з основних переваг SQLite - його маленький розмір та низькі вимоги до системних ресурсів, що робить його ідеальним вибором для мобільних пристроїв з обмеженими ресурсами. Він володіє високою швидкістю виконання запитів, що дозволяє ефективно опрацьовувати великі обсяги даних.

SQLite підтримує стандартні SQL-запити, що дозволяють виконувати операції вставки, оновлення, видалення та вибірки даних. Вона також підтримує транзакції, що забезпечує цілісність та надійність даних. База даних SQLite може бути легко інтегрована з додатком, надаючи можливість зберігати та отримувати дані з дискового простору. Крім того, SQLite є переносним рішенням, яке підтримується на різних операційних системах, включаючи Android, iOS, Windows та інші. Це робить його універсальним інструментом для розробки мобільних додатків, які мають сумісність з різними платформами.

Одним з недоліків SQLite є його обмежена підтримка високорівневих операцій, таких як відображення об'єктно-орієнтованих моделей даних. У порівнянні з деякими іншими базами даних, SQLite має обмежені можливості для

роботи зі складними структурами даних, а також обмежені можливості для масштабування і роботи з великими обсягами даних. SQLite підходить переважно для невеликих та середніх проектів, де потреба у великій кількості одночасних з'єднань з базою даних або розподілених операцій є невеликою.

Проте, незважаючи на ці обмеження, SQLite все ще є популярним вибором для багатьох мобільних додатків завдяки своїм перевагам, таким як простота використання, швидкодія та переносність. Загалом, база даних SQLite є потужним та ефективним інструментом для розробки мобільних додатків, забезпечуючи надійне зберігання та операції з даними. При правильному використанні і оптимізації, вона може задовольнити потреби більшості мобільних додатків та забезпечити надійну роботу з базою даних у мобільному середовищі.

2.5 Висновки

Висновки щодо використання інструментів для розробки мобільних додатків:

1. Visual Studio: Visual Studio є потужним та універсальним інтегроване середовище розробки (IDE), яке надає широкі можливості для розробки мобільних додатків. Його переваги включають зручний інтерфейс, розширену підтримку мови C# та інтеграцію з іншими інструментами та сервісами.
2. Фреймворк .NET: Фреймворк .NET є потужним інструментом для розробки мобільних додатків, оскільки він надає розширені можливості програмування, включаючи підтримку мови C# та багатофункціональні бібліотеки.
3. Плагін Xamarin для Visual Studio: Плагін Xamarin дозволяє розробникам використовувати мову програмування C# та фреймворк .NET для розробки мобільних додатків для платформ Android та iOS. Він забезпечує переносимість коду та спрощує процес розробки.
4. Бібліотека для інструментів Android SDK: Бібліотека для інструментів Android SDK містить набір інструментів, бібліотек та ресурсів, які допомагають розробникам створювати мобільні додатки для платформи Android. Вона надає доступ до різноманітних функцій та можливостей платформи Android.

5. Декларативна мова розмітки XAML: XAML є мовою розмітки, яка дозволяє розробникам описувати інтерфейс користувача у декларативному стилі. Вона спрощує розробку та розміщення елементів у додатку, забезпечуючи розділення між логікою програми та представленням.

6. Об'єктно-орієнтована мова програмування C#: Мова C# є потужним інструментом для розробки мобільних додатків, оскільки вона надає розробникам багато можливостей об'єктно-орієнтованого програмування. Завдяки своїй синтаксичній зрозумілості та широкому набору функціональних можливостей, C# сприяє швидкому та ефективному написанню коду для розробки мобільних додатків.

7. Компоненти .xaml.cs: Компоненти .xaml.cs представляють код на мові C#, який взаємодіє з елементами інтерфейсу, визначеними в файлі XAML. Ці компоненти надають можливість керувати поведінкою та динамічною зміною інтерфейсу додатку, включаючи обробку подій, маніпулювання даними та виконання різних операцій.

8. База даних SQLite: SQLite є легковаго та вбудованою базою даних, яка може бути використана для зберігання та керування даними в мобільних додатках. Вона надає широкі можливості для роботи з реляційними даними та дозволяє ефективно зберігати, запитувати та оновлювати дані в додатку.

Застосування засобів розробки, таких як Visual Studio, фреймворк .NET, плагін Xamarin для Visual Studio, бібліотека для інструментів Android SDK, декларативна мова розмітки XAML, об'єктно-орієнтована мова програмування C#, компоненти .xaml.cs та база даних SQLite, є цілком доцільним для розробки мобільних додатків.

Ці засоби надають розробникам широкий набір інструментів та функціональних можливостей для створення якісних, швидких та ефективних мобільних додатків. Використання Visual Studio забезпечує зручне та продуктивне середовище розробки, а фреймворк .NET надає потужні можливості програмування та багатофункціональні бібліотеки.

3. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Під час розробки програмного продукту будуть виконані такі етапи:

• Розробка інтерфейсу програми

• Реалізація обробника запитів

• Створення бази даних

• Впровадження клієнт-серверної архітектури.

3.1 . Архітектура системи

Система включає такі складові:

• Frontend - інтерфейс, доступний для користувача через мобільний додаток;

• Backend - модуль, що обробляє запити від користувачів;

• База даних;

Архітектура додатку зображена на блок-схемі системи (рисунок 3.1).

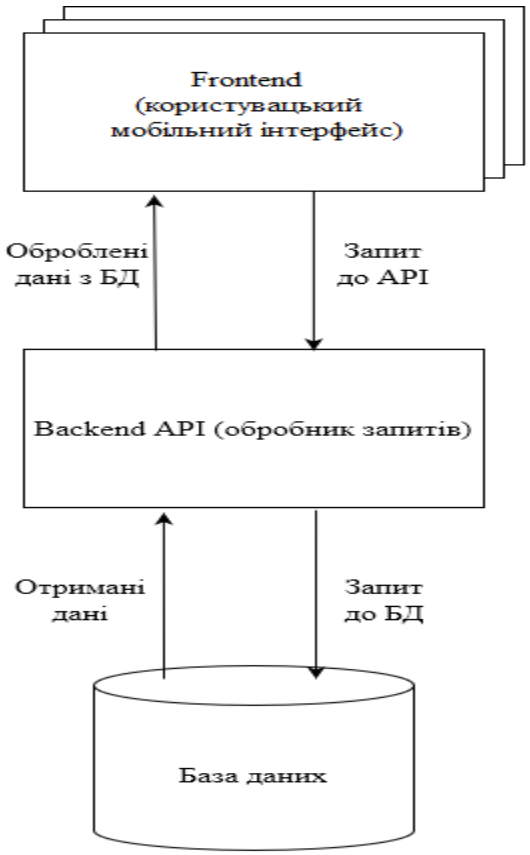


Рисунок 3.1 – Блок-схема архітектури мобільного додатку

Користувацький інтерфейс створений за допомогою розмітки XAML.

Обробник запитів написаний мовою C# на технології Xamarin для .NET.

База даних створена за допомогою реляційної бд SQLite.

3.2 Опис алгоритму програмної системи

1. Головна сторінка – перший екран додатку, на який користувач потрапляє при вході, відображає актуальний перелік доступних категорій правил етикету.
2. Спливаюче вікно додавання категорії – на яке попадає користувач з Головної сторінки, коли хоче додати нову категорію. Він заповнює всі необхідні дані для нової категорії, після чого вона додається в базу даних і з’являються на головній сторінці.
3. Сторінка категорії – сюди користувач попадає, коли заходить на одну з категорій з Головної сторінки. Тут відображаються всі групи, що належать до поточної категорії. Також, тут користувач має змогу редагувати дану категорію і створити в ній нову групу.
4. Спливаюче вікно редагування категорії– на яке попадає користувач зі Сторінки категорії, коли хоче провести маніпуляції над поточною категорією. Тут можна перейменувати або видалити категорію.
5. Спливаюче вікно додавання групи– на яке попадає користувач зі Сторінки категорії, коли хоче додати нову групу в середині категорії. Він заповнює всі необхідні дані для нової категорії, після чого вона додається в базу даних і з’являються на Сторінці категорії.
6. Сторінка групи – сюди користувач попадає, коли заходить на одну з категорій зі Сторінки категорії. Тут відображаються всі навчальні картки, що належать до поточної групи. Також, тут користувач має змогу редагувати дану групу і створити в ній нову навчальну картку або видалити картку.
7. Спливаюче вікно редагування групи – на яке попадає користувач зі Сторінки групи, коли хоче провести маніпуляції над поточною групою. Тут можна перейменувати або видалити групу.

8. Сторінка навчальної картки– на яку попадає користувач зі Сторінки групи, коли хоче додати або редагувати навчальну картку в середині поточної групи. Він заповнює всі необхідні дані для картки, а саме: поле «Слово або фраза», поле «Опис та приклад вживання», та поле «Додаткові примітки (необов'язково)», після чого картка зберігається в базі даних і з'являються на Сторінці групи.

3.3. Тестування та оцінка якості програмного продукту

3.3.1. Ручне тестування

Ручне тестування мобільного додатку є процесом, в ході якого тестувач виконує різні дії та перевірки на реальних мобільних пристроях з метою виявлення помилок, проблем та некоректностей у функціональності та взаємодії додатку з користувачем. Це важлива складова частина процесу розробки додатків, оскільки дозволяє перевірити, чи відповідає додаток заданим вимогам, працює стабільно і зручно для користувача.

Ручне тестування дозволяє отримати перший досвід користувача реальної людини. До того ж, для невеликих проектів розробка автоматизованих сценаріїв тестування може виявитися надто затратною [1].

Ручне тестування мобільного додатку включає наступні етапи:

1. Планування тестування: Тестувальник аналізує вимоги до додатку, розуміє його функціонал і визначає стратегію тестування. В цьому етапі визначаються основні сценарії тестування, розробляється тестова документація.
2. Підготовка тестового середовища: Тестувальник налаштовує мобільні пристрої для тестування, встановлює необхідні версії операційної системи та програмного забезпечення, зокрема додаток, який потрібно протестувати. Також в цьому етапі проводиться підготовка тестових даних.
3. Виконання тестування: Тестувальник виконує різні дії та перевірки у додатку, спостерігає за його реакцією та реагує на взаємодію з користувачем.

В цьому процесі тестувальник перевіряє різні аспекти додатку, включаючи функціональність, взаємодію з користувачем, реакцію на різні введення та сценарії використання. Він також виявляє та документує помилки, некоректну роботу, проблеми

4. Повторне тестування: Після виправлення виявлених помилок розробниками, тестувальник повторно перевіряє ці проблеми, переконуючись, що вони були виправлені та додаток працює коректно. Це важливий етап, оскільки він дозволяє переконатись у відсутності повторних помилок після внесення змін у код.

Ручне тестування мобільного додатку має свої переваги та обмеження. Основні переваги ручного тестування мобільного додатку включають:

1. Гнучкість: Ручне тестування дозволяє тестувальнику виконувати різноманітні дії та перевірки, що може виявити проблеми, які не були передбачені або виявлені автоматичними тестами. Тестувальник може пристосувати свій підхід до специфічних потреб та контексту додатку.
2. Легкість у виконанні: Ручне тестування не вимагає складних технічних знань або програмування. Це дозволяє залучати не лише розробників, але й інших членів команди, таких як тестувальники або представники замовника, до процесу перевірки додатку.
3. Здатність до емуляції реального користувача: Ручне тестування дозволяє тестувальнику спостерігати за реакцією додатку на різні дії та сценарії використання, що дозволяє зрозуміти, як добре додаток взаємодіє з користувачем, його зручність та ефективність.

Незважаючи на переваги, ручне тестування має також деякі недоліки та обмеження:

1. Час та ресурси: Ручне тестування може бути часо- та ресурсозатратним процесом, особливо для великих та складних додатків. Вимагає багато людських ресурсів та зусиль для виконання широкого спектру тестів та перевірок.

2. Людський фактор: Ручне тестування піддається людській помилці. Тестувач може пропустити деякі проблеми або не виявити їх через обмежений час, увагу або недостатню експертизу.
3. Обмежені масштаби: Ручне тестування може бути обмеженим у відношенні покриття функціональності та різноманітності мобільних пристроїв. Оскільки на ринку існує велика кількість пристроїв з різними операційними системами, версіями та конфігураціями, неможливо протестувати додаток на всіх з них.
4. Відсутність повторюваності: Ручне тестування може бути вразливим до залежності від навичок тестувача та його інтерпретації вимог. Результати тестування можуть варіюватись залежно від особистих знань, досвіду та умінь тестувача, що може призвести до неповторюваності результатів.
5. Неефективність для повторних тестів: Ручне тестування може бути неефективним для повторного виконання одних і тих самих тестів після внесення змін у додаток. Воно вимагає повторного виконання всіх кроків тестування, навіть тих, які не змінилися, що займає багато часу та ресурсів.

Незважаючи на ці обмеження, ручне тестування має своє місце в процесі розробки мобільних додатків, особливо на ранніх етапах, коли важливо швидко виявити помилки та зробити швидкі корективи. Воно може бути ефективним для тестування нових функцій, взаємодії з користувачем та проведення експериментальних перевірок.

3.3.2. Автоматизоване тестування

Автоматизоване тестування мобільних додатків є процесом використання спеціальних програмних засобів та інструментів для виконання тестових сценаріїв та перевірок без прямої участі людини. Воно спрощує та прискорює процес тестування, дозволяючи автоматично виконувати тести, аналізувати результати та виявляти проблеми.

Автоматизоване тестування мобільних додатків включає наступні етапи:

1. Вибір інструментарію: Важливо вибрати підходящі інструменти для автоматизованого тестування мобільних додатків. Інструменти можуть включати фреймворки для написання тестових скриптів, засоби запису та відтворення дій, інструменти для взаємодії з мобільними пристроями тощо.
2. Розробка тестових скриптів: Тестувальник розробляє тестові скрипти, що включають послідовність дій та перевірок, які мають бути виконані автоматично. Ці скрипти можуть бути написані мовами програмування, які підтримуються вибраним інструментарієм.
3. Налаштування тестового середовища: Тестувальник налаштовує середовище для автоматизованого тестування, включаючи підключення мобільних пристроїв або емуляторів, встановлення необхідного програмного забезпечення та налаштування тестових даних.
4. Виконання автоматизованих тестів: Після налаштування тестового середовища тестувальник запускає автоматизовані тестові скрипти, які автоматично взаємодіють з мобільним додатком, виконують дії та перевірки. Це може включати запуск додатку на різних пристроях або емуляторах, введення даних, натискання на кнопки, перевірку відповідей та результатів, аналіз поведінки додатку тощо.
5. Аналіз результатів: Після завершення автоматизованих тестів, інструменти аналізують результати виконання, що включає інформацію про пройдені тести, виявлені помилки та недоліки. Тестувальник аналізує ці результати для виявлення проблем та прийняття рішень стосовно подальших дій.
6. Звіт та затвердження: Тестувальник генерує звіт, в якому відображаються результати автоматизованих тестів, виявлені проблеми, їх опис, кроки для відтворення та інші деталі. Цей звіт подається команді розробників та іншим зацікавленим сторонам для подальшого аналізу та прийняття рішень.

Автоматизоване тестування мобільних додатків дозволяє забезпечити більшу швидкість та ефективність процесу тестування, скоротити час виконання тестів та покращити покриття функціональності. Воно особливо корисне при регресійному тестуванні, коли необхідно перевірити, чи впливають внесені зміни на раніше

працюючий функціонал додатку. Також автоматизоване тестування сприяє виявленню проблем, які можуть бути важко виявити ручним способом, таких як перевірка великих обсягів даних або взаємодії з різними пристроями та конфігураціями.

Цикл розробки вимагає багаторазового виконання одного й того ж набору тестів під час послідовності розробки. Використовуючи автоматизацію, можна написати набір тестів і відтворювати його повторно у разі необхідності. Як тільки набір тестів автоматизовано, втручання людини не потрібне [2]. Тестування продуктивності (навантажувальне, стресове, об’ємне) проводиться з метою перевірки працездатності продукту в умовах, максимально наближених до реальних, з очікуваними навантаженнями та обсягом даних. [3]

Деякі з обмежень автоматизованого тестування мобільних додатків включають:

1. Складність в розробці: Розробка автоматизованих тестових скриптів може бути вимогливою і часо-затратною задачею. Це вимагає знань програмування та досвіду в автоматизації тестування.
2. Вразливість до змін: Якщо мобільний додаток часто зазнає змін у своїй функціональності або інтерфейсі, автоматизовані тестові скрипти можуть вимагати постійного оновлення та налаштування, щоб вони продовжували коректно працювати.
3. Недостатня репрезентативність: Автоматизовані тестові скрипти можуть не завжди забезпечувати повну репрезентативність реального використання додатку. Вони можуть не враховувати всі можливі сценарії взаємодії користувача та потенційні проблеми, які можуть виникати в реальному середовищі.
4. Витрати на підтримку тестових скриптів: Підтримка автоматизованих тестових скриптів вимагає часу та ресурсів. Якщо додаток регулярно зазнає змін або має велику кількість тестових сценаріїв, підтримка може стати складною та витратною задачею.
5. Відсутність "людського інтуїтивного розуміння": Автоматизовані тестові скрипти не завжди можуть виявити проблеми, які можуть бути помічені

людиною через її інтуїтивне розуміння або досвід. Певні типи помилок можуть бути важкими для автоматизованого виявлення.

Незважаючи на ці обмеження, автоматизоване тестування мобільних додатків має багато переваг і є незамінним елементом процесу тестування. Деякі з переваг автоматизованого тестування мобільних додатків включають:

1. Ефективність і часозбереження: Автоматизоване тестування дозволяє виконувати велику кількість тестів швидше та ефективніше, порівняно з ручним тестуванням. Воно може виконувати тестові сценарії відразу на кількох пристроях або емуляторах, що дозволяє скоротити час, необхідний для проведення тестування.
2. Покриття функціональності: Автоматизоване тестування дозволяє охоплювати більшу кількість функціональності додатку. Завдяки автоматизації, можна виконувати тести на різних пристроях, з різними операційними системами та конфігураціями, що забезпечує широке покриття.
3. Регресійне тестування: Автоматизовані тестові скрипти дозволяють швидко та автоматично виконувати регресійне тестування, перевіряючи, чи не впливають внесені зміни на раніше працюючий функціонал. Це допомагає виявити проблеми, що можуть виникнути внаслідок змін у додатку.
4. Підвищення якості: Автоматизоване тестування дозволяє виявляти помилки, що можуть бути пропущені під час ручного тестування. Воно допомагає виявляти проблеми швидше, робить тестування більш систематичним та повторюваним, тим самим підвищуючи якість додатку.

Автоматизоване тестування мобільних додатків є важливою та необхідною складовою процесу тестування. Воно дозволяє забезпечити ефективність, швидкість та покриття функціональності, підвищуючи якість додатку. Автоматизація тестування зменшує час, необхідний для виконання тестів, дозволяє проводити регресійне тестування та забезпечує більшу точність виявлення помилок. Втім, варто пам'ятати про обмеження автоматизованого тестування, такі

як складність в розробці та підтримці тестових скриптів, недостатню репрезентативність та вразливість до змін у додатку. Комбінація автоматизованого та ручного тестування забезпечує найкращі результати, допомагаючи забезпечити високу якість мобільних додатків.

3.4. Системні вимоги

Для запуску додатку на телефоні Android потрібно враховувати наступні системні вимоги:

1. Операційна система: Android версії 4.4 (KitKat) або вище.
2. Процесор: ARM або x86 архітектура з підтримкою NEON-інструкцій.
3. Оперативна пам'ять (RAM): Рекомендовано мінімум 1 ГБ RAM для оптимальної продуктивності.
4. Внутрішня пам'ять: Наявність достатнього вільного місця для встановлення додатку. Розмір додатку може варіюватися в залежності від його складності.
5. Дисплей: Роздільна здатність екрану 480x800 пікселів або вище.
6. Інтернет-підключення: Для деяких функцій додатку може знадобитися доступ до Інтернету, тому необхідне активне підключення до Wi-Fi або мобільної мережі.
7. Версія програмного забезпечення: Додаток може вимагати певну мінімальну версію Android, яка залежить від використовуваних функцій та API.

Враховуючи ці системні вимоги, додаток зможе запускатися та працювати на телефоні з ОС Android, який відповідає вимогам, і забезпечувати задану функціональність для користувача.

Основні системні вимоги для комп'ютера, на якому можна запускати даний програмний продукт через ПЗ «Android Emulator», включають:

1. Операційна система: Комп'ютер повинен працювати під управлінням операційної системи, яка підтримує віртуалізацію ізоляції, такою як Windows, macOS або Linux.

2. Процесор: Рекомендується мати процесор з архітектурою x86 або x86_64, оскільки це забезпечує кращу сумісність та продуктивність для емуляції Android.
3. Пам'ять (RAM): Мінімальна кількість оперативної пам'яті для запуску Android Emulator зазвичай становить 4 ГБ, проте рекомендується мати 8 ГБ або більше для оптимальної продуктивності.
4. Простір на жорсткому диску: Для встановлення Android Emulator і збереження образів Android-пристроїв необхідно мати достатньо вільного простору на жорсткому диску. Загалом, рекомендується мати щонайменше 2 ГБ вільного простору.
5. Віртуалізація: На комп'ютері повинна бути включена підтримка апаратної віртуалізації (наприклад, Intel VT або AMD-V), оскільки Android Emulator використовує віртуалізацію для емуляції Android-пристроїв.
6. Графічна підсистема: Комп'ютер повинен мати підтримку OpenGL 2.0 або новіше для відображення графічних елементів в Android Emulator.

Варто зауважити, що конкретні системні вимоги можуть залежати від версії Android Emulator і операційної системи, тому рекомендується перевірити документацію та рекомендації виробника для точних вимог.

3.5 Висновки

Загальним висновком є те, що розробка мобільної програмної системи для вивчення правил етикету є складним та відповідальним процесом. Архітектура системи була розроблена з урахуванням потреб користувачів та вимог до продукту. Опис алгоритму програмної системи включає розробку зручного та інтуїтивно зрозумілого інтерфейсу для користувачів. Тестування та оцінка якості програмного продукту здійснювалась як ручним, так і автоматизованим тестуванням, з метою забезпечення стабільності та надійності додатку. Враховуючи системні вимоги, були забезпечені необхідні ресурси для ефективної роботи програмної системи.

Розроблений додаток має потенціал стати корисним інструментом для вивчення правил етикету та поліпшення поведінки у різних сферах життя.

4. РОБОТА КОРИСТУВАЧА З ІНТЕРФЕЙСОМ

4.1 Перший вхід в додаток

Користувач відкриває додаток і попадає на головну сторінку, де розташовані наступні елементи:

- Список категорій правил етикету
- Кнопка «Нова категорія»

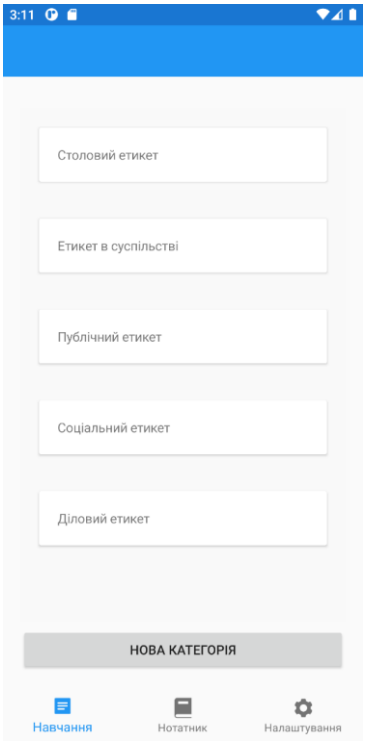


Рисунок 4.1 – Головна сторінка

4.2 Додавання нової категорії

Користувач натискає на кнопку «Нова категорія», після чого на екрані спливає вікно додавання категорії з текстовим полем для введення назви категорії. У вікні є дві можливих опції:

- Кнопка «Ок» - для підтвердження додавання нової категорії.
- Кнопка «Відмінити» для скасування операції та закриття вікна.

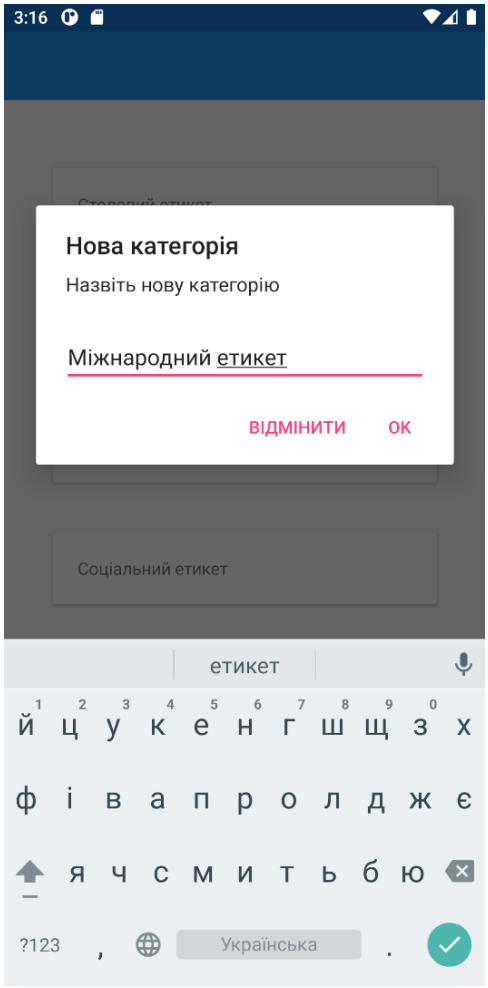


Рисунок 4.2 – Вікно додавання нової категорії

4.2.1 Результат додавання категорії

Після успішного додавання, категорія одразу з’являється та відображається на головній сторінці:

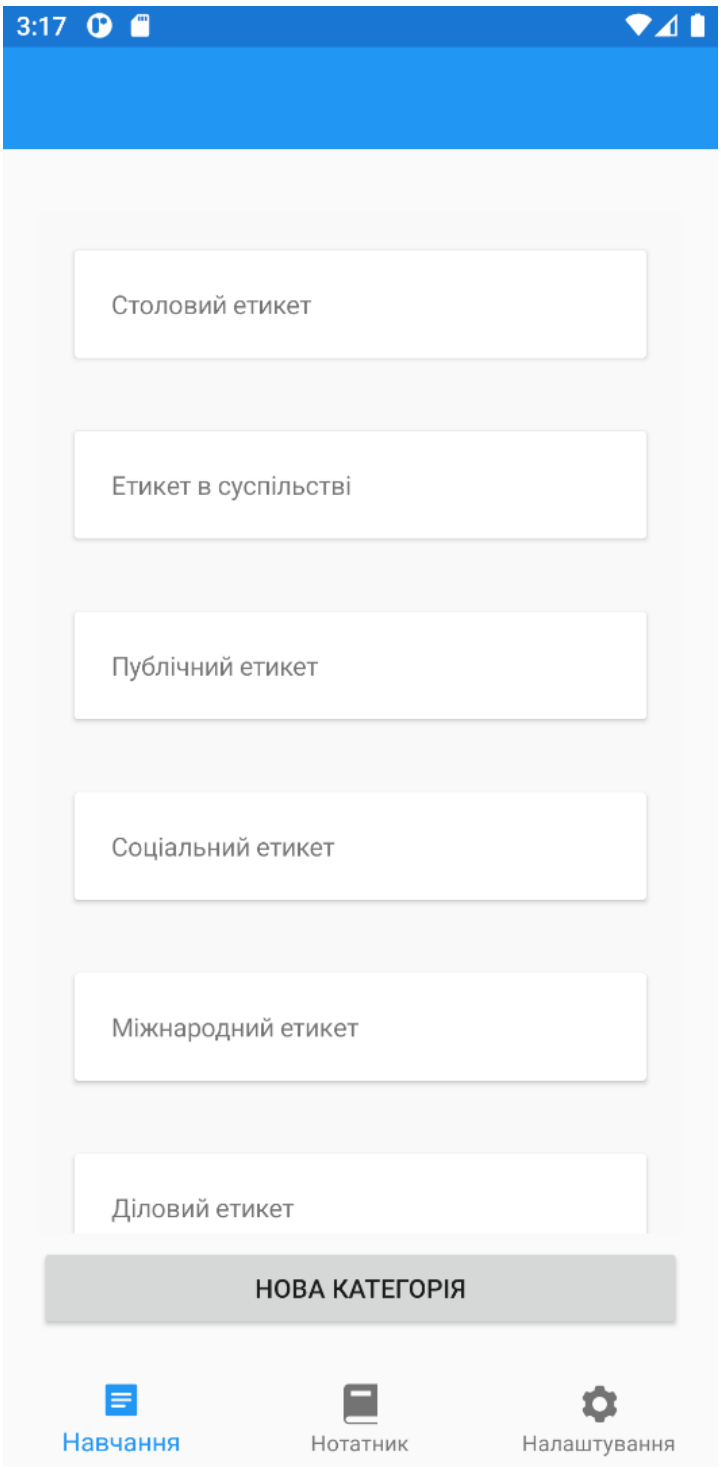


Рисунок 4.2.1 – Результат додавання категорії на головній сторінці

4.3 Сторінка категорії

Користувач переходить до обраної категорії (наприклад, «Соціальний етикет») з головної сторінки та опиняється на її сторінці, де відображені наступні елементи:

• Навігація по додатку у шапці:

1. Кнопка «Повернутись назад» (стрілка вліво)
2. Назва поточної категорії
3. Кнопка «Опції» (три крапки вертикально) для редагування категорії.

• Список груп, що належать до поточної категорії

• Кнопка «Нова група» для створення нової групи в цій категорії (працює так само, як і «Нова категорія»)

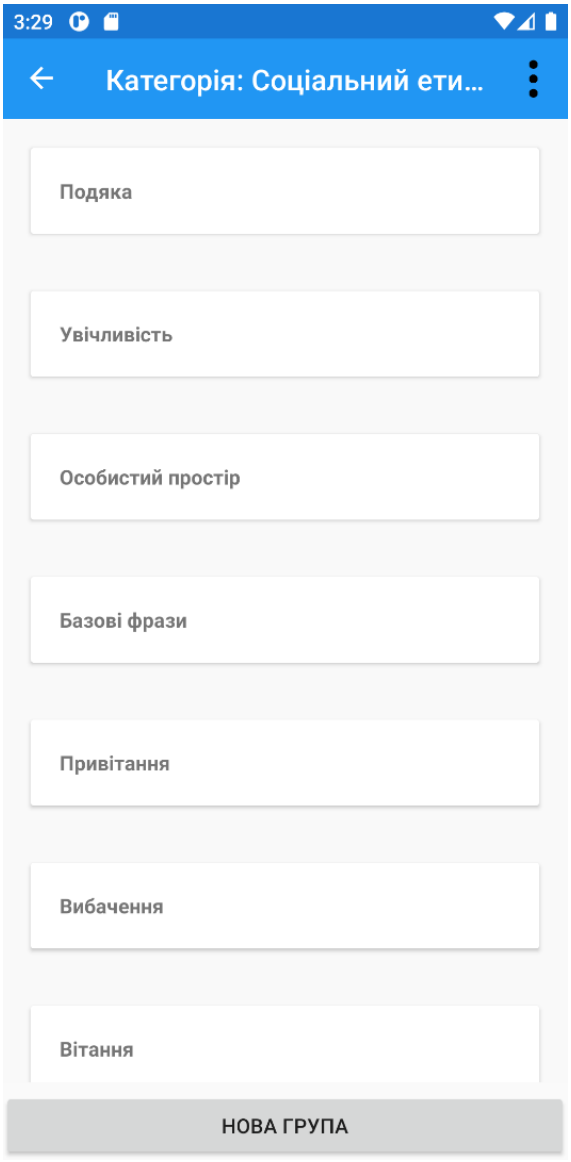


Рисунок 4.3 – Сторінка категорії

4.3.1 Редагування категорії

Після натискання кнопки «Опції», на екрані спливає вікно редагування категорії. У вікні є три можливих опції:

- Кнопка «Перейменувати категорію» - для надання категорії нової назви
- Кнопка «Видалити категорію» - для видалення категорії з бази даних
- Кнопка «Відмінити» для скасування операції та закриття вікна

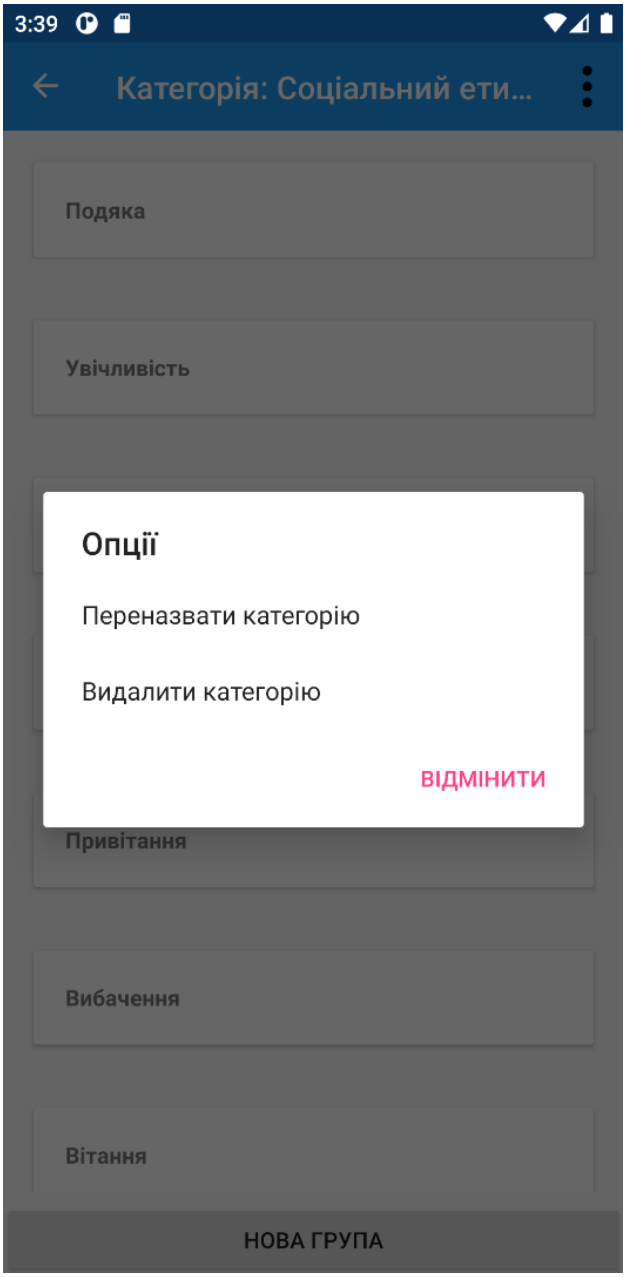


Рисунок 4.3.1 – Спливаюче вікно «Опції» для редагування категорії

4.3.2 Перейменування категорії

Користувач натискає на кнопку «Перейменувати категорію», після чого на екрані спливає вікно «Перейменувати цю категорію» з текстовим полем для введення нової назви категорії. У вікні є дві можливих опції:

- Кнопка «Підтвердити» - для підтвердження перейменування категоріх
- Кнопка «Відмінити» для скасування операції та закриття вікна

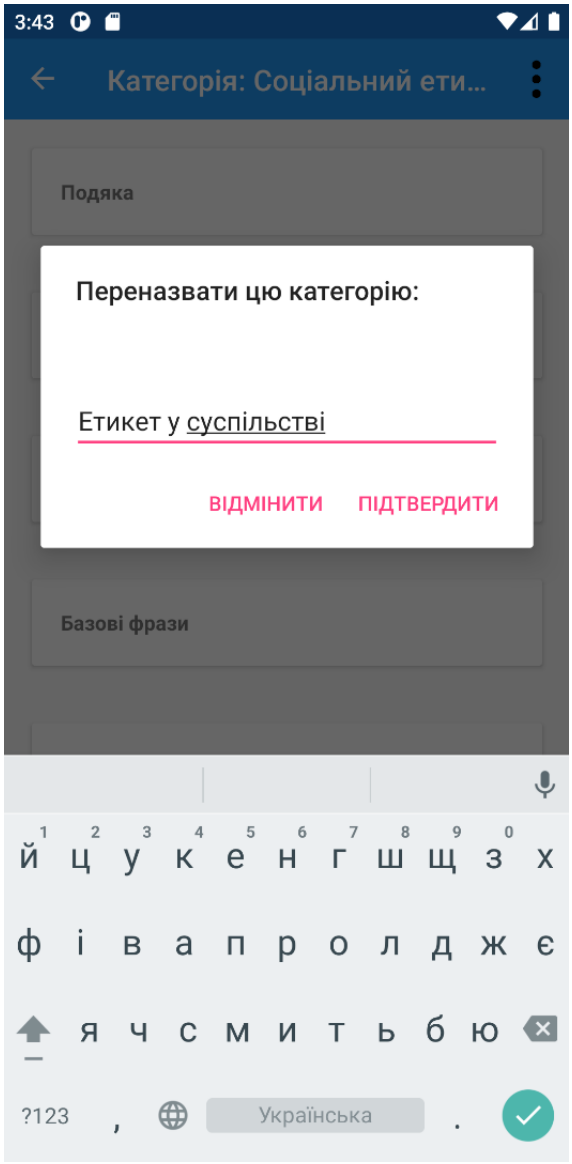


Рисунок 4.3.2 – Спливаюче вікно «Перейменувати цю категорію»

4.3.3 Результат перейменування категорії

Після успішного перейменування, нова назва категорії одразу відображається в шапці навігації:

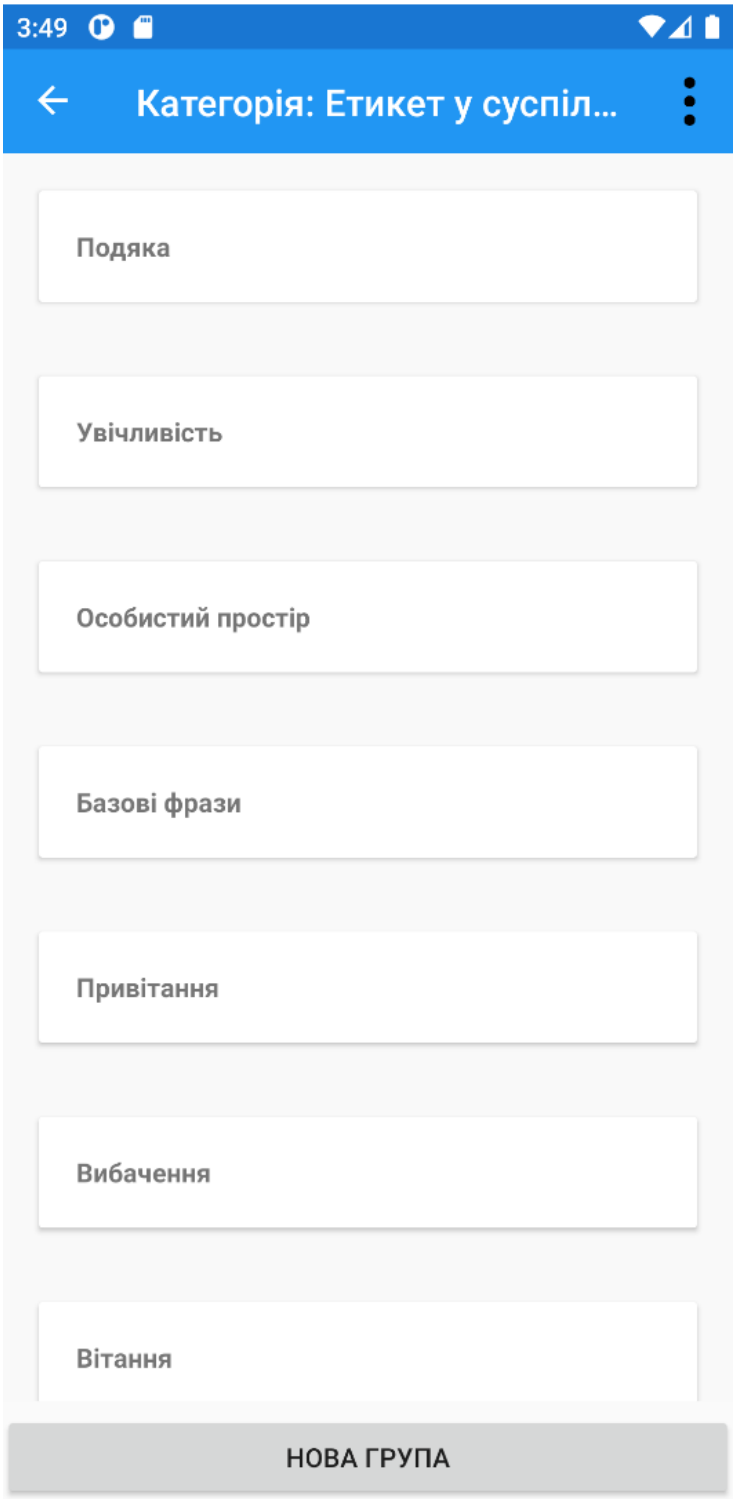


Рисунок 4.3.2 – Результат перейменування категорії

4.4 Сторінка групи

Користувач переходить до обраної групи (наприклад, «Привітання») зі сторінки категорії та опиняється на відповідній сторінці, де відображені наступні елементи:

• Навігація по додатку у шапці:

1. Кнопка «Повернутись назад» (стрілка вліво)
2. Назва поточної групи
3. Кнопка «Опції» (три крапки вертикально) для редагування групи (працює так само, як і «Опції» на сторінці категорії).

• Список навчальних карток, що належать до поточної групи, з кнопкою «Видалення» (смітник) на кожній із карток.

• Кнопка «Нова картка» для створення нової навчальної картки в цій групі

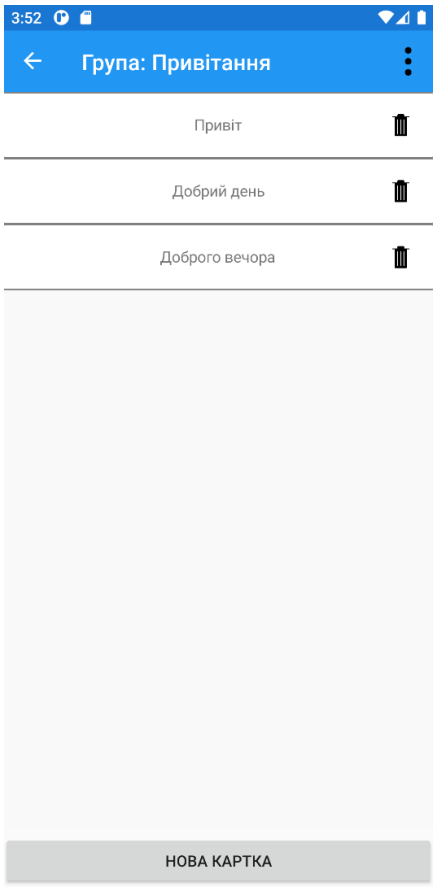


Рисунок 4.4 – Вікно групи

4.4.1 Видалення навчальної картки

Після натискання кнопки «Видалення» на певній картці (наприклад «Привіт»), на екрані спливає вікно «Дійсно видалити цю картку?» для перевірки випадковості наміру користувача. У вікні є дві можливих опції:

- Кнопка «Видалити» - для видалення навчальної картки з бази даних
- Кнопка «Відмінити» для скасування операції та закриття вікна

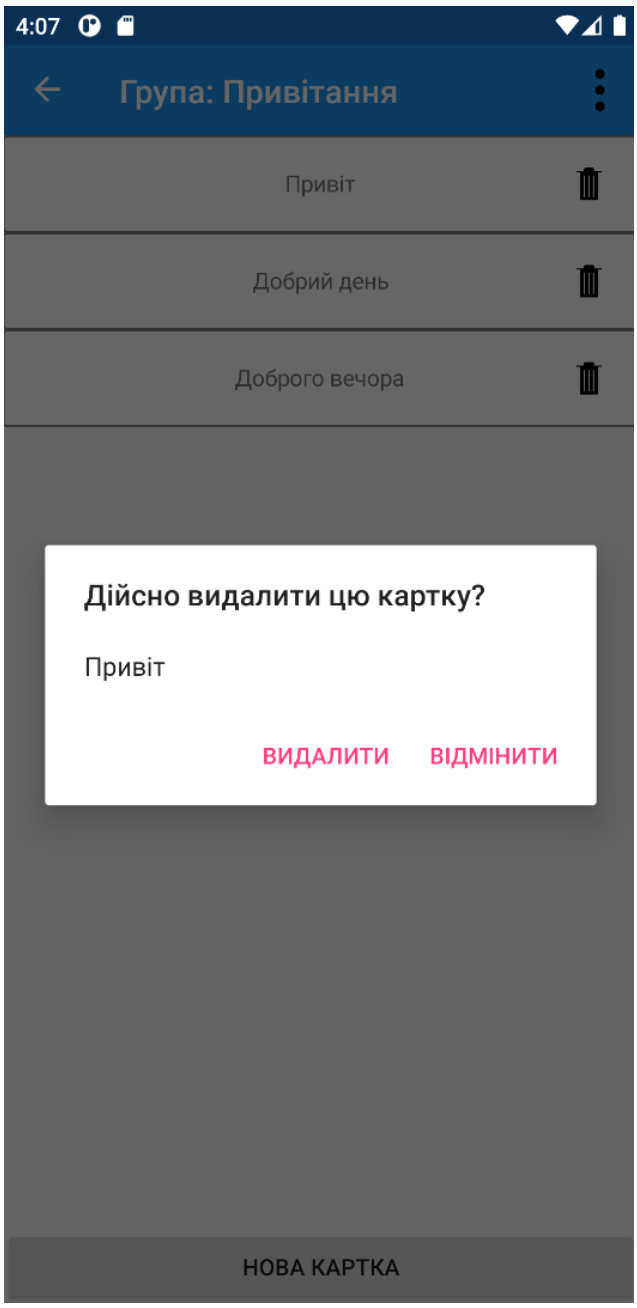


Рисунок 4.4.1 – Спливаюче вікно «Опції» для редагування категорії

4.4.2 Результат видалення навчальної картки

Після підтвердження видалення, картка зникає зі сторінки групи:

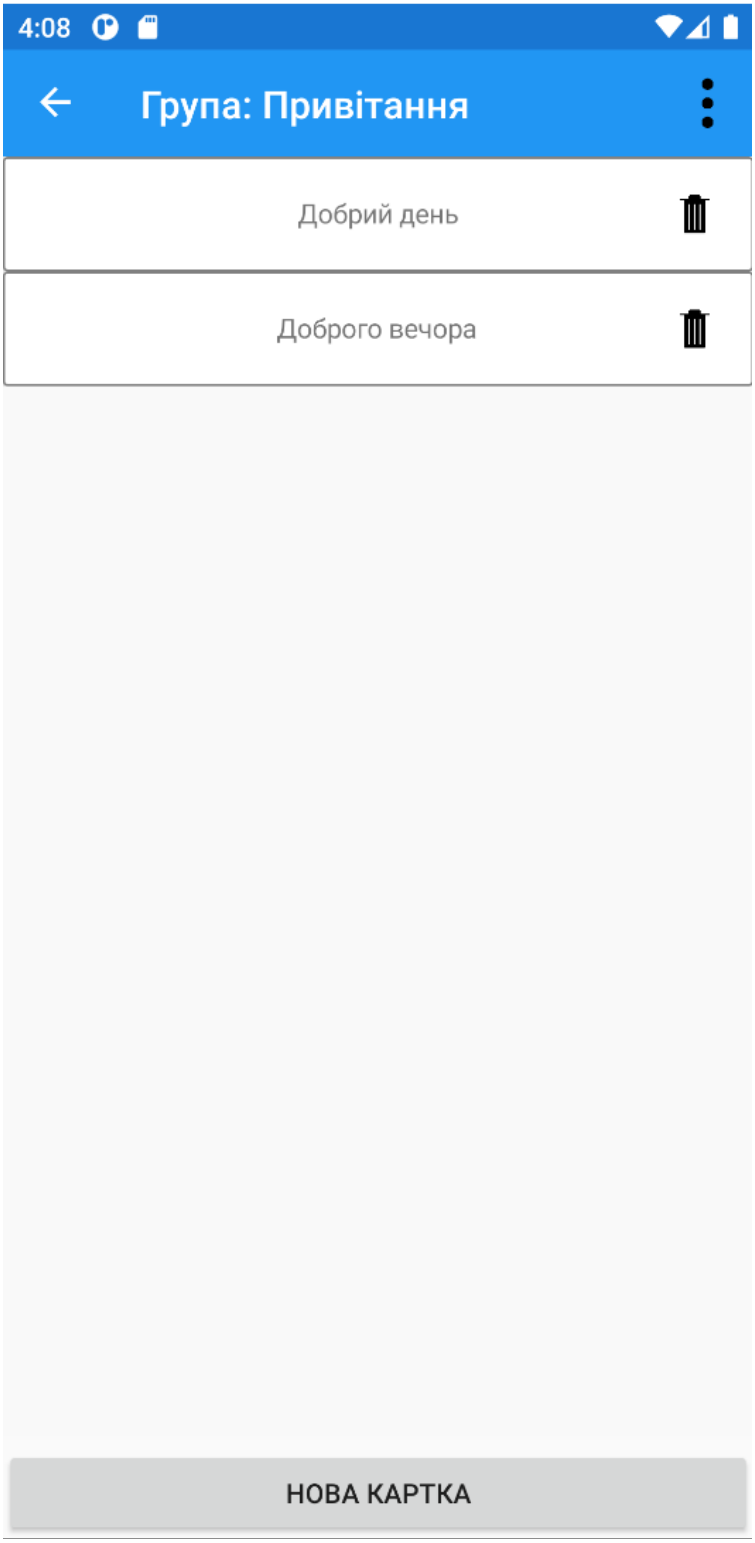


Рисунок 4.4.2 – Результат перейменування категорії

4.5 Сторінка навчальної картки

Користувач натискає кнопку «Нова картка» (або на будь яку картку) на сторінці групи, після чого опиняється на сторінці картки, де відображені наступні елементи:

- Навігація по додатку у шапці:
    1. Кнопка «Повернутись назад» (стрілка вліво)
    2. Назва картки (за наявності)
- Текстове поле для введення «Слово або фраза».
- Текстове поле для введення «Опис та приклад вживання».
- Текстове поле для введення «Додаткові примітки (необов'язково)».
- Кнопка «Зберегти» для збереження вмісту навчальної картки в базі даних.
- Кнопка «Стерти» для спустошення вмісту текстових полів.

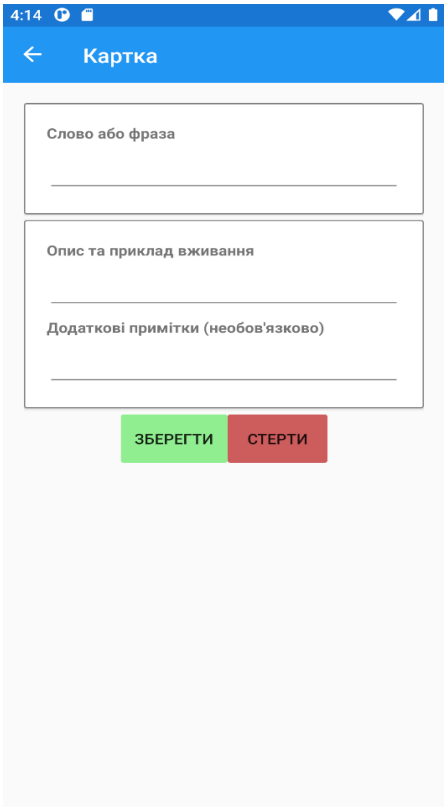


Рисунок 4.5 – Сторінка картки

4.5.1 Заповнена навчальна картка

Користувач заповнює картку, після чого натискає «Зберегти», після чого попаде назад на сторінку групи. У випадку створення нової картки, вона з’явиться на сторінці групи. Так само вміст будь якої картки можна редагувати і зберегти, після чого вона оновиться.

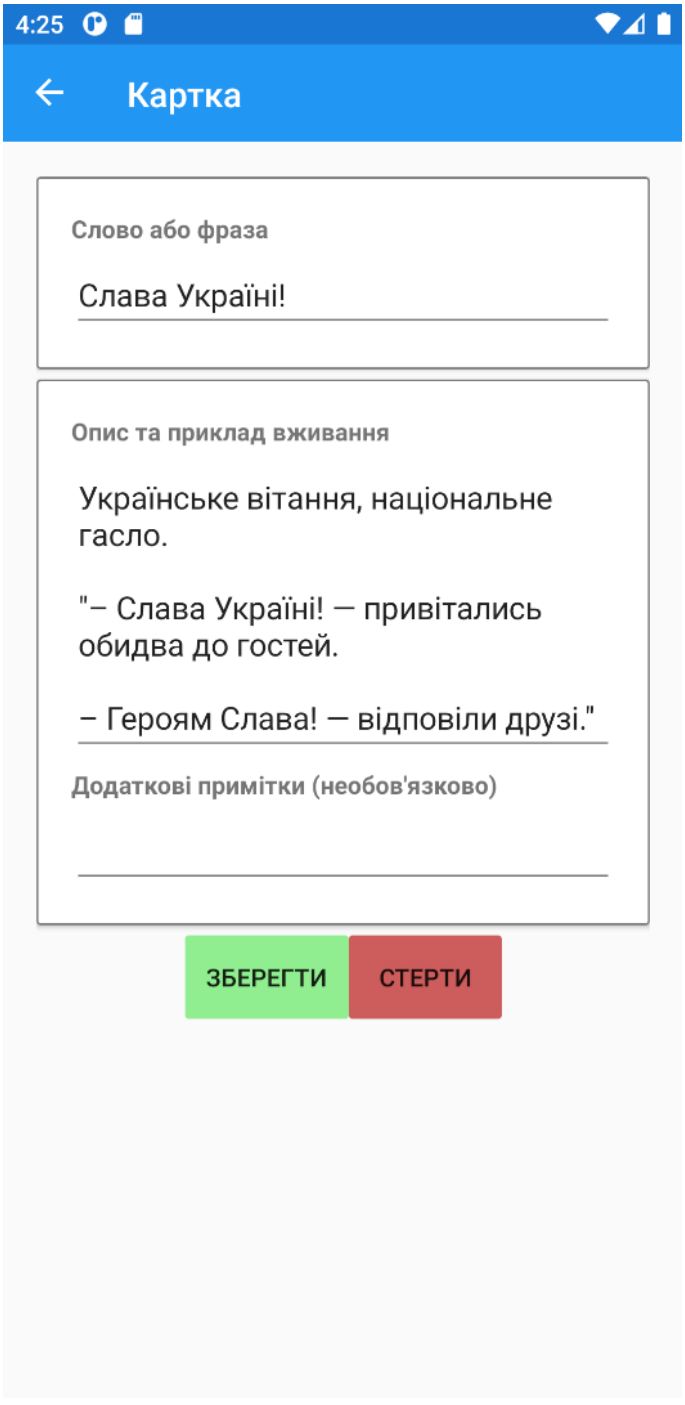


Рисунок 4.5.1 – Заповнена сторінка картки

4.5.2 Результат додавання навчальної картки

Після збереження картки вона відображається на сторінці групи:

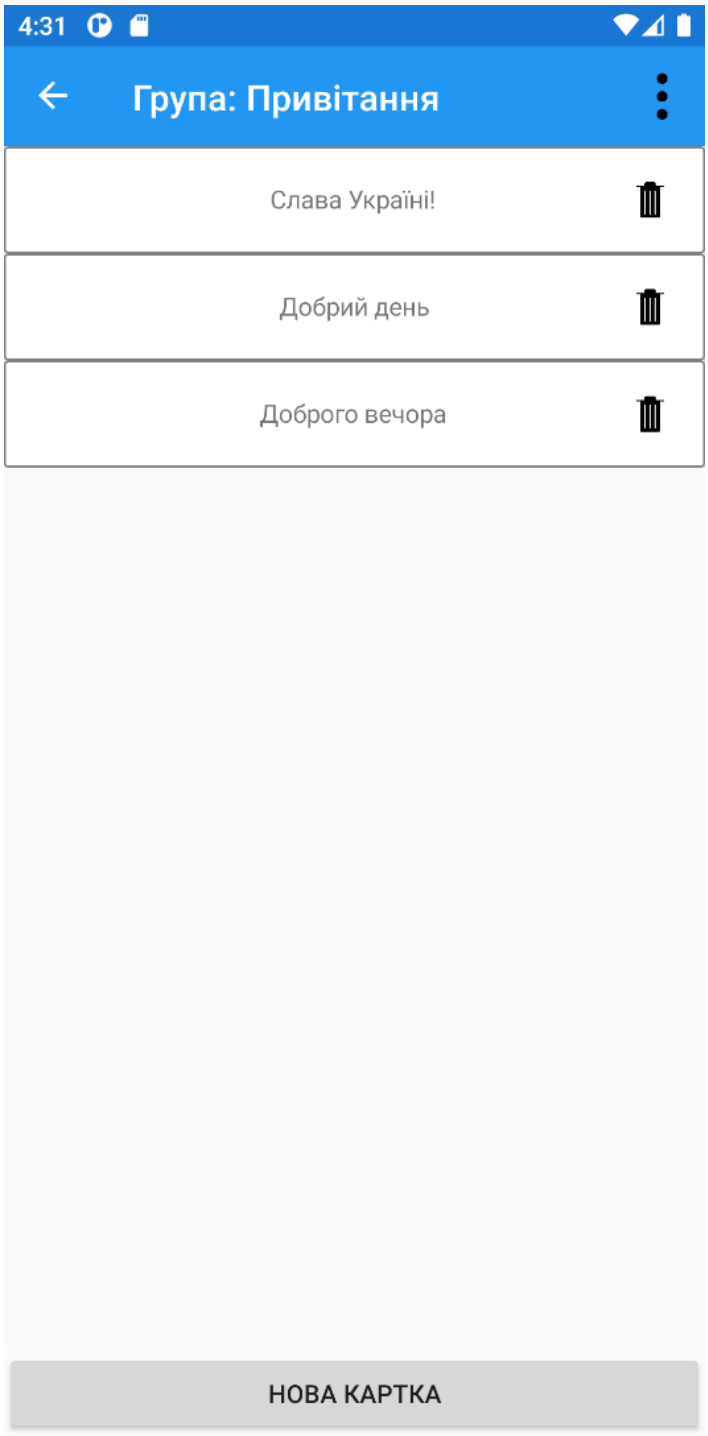


Рисунок 4.5.2 – Нова картка з’явилась на сторінці групи

4.6. Висновки

В розділі "Робота користувача з інтерфейсом" детально описано взаємодію користувача з додатком по вивченню правил етикету. Висновки щодо цього розділу наступні:

1. Перший вхід в додаток є простим і зручним процесом, який дозволяє користувачеві швидко розпочати вивчення правил етикету.
2. Додавання нової категорії дозволяє користувачеві організувати свій вміст за певними темами або контекстами. Ця функція робить додаток більш гнучким і пристосованим до індивідуальних потреб користувача.
3. Сторінка категорії надає зручний інтерфейс для перегляду та управління навчальними картками в межах обраної категорії. Користувач може редагувати, перейменовувати або видаляти картки, що дозволяє зберігати і актуалізувати вміст відповідно до своїх потреб.
4. Сторінка групи дозволяє користувачеві переглядати картки в межах обраної групи. Видалення навчальної картки є простим і швидким процесом, який дозволяє користувачеві управляти своїм вмістом.
5. Сторінка навчальної картки надає можливість користувачеві заповнювати картку з необхідною інформацією про правила етикету. Результат додавання навчальної картки є візуально зручним та забезпечує легкий доступ до збереженої інформації.

Загальним висновок щодо розділу "Робота користувача з інтерфейсом" є те, що додаток надає зручний і простий інтерфейс, що дозволяє користувачам ефективно вивчати та організовувати

ВИСНОВКИ

Дипломна робота присвячена розробці мобільного додатку для вивчення правил етикету. У розділі вступу було викладено актуальність теми та визначено мету та завдання дослідження.

У першому розділі був проведений огляд існуючих мобільних додатків з вивчення етикету, де було розглянуто додаток "Etiquette" і "Діловий Етикет". На основі проведеного аналізу були зроблені висновки.

У другому розділі було описано засоби розробки програмного продукту, зокрема середовище розробки Visual Studio, фреймворк .NET, плагін Xamarin для Visual Studio, бібліотеку для інструментів Android SDK, декларативну мову розмітки XAML, об'єктно-орієнтовану мову програмування C# та базу даних SQLite. Висновки даного розділу стосуються засобів, які були використані для розробки програмного продукту.

Третій розділ містить опис програмної реалізації, зокрема архітектуру системи, опис алгоритму програмної системи, тестування та оцінку якості продукту та системні вимоги. Висновки даного розділу стосуються опису та оцінки реалізованої програмної системи.

У четвертому розділі була описана робота користувача з інтерфейсом додатку. Були розглянуті кроки першого входу в додаток, додавання нової категорії, редагування та перейменування категорії, робота зі сторінкою групи та навчальної картки. Висновки даного розділу стосуються зручності та ефективності роботи користувача з інтерфейсом.

Поставлені завдання та мета були досягнуті. Було розроблено мобільний додаток для вивчення правил етикету, який забезпечує зручний і простий інтерфейс для користувачів. Додаток дозволяє користувачам вивчати правила етикету через навчальні картки, організовані за категоріями і групами. Користувачі можуть додавати нові категорії, редагувати і перейменовувати їх, а також видаляти навчальні картки. Додаток також надає можливість заповнювати навчальні картки з необхідною інформацією про правила етикету.

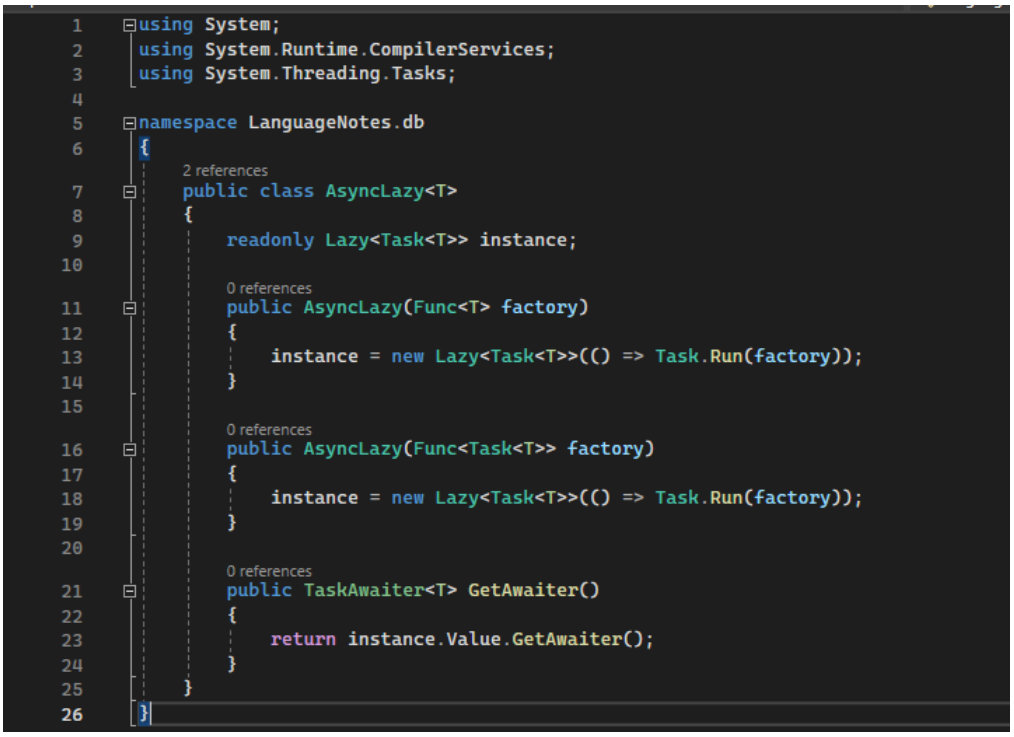
Загальний висновок полягає в тому, що розроблений мобільний додаток є ефективним інструментом для вивчення правил етикету. Він спрощує доступ до інформації про етикет, дозволяє користувачам організовувати і керувати навчальним матеріалом та забезпечує зручний інтерфейс взаємодії. Поставлені завдання та мета дипломної роботи були успішно виконані, а розроблений додаток має потенціал для подальшого розширення та вдосконалення.

ПЕРЕЛІК ПОСИЛАНЬ

1. Тестування мобільних додатків – методи та особливості [Електронний ресурс] / Avada-media – Режим доступу: https://avada-media.ua/ua/services/testirovaniye-mobilnykh-prilozheniy-metody-i-osobennosti
2. Автоматизоване тестування [Електронний ресурс] / Qalight – Режим доступу: https://qalight.ua/baza-znaniy/avtomatizovane-testuvannya/
3. Автоматизуємо тестування: коли, навіщо і кому це потрібно [Електронний ресурс] / Newline.tech – Режим доступу: https://newline.tech/test-automation-when-why-and-who-needs-it_uk/
4. Microsoft Visual Studio [Електронний ресурс] / Wikipedia.. – Режим доступу: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio
5. NET_Framework [Електронний ресурс] / Wikipedia.. – Режим доступу: https://uk.wikipedia.org/wiki/.NET_Framework
6. Xamarin [Електронний ресурс] / Wikipedia.. – Режим доступу: https://en.wikipedia.org/wiki/Xamarin
7. XAML – мова розмітки інтерфейсі Silverlight додатків [Електронний ресурс] / Портал знань – Режим доступу: http://www.znannya.org/?view=silverlight-intro1
8. C# [Електронний ресурс] / Wikipedia – Режим доступу: https://uk.wikipedia.org/wiki/C_Sharp
9. SQLite [Електронний ресурс] / Wikipedia – Режим доступу: https://uk.wikipedia.org/wiki/SQLite

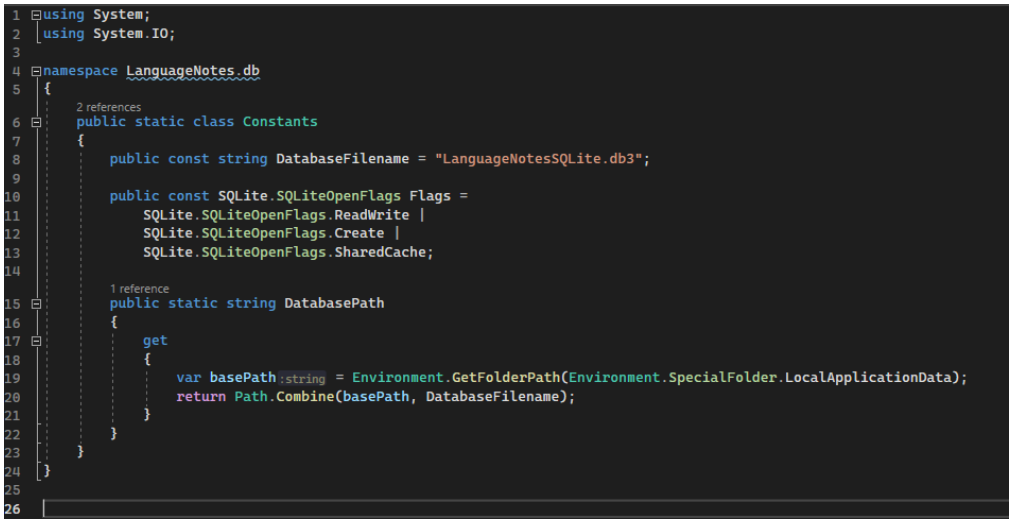
ДОДАТОК А

Приклади з коду



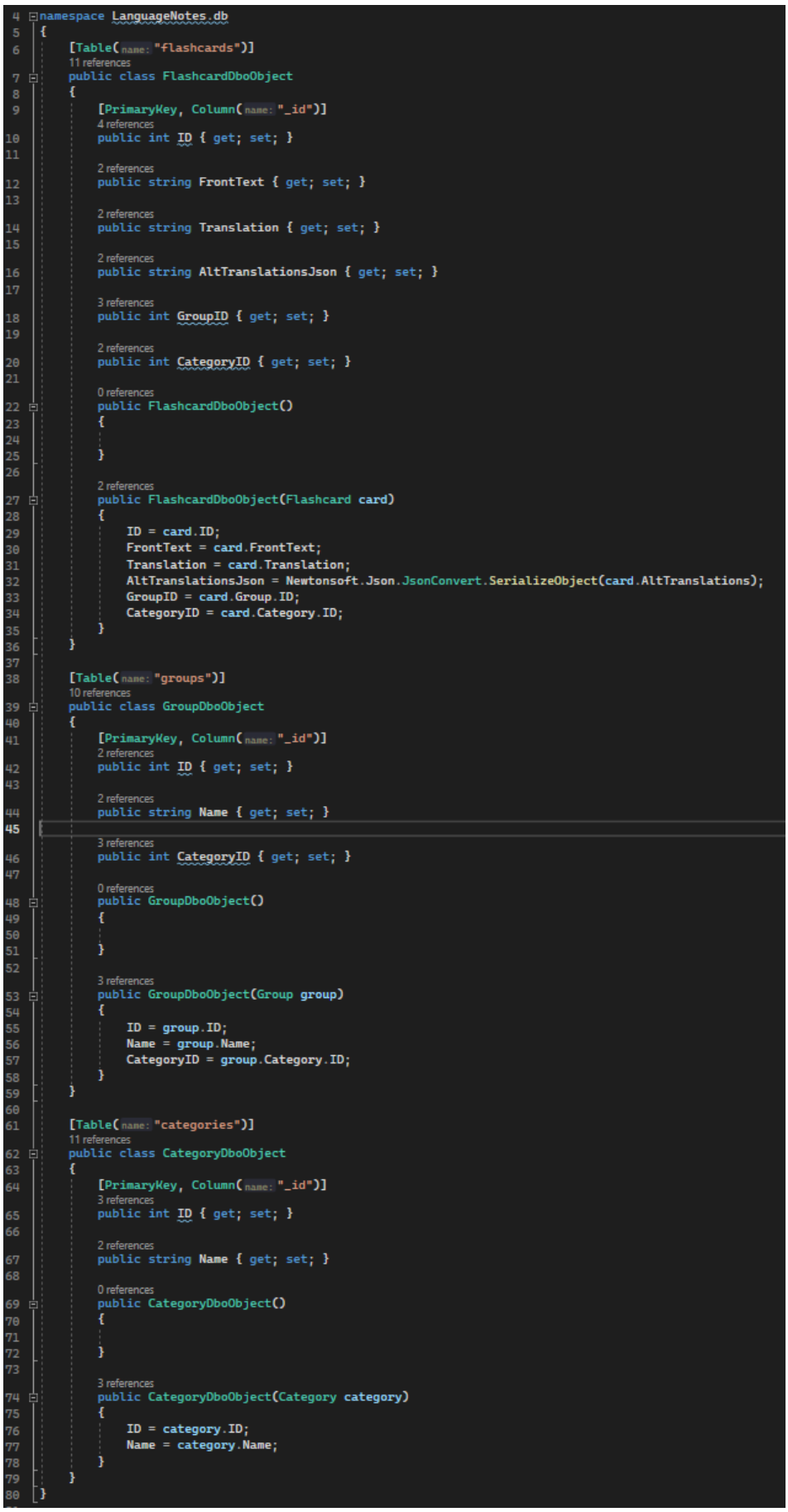
```
using System;
using System.Runtime.CompilerServices;
using System.Threading.Tasks;

namespace LanguageNotes.db
{
    2 references
    public class AsyncLazy<T>
    {
        readonly Lazy<Task<T>> instance;

        0 references
        public AsyncLazy(Func<T> factory)
        {
            instance = new Lazy<Task<T>>(() => Task.Run(factory));
        }

        0 references
        public AsyncLazy(Func<Task<T>> factory)
        {
            instance = new Lazy<Task<T>>(() => Task.Run(factory));
        }

        0 references
        public TaskAwaiter<T> GetAwaiter()
        {
            return instance.Value.GetAwaiter();
        }
    }
}
```

Рисунок 1. – Підключення асинхронності для взаємодії з базою даних



```
using System;
using System.IO;

namespace LanguageNotes.db
{
    2 references
    public static class Constants
    {
        public const string DatabaseFilename = "LanguageNotesSQLite.db3";

        public const SQLite.SQLiteOpenFlags Flags =
            SQLite.SQLiteOpenFlags.ReadWrite |
            SQLite.SQLiteOpenFlags.Create |
            SQLite.SQLiteOpenFlags.SharedCache;

        1 reference
        public static string DatabasePath
        {
            get
            {
                var basePath :string = Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData);
                return Path.Combine(basePath, DatabaseFilename);
            }
        }
    }
}
```

Рисунок 2. – Заведення констант для бази даних

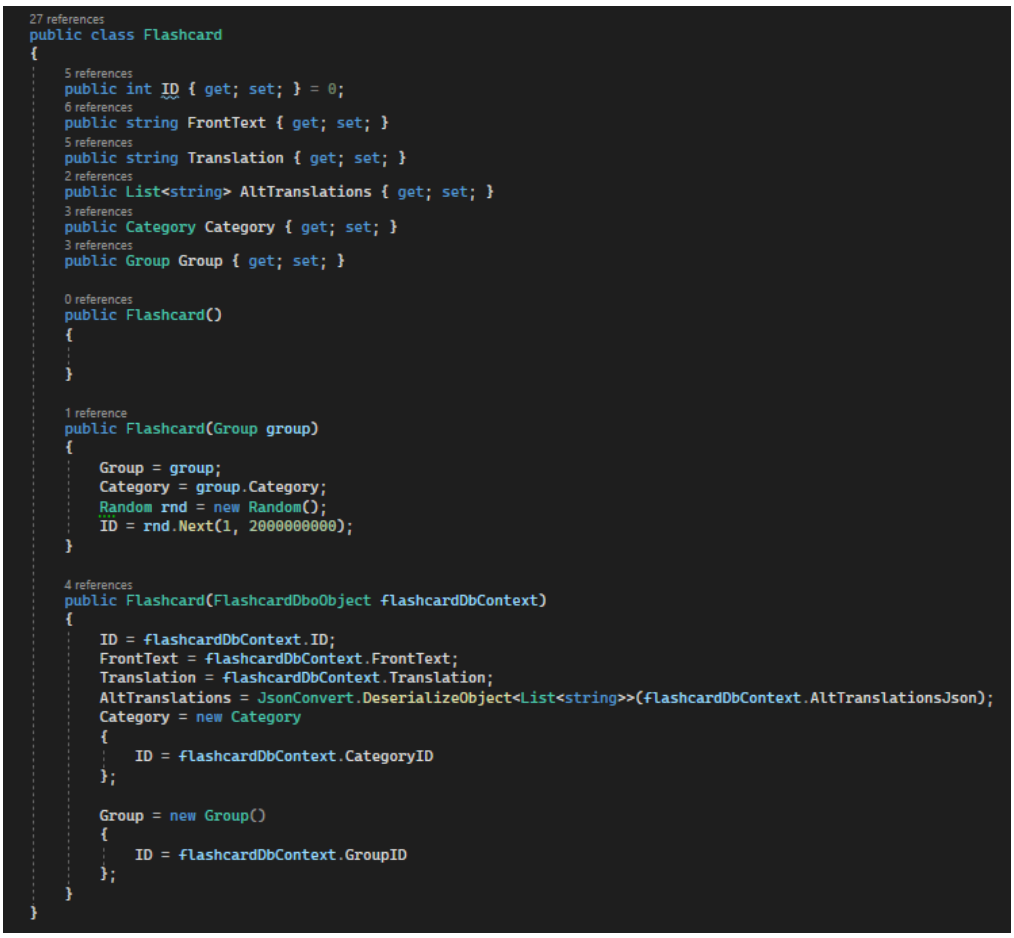


```
namespace LanguageNotes.db
{
    [Table("flashcards")]
    11 references
    public class FlashcardDboObject
    {
        [PrimaryKey, Column("_id")]
        4 references
        public int ID { get; set; }

        2 references
        public string FrontText { get; set; }

        2 references
        public string Translation { get; set; }

        2 references
        public string AltTranslationsJson { get; set; }

        3 references
        public int GroupID { get; set; }

        2 references
        public int CategoryID { get; set; }

        0 references
        public FlashcardDboObject()
        {

        }

        2 references
        public FlashcardDboObject(Flashcard card)
        {
            ID = card.ID;
            FrontText = card.FrontText;
            Translation = card.Translation;
            AltTranslationsJson = Newtonsoft.Json.JsonConvert.SerializeObject(card.AltTranslations);
            GroupID = card.Group.ID;
            CategoryID = card.Category.ID;
        }
    }

    [Table("groups")]
    10 references
    public class GroupDboObject
    {
        [PrimaryKey, Column("_id")]
        2 references
        public int ID { get; set; }

        2 references
        public string Name { get; set; }

        3 references
        public int CategoryID { get; set; }

        0 references
        public GroupDboObject()
        {

        }

        3 references
        public GroupDboObject(Group group)
        {
            ID = group.ID;
            Name = group.Name;
            CategoryID = group.Category.ID;
        }
    }

    [Table("categories")]
    11 references
    public class CategoryDboObject
    {
        [PrimaryKey, Column("_id")]
        3 references
        public int ID { get; set; }

        2 references
        public string Name { get; set; }

        0 references
        public CategoryDboObject()
        {

        }

        3 references
        public CategoryDboObject(Category category)
        {
            ID = category.ID;
            Name = category.Name;
        }
    }
}
```

Рисунок 3. – Створення сущностей для бази даних

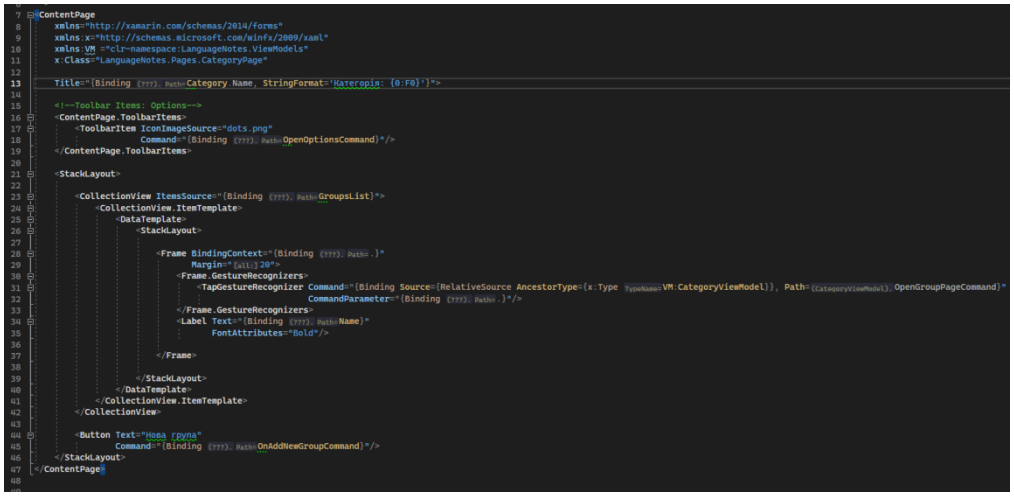
```
namespace LanguageNotes.db
{
    18 references
    public class FlashcardsDatabase
    {
        static SQLiteConnection Database;

        private static readonly FlashcardsDatabase instance = new FlashcardsDatabase();

        1 reference
        public FlashcardsDatabase()
        {
            Database = new SQLiteConnection(Constants.DatabasePath, Constants.Flags);
            var result = Database.CreateTable<FlashcardDboObject>();
            result = Database.CreateTable<GroupDboObject>();
            result = Database.CreateTable<CategoryDboObject>();
        }

        //Database Operations
        1 reference
        public static int SaveNoteCard(Flashcard card)
        {
            var noteCard = new FlashcardDboObject(card);

            bool foundIt = false;
            foreach (var flashcard in GetAllNoteCards())
            {
                if (flashcard.ID == noteCard.ID)
                {
                    foundIt = true;
                    break;
                }
            }
            return foundIt == false ? Database.Insert(noteCard) : Database.Update(noteCard);
        }

        1 reference
        public static int DeleteNoteCard(Flashcard noteCard)
        {
            var dbo = new FlashcardDboObject(noteCard);
            return Database.Delete(dbo);
        }

        1 reference
        public static IList<Flashcard> GetAllNoteCards()
        {
            var cards = new List<Flashcard>();
            var dboList = Database.Table<FlashcardDboObject>() // TableQuery<FlashcardDboObject>
                    .ToList();
            dboList.ForEach(obj :FlashcardDboObject => cards.Add( item: new Flashcard(obj)));
            return cards;
        }

        0 references
        public static Flashcard GetNoteCard(Flashcard noteCard)
        {
            return new Flashcard(Database.Table<FlashcardDboObject>()
                                         .Where(card :FlashcardDboObject => card.ID == noteCard.ID) // TableQuery<FlashcardDboObject>
                                         .First());
        }

        3 references
        public static IList<Flashcard> GetNoteCardsInGroup(Group group)
        {
            var result = new List<Flashcard>();
            var dboList = Database.Table<FlashcardDboObject>()
                                        .Where(card :FlashcardDboObject => card.GroupID == group.ID) // TableQuery<FlashcardDboObject>
                                        .ToList();
            dboList.ForEach(obj :FlashcardDboObject => result.Add( item: new Flashcard(obj)));
            return result;
        }
```

Рисунок 4. – Визначення основних методів для взаємодії з об'єктами бази даних на прикладі категорій



```
27 references
public class Flashcard
{
    5 references
    public int ID { get; set; } = 0;
    6 references
    public string FrontText { get; set; }
    5 references
    public string Translation { get; set; }
    2 references
    public List<string> AltTranslations { get; set; }
    3 references
    public Category Category { get; set; }
    3 references
    public Group Group { get; set; }

    0 references
    public Flashcard()
    {

    }

    1 reference
    public Flashcard(Group group)
    {
        Group = group;
        Category = group.Category;
        Random rnd = new Random();
        ID = rnd.Next(1, 2000000000);
    }

    4 references
    public Flashcard(FlashcardDboObject flashcardDbContext)
    {
        ID = flashcardDbContext.ID;
        FrontText = flashcardDbContext.FrontText;
        Translation = flashcardDbContext.Translation;
        AltTranslations = JsonConvert.DeserializeObject<List<string>>(flashcardDbContext.AltTranslationsJson);
        Category = new Category
        {
            ID = flashcardDbContext.CategoryID
        };

        Group = new Group()
        {
            ID = flashcardDbContext.GroupID
        };
    }
}
```

Рисунок 5. – Створення класу з відповідними властивостями та конструкторами на прикладі навчальної картки



```
<ContentPage
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:VM ="clr-namespace:LanguageNotes.ViewModels"
    x:Class="LanguageNotes.Pages.CategoryPage"

    Title="{Binding (???). Path=Category.Name, StringFormat='Категорія: {0:F0}'}">

    <!--Toolbar Items: Options-->
    <ContentPage.ToolbarItems>
        <ToolbarItem IconImageSource="dots.png"
                     Command="{Binding (???). Path=OpenOptionsCommand}"/>
    </ContentPage.ToolbarItems>

    <StackLayout>

        <CollectionView ItemsSource="{Binding (???). Path=GroupsList}">
            <CollectionView.ItemTemplate>
                <DataTemplate>
                    <StackLayout>

                        <Frame BindingContext="{Binding (???). Path=.}"
                               Margin="{all:}20">
                            <Frame.GestureRecognizers>
                                <TapGestureRecognizer Command="{Binding Source={RelativeSource AncestorType={x:Type TypeName=VM:CategoryViewModel}}, Path={CategoryViewModel}. OpenGroupPageCommand}"
                                                      CommandParameter="{Binding (???). Path=.}"/>
                            </Frame.GestureRecognizers>
                            <Label Text="{Binding (???). Path=Name}"
                                   FontAttributes="Bold"/>

                        </Frame>

                    </StackLayout>
                </DataTemplate>
            </CollectionView.ItemTemplate>
        </CollectionView>

        <Button Text="Нова група"
                Command="{Binding (???). Path=OnAddNewGroupCommand}"/>
    </StackLayout>
</ContentPage>
```

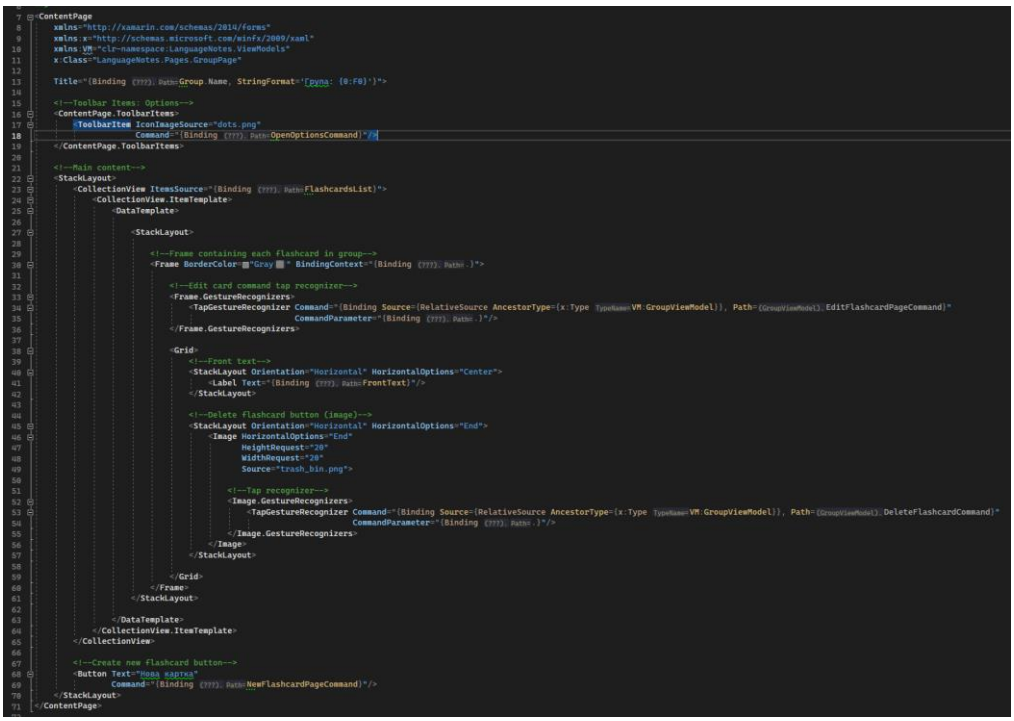
Рисунок 6. – Сторінка категорії мовою розмітки xaml

```
using Xamarin.Forms;
using LanguageNotes.Models;
using LanguageNotes.ViewModels;

namespace LanguageNotes.Pages
{
    5 references
    public partial class CategoryPage : ContentPage
    {
        CategoryViewModel viewModel;
        1 reference
        public CategoryPage(Category category)
        {
            InitializeComponent();
            viewModel = new CategoryViewModel(category);
            BindingContext = viewModel;
        }

        0 references
        protected override void OnAppearing()
        {
            base.OnAppearing();
            viewModel.OnAppearing();
        }
    }
}

```

Рисунок 7. – Підоб'єкт сторінки категорії xaml.cs

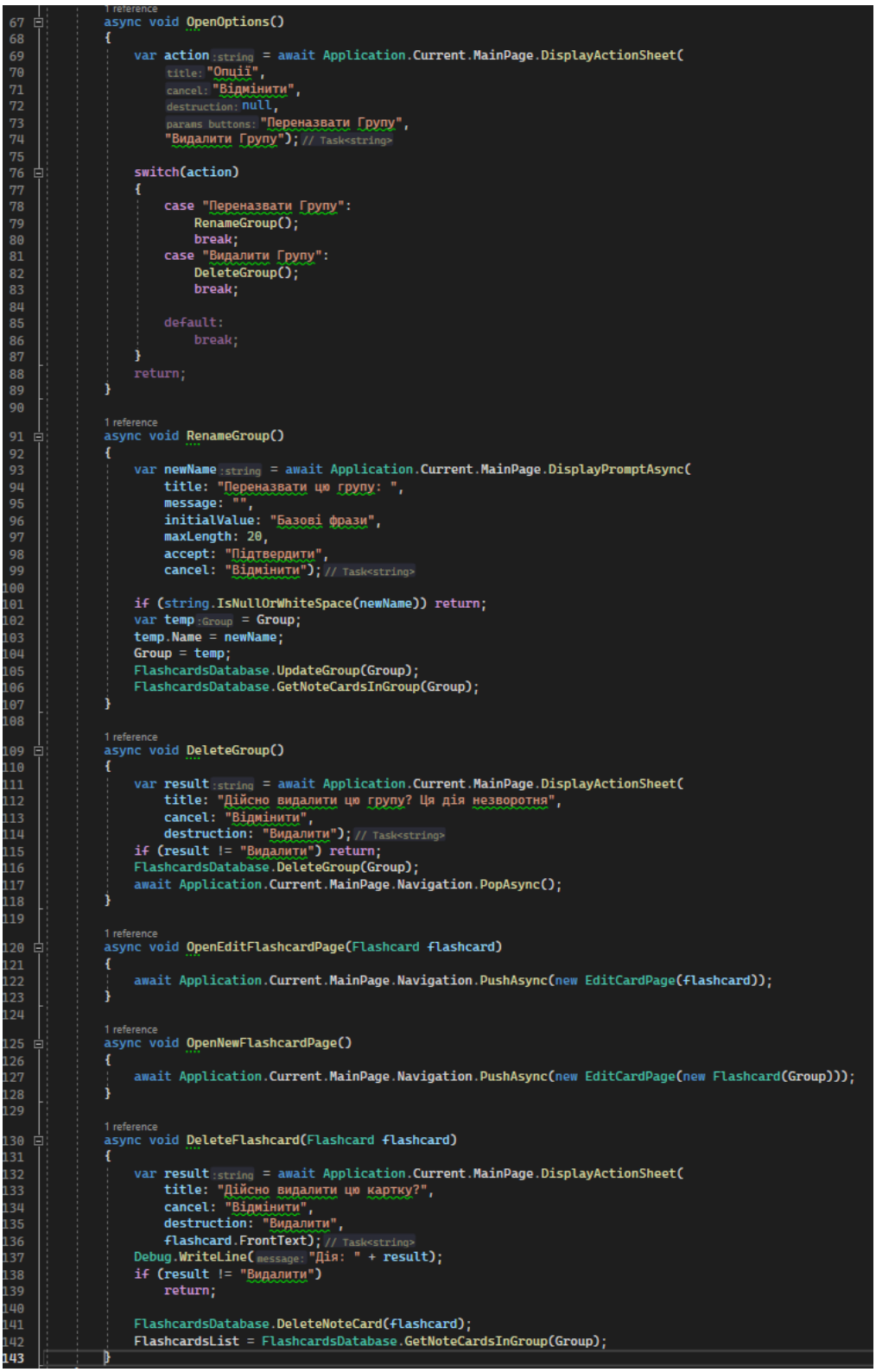
```
<ContentPage
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:VM="clr-namespace:LanguageNotes.ViewModels"
    x:Class="LanguageNotes.Pages.GroupPage"

    Title="{Binding {???}.Path=Group.Name, StringFormat='Група: {0:F0}'}">

    <!--Toolbar Items: Options-->
    <ContentPage.ToolbarItems>
        <ToolbarItem IconImageSource="dots.png"
                     Command="{Binding {???}.Path=OpenOptionsCommand}"/>
    </ContentPage.ToolbarItems>

    <!--Main content-->
    <StackLayout>
        <CollectionView ItemsSource="{Binding {???}.Path=FlashcardsList}">
            <CollectionView.ItemTemplate>
                <DataTemplate>

                    <StackLayout>

                        <!--Frame containing each flashcard in group-->
                        <Frame BorderColor="Gray " BindingContext="{Binding {???}.Path=.}">

                            <!--Edit card command tap recognizer-->
                            <Frame.GestureRecognizers>
                                <TapGestureRecognizer Command="{Binding Source={RelativeSource AncestorType={x:Type TypeName=VM:GroupViewModel}}, Path={GroupViewModel}.EditFlashcardPageCommand}"
                                                      CommandParameter="{Binding {???}.Path=.}"/>
                            </Frame.GestureRecognizers>

                            <Grid>
                                <!--Front text-->
                                <StackLayout Orientation="Horizontal" HorizontalOptions="Center">
                                    <Label Text="{Binding {???}.Path=FrontText}"/>
                                </StackLayout>

                                <!--Delete flashcard button (image)-->
                                <StackLayout Orientation="Horizontal" HorizontalOptions="End">
                                    <Image HorizontalOptions="End"
                                           HeightRequest="20"
                                           WidthRequest="20"
                                           Source="trash_bin.png">

                                        <!--Tap recognizer-->
                                        <Image.GestureRecognizers>
                                            <TapGestureRecognizer Command="{Binding Source={RelativeSource AncestorType={x:Type TypeName=VM:GroupViewModel}}, Path={GroupViewModel}.DeleteFlashcardCommand}"
                                                                  CommandParameter="{Binding {???}.Path=.}"/>
                                        </Image.GestureRecognizers>
                                    </Image>
                                </StackLayout>

                            </Grid>
                        </Frame>
                    </StackLayout>

                </DataTemplate>
            </CollectionView.ItemTemplate>
        </CollectionView>

        <!--Create new flashcard button-->
        <Button Text="Нова картка"
                Command="{Binding {???}.Path=NewFlashcardPageCommand}"/>
    </StackLayout>
</ContentPage>
```

Рисунок 8. – Сторінка редагування картки



```
<ContentPage
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:VM="clr-namespace:LanguageNotes.ViewModels"
    x:Class="LanguageNotes.Pages.GroupPage"

    Title="{Binding {???}. Path=Group.Name, StringFormat='Група: {0:F0}'}">

    <!--Toolbar Items: Options-->
    <ContentPage.ToolbarItems>
        <ToolbarItem IconImageSource="dots.png"
                     Command="{Binding {???}. Path=OpenOptionsCommand}"/>
    </ContentPage.ToolbarItems>

    <!--Main content-->
    <StackLayout>
        <CollectionView ItemsSource="{Binding {???}. Path=FlashcardsList}">
            <CollectionView.ItemTemplate>
                <DataTemplate>

                    <StackLayout>

                        <!--Frame containing each flashcard in group-->
                        <Frame BorderColor="Gray " BindingContext="{Binding {???}. Path=.}">

                            <!--Edit card command tap recognizer-->
                            <Frame.GestureRecognizers>
                                <TapGestureRecognizer Command="{Binding Source={RelativeSource AncestorType={x:Type TypeName=VM:GroupViewModel}}, Path={GroupViewModel}. EditFlashcardPageCommand}"
                                                      CommandParameter="{Binding {???}. Path=.}"/>
                            </Frame.GestureRecognizers>

                            <Grid>
                                <!--Front text-->
                                <StackLayout Orientation="Horizontal" HorizontalOptions="Center">
                                    <Label Text="{Binding {???}. Path=FrontText}"/>
                                </StackLayout>

                                <!--Delete flashcard button (image)-->
                                <StackLayout Orientation="Horizontal" HorizontalOptions="End">
                                    <Image HorizontalOptions="End"
                                           HeightRequest="20"
                                           WidthRequest="20"
                                           Source="trash_bin.png">

                                        <!--Tap recognizer-->
                                        <Image.GestureRecognizers>
                                            <TapGestureRecognizer Command="{Binding Source={RelativeSource AncestorType={x:Type TypeName=VM:GroupViewModel}}, Path={GroupViewModel}. DeleteFlashcardCommand}"
                                                                  CommandParameter="{Binding {???}. Path=.}"/>
                                        </Image.GestureRecognizers>
                                    </Image>
                                </StackLayout>

                            </Grid>
                        </Frame>
                    </StackLayout>

                </DataTemplate>
            </CollectionView.ItemTemplate>
        </CollectionView>

        <!--Create new flashcard button-->
        <Button Text="Нова картка"
                Command="{Binding {???}. Path=NewFlashcardPageCommand}"/>
    </StackLayout>
</ContentPage>
```

Рисунок 9. – Сторінка групи



```
1 reference
async void OpenOptions()
{
    var action :string = await Application.Current.MainPage.DisplayActionSheet(
        title: "Опції",
        cancel: "Відмінити",
        destruction: null,
        params buttons: "Переназвати Групу",
        "Видалити Групу"); // Task<string>

    switch(action)
    {
        case "Переназвати Групу":
            RenameGroup();
            break;
        case "Видалити Групу":
            DeleteGroup();
            break;

        default:
            break;
    }
    return;
}

1 reference
async void RenameGroup()
{
    var newName :string = await Application.Current.MainPage.DisplayPromptAsync(
        title: "Переназвати цю групу: ",
        message: "",
        initialValue: "Базові фрази",
        maxLength: 20,
        accept: "Підтвердити",
        cancel: "Відмінити"); // Task<string>

    if (string.IsNullOrWhiteSpace(newName)) return;
    var temp :Group = Group;
    temp.Name = newName;
    Group = temp;
    FlashcardsDatabase.UpdateGroup(Group);
    FlashcardsDatabase.GetNoteCardsInGroup(Group);
}

1 reference
async void DeleteGroup()
{
    var result :string = await Application.Current.MainPage.DisplayActionSheet(
        title: "Дійсно видалити цю групу? Ця дія незворотня",
        cancel: "Відмінити",
        destruction: "Видалити"); // Task<string>
    if (result != "Видалити") return;
    FlashcardsDatabase.DeleteGroup(Group);
    await Application.Current.MainPage.Navigation.PopAsync();
}

1 reference
async void OpenEditFlashcardPage(Flashcard flashcard)
{
    await Application.Current.MainPage.Navigation.PushAsync(new EditCardPage(flashcard));
}

1 reference
async void OpenNewFlashcardPage()
{
    await Application.Current.MainPage.Navigation.PushAsync(new EditCardPage(new Flashcard(Group)));
}

1 reference
async void DeleteFlashcard(Flashcard flashcard)
{
    var result :string = await Application.Current.MainPage.DisplayActionSheet(
        title: "Дійсно видалити цю картку?",
        cancel: "Відмінити",
        destruction: "Видалити",
        flashcard.FrontText); // Task<string>
    Debug.WriteLine( message: "Дія: " + result);
    if (result != "Видалити")
        return;

    FlashcardsDatabase.DeleteNoteCard(flashcard);
    FlashcardsList = FlashcardsDatabase.GetNoteCardsInGroup(Group);
}
```

Рисунок 10. – Асинхронні методи для взаємодії з об'єктами в мобільному додатку

ДОДАТОК Б

Демонстраційні матеріали

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ ВИВЧЕННЯ ПРАВИЛ ЕТИКЕТУ МОВОЮ C#

Виконав студент 4 курсу
групи ПД–43
Король Роман Євгенійович

Керівник роботи:
К.е.н, доц, доцент кафедри ІПЗ Аверічев Ігор Миколайович

Київ – 2023

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета – спрощення процесу вивчення правил етикету різних сфер життя, за рахунок використання мобільного додатку мовою C#.

Об’єкт – процес вивчення правил етикету різних сфер життя.

Предмет - мобільна платформа для вивчення правил етикету різних сфер життя.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Провести аналіз існуючих мобільних додатків з вивчення етикету.
2. Розробити мобільний додаток для вивчення правил етикету.
3. Інтегрувати в додаток бібліотеку правил етикету, що надаватиме корисну інформацію про різноманітні мовленнєві інструменти етикетної поведінки та приклади їх вживання у реальному житті для практичного навчання.
4. Виконати огляд засобів розробки програмного продукту.
5. Описати програмну реалізацію.
6. Протестувати та оцінити якості системи.
7. Описати роботу користувача з інтерфейсом.

АНАЛІЗ АНАЛОГІВ

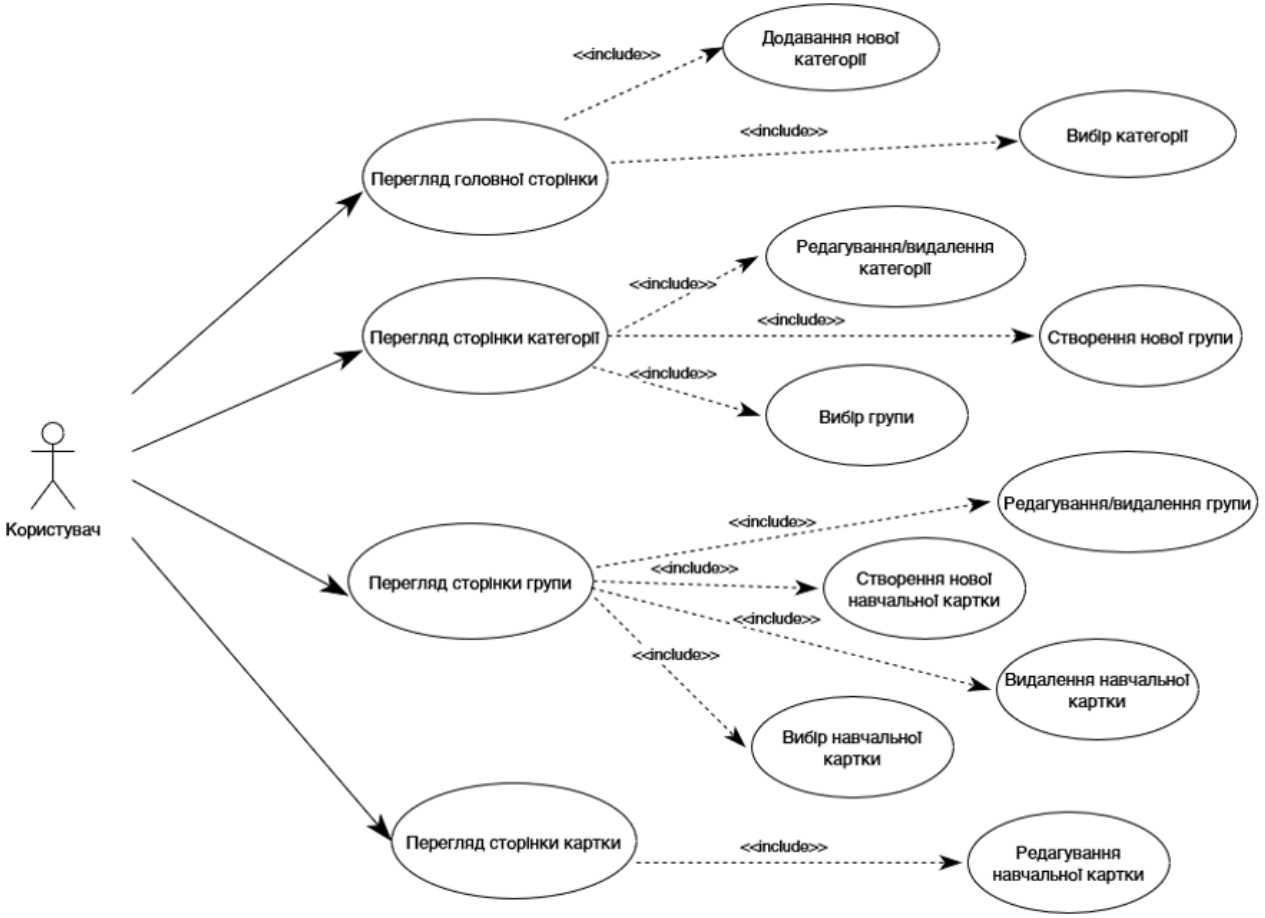
Додаток	Переваги	Недоліки
 «Etiquette»	Широке охоплення етикету: "Etiquette" включає понад 50 елементів етикету, що дозволяє користувачам ознайомитись з широким спектром соціальних норм та правил. Це забезпечує повноту та різноманітність змісту додатку.	Додаток "Etiquette" не надає можливість користувачам додавати або редагувати правила етикету. Весь контент є фіксованим і не може бути змінений або доповнений користувачами.
 «Діловий етикет»	1) Коротка книга порад: Додаток містить компактну, але інформативну книгу з практичними порадами та рекомендаціями щодо ділового етикету. Вона надає користувачам доступ до необхідної інформації без зайвого перевантаження. 2) Чіткі і лаконічні пояснення: Кожне правило ділового етикету пояснюється чітко і лаконічно, зрозуміло для будь-якого користувача. Пояснення базуються на конкретних прикладах з реального життя, що допомагає легше усвідомити, як правильно застосовувати ці правила.	1) Вузька спеціалізація: Додаток фокусується виключно на діловому етикеті і не надає інформацію про інші аспекти етикету. Це може бути обмеженням для користувачів, які також зацікавлені в інших сферах етикету. 2) Користувачі не можуть доповнювати додаток новими правилами або адаптувати його до своїх конкретних потреб чи контексту.
 Розроблений додаток «Правила етикету»	1) Додаток надає користувачам зручний та легкий доступ до широкого переліку правил етикету, що дозволяє швидко знайти потрібну інформацію та зрозуміти як застосовувати її у різних ситуаціях. 2) Широкий функціонал: додаток не обмежується відображенням правил етикету, він також надає користувачам можливість додавати нові категорії та редагувати їх, а також створювати навчальні картки зі своїми правилами етикету. Це розширює можливості додатку та робить його більш гнучким для індивідуальних потреб користувачів.	Вимога до наявності мобільного пристрою: Для використання нашого додатку користувачам потрібно мати мобільний пристрій. Це може бути обмеженням для тих, хто не має доступу до сучасних мобільних технологій.

ВИМОГИ ДО ДОДАТКУ

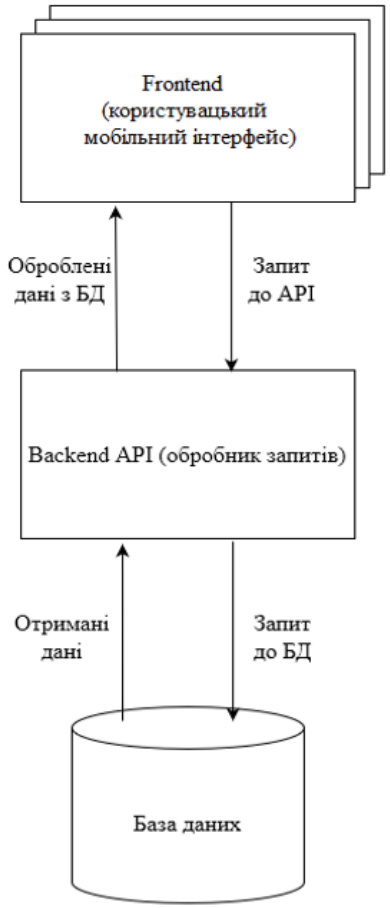
1. Забезпечення інтерактивних функцій, можливості персоналізації та практичного навчання етикету.
2. Інформаційна система має включати в себе ієрархічну бібліотеку передвстановлених правил для навчання: правила поділені на категорії вжитку (побутовий етикет, діловий, столовий тощо), групи правил (привітання, прощання тощо) та навчальні картки по кожному з правил, що містяться у групах.
3. Кожен модуль ієрархії повинен мати опції додавання, редагування та видалення, для більшої персоналізації програмного продукту.
4. Мобільний додаток повинен містити навчальний контент, опис, приклади та пояснення правил етикету для вжитку в різних сферах життя.

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

ДІАГРАМА ПРЕЦЕНДЕНТІВ



АРХІТЕКТУРА СИСТЕМИ



ДІАГРАМА КЛАСІВ

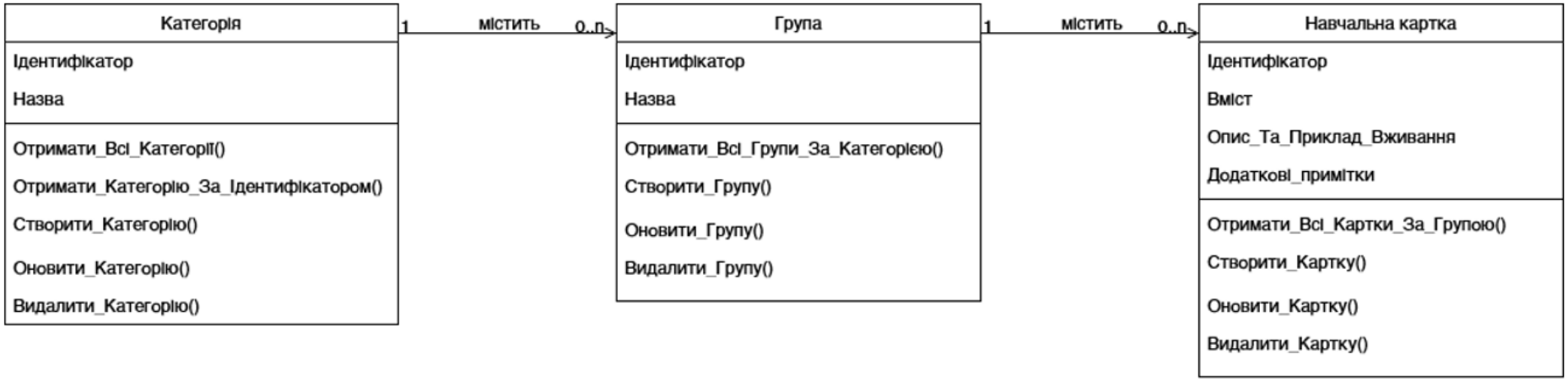
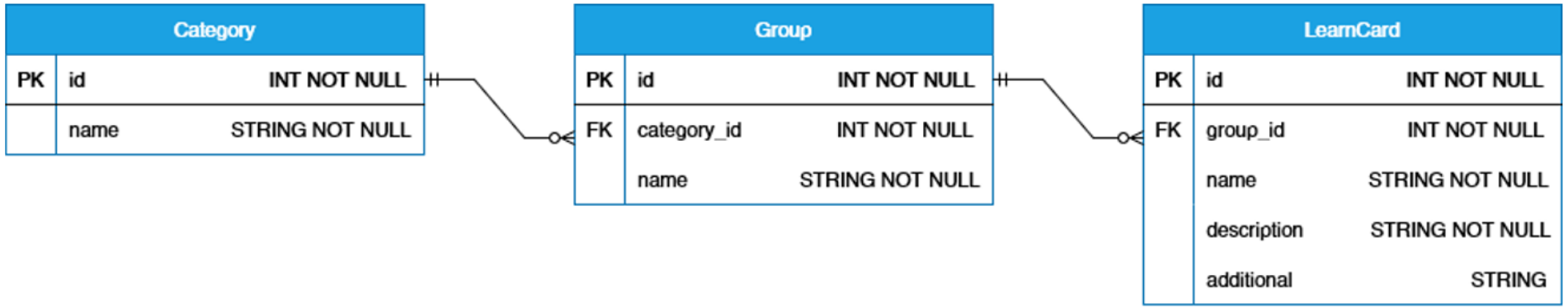
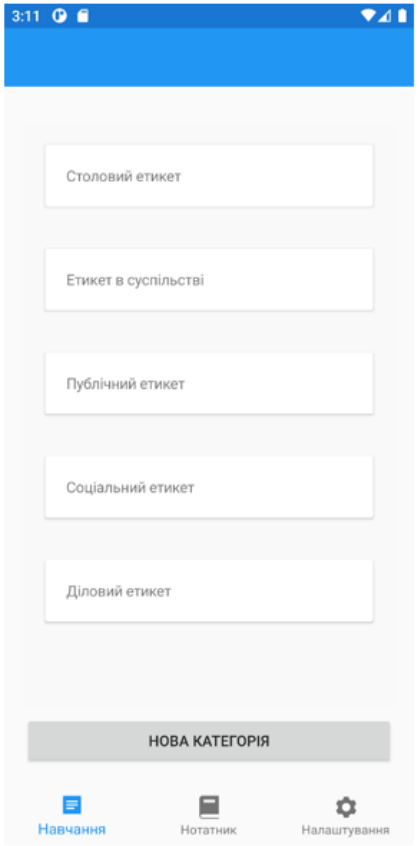


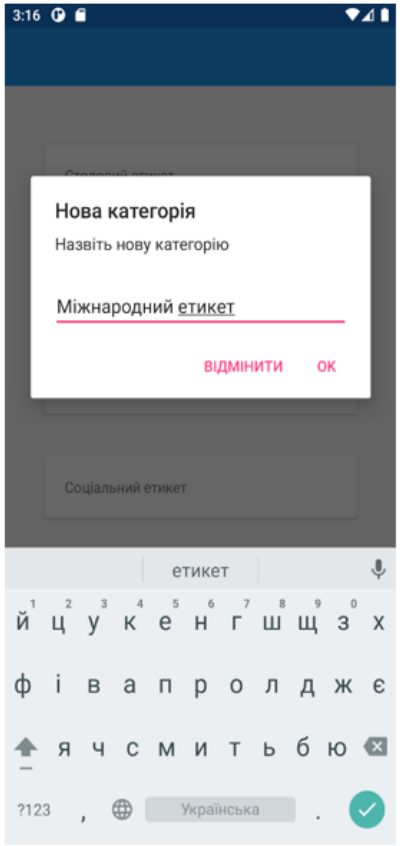
СХЕМА БАЗИ ДАНИХ



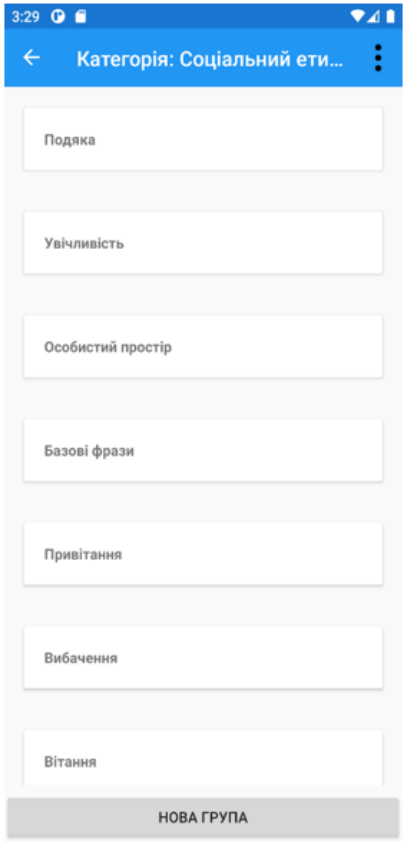
ЕКРАННІ ФОРМИ



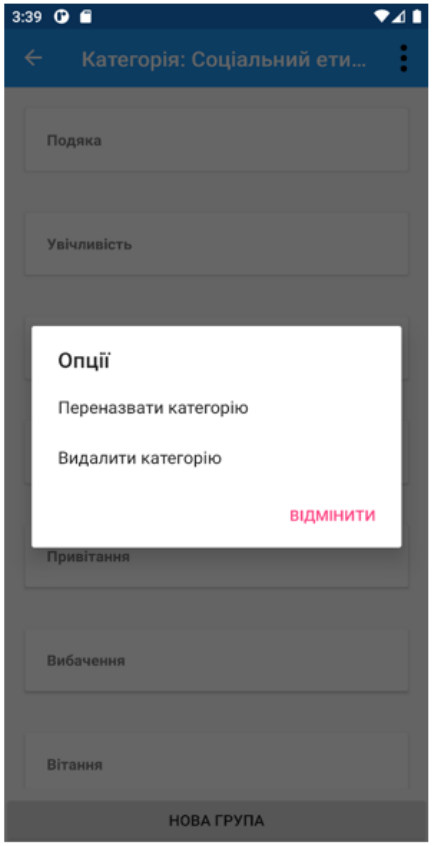
Головна сторінка



Додавання категорії

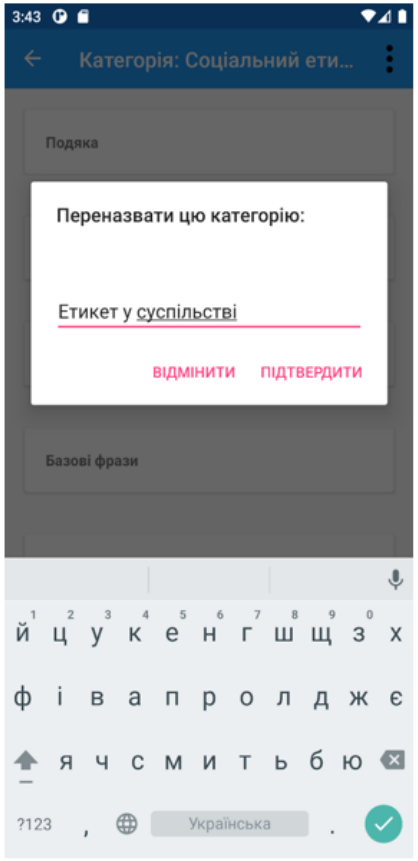


Сторінка категорії

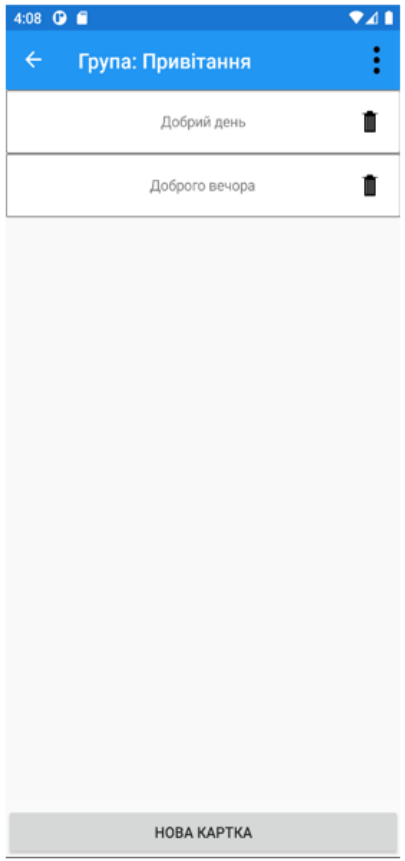


Опції категорії

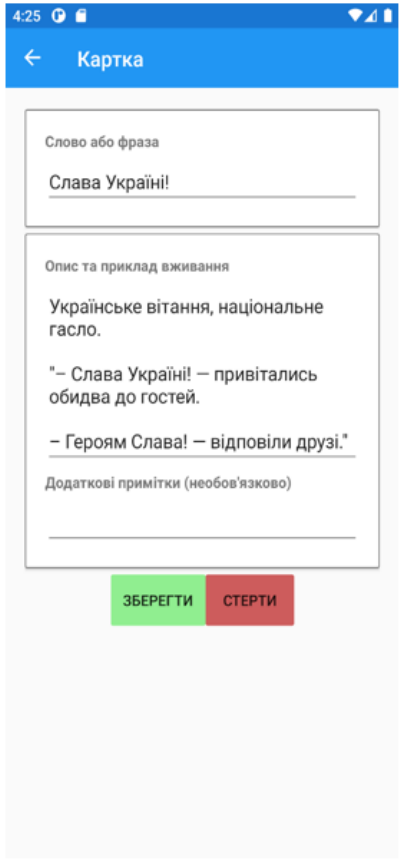
ЕКРАННІ ФОРМИ



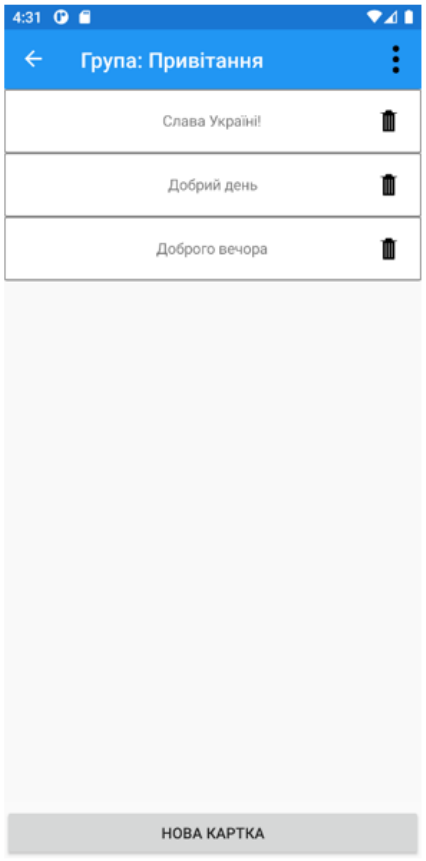
Переназвання категорії



Сторінка групи

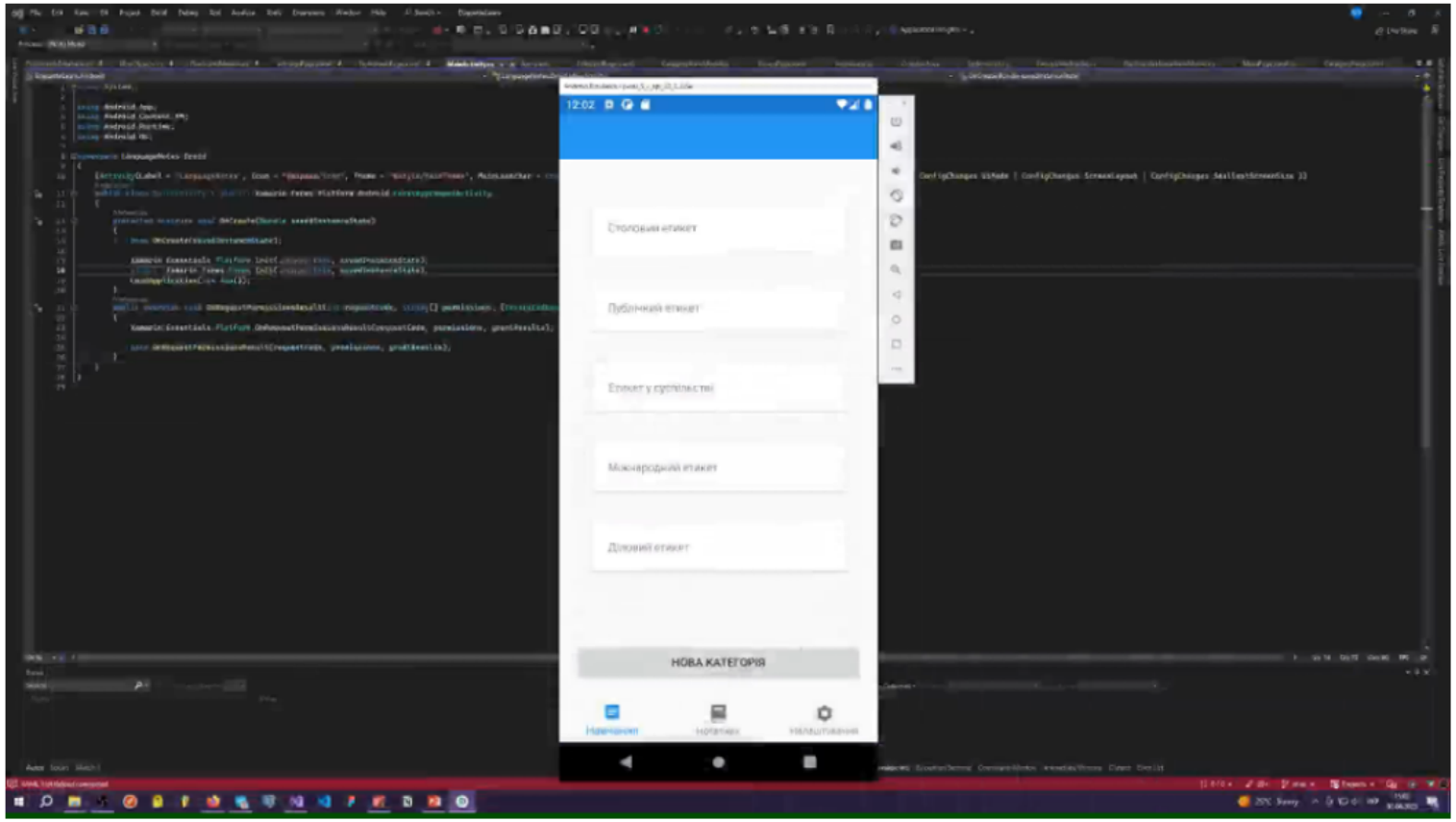


Сторінка картки



Результат додавання картки

ДЕМОНСТРАЦІЯ РОБОТИ



АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Аверічев І.М., Король Р.Є. Розробка мобільного додатку для вивчення правил етикету мовою c# // Всеукраїнська науково-технічна конференція “Застосування програмного забезпечення в інфокомунікаційних технологіях” - Київ: ДУТ, 2023. – С. 106-107;
2. Аверічев І.М., Король Р.Є. Засоби програмної реалізації мобільного додатку для вивчення правил етикету// Всеукраїнська науково-технічна конференція “Застосування програмного забезпечення в інфокомунікаційних технологіях” - Київ: ДУТ, 2023. – С. 108-109.

ВИСНОВКИ

1. Розроблено мобільний додаток для вивчення правил етикету, який забезпечує зручний і простий інтерфейс для користувачів.
2. Надано змогу користувачам вивчати правила етикету через навчальні картки, організовані ієрархічно за категоріями і групами.
3. Розроблено опції, завдяки яким користувачі можуть додавати нові категорії, редагувати і змінювати назви їх, а також видаляти навчальні картки.
4. Додано можливість заповнювати навчальні картки з необхідною інформацією про правила етикету, що дозволяє персоналізувати навчання.

ДЯКУЮ ЗА УВАГУ!