ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка застосунку для моніторингу навчального навантаження студентів. Спец-частина: Розробка Frontend-частини застосунку»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Сергій РІДКОВЕЦЬ
(підпис)

Виконав: здобувач вищої освіти групи ПД-41

Сергій РІДКОВЕЦЬ

Керівник: Ірина ЗАМРІЙ
д.т.н., професор

Рецензент: _____

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Рідковцю Сергію Вікторовичу

1. Тема кваліфікаційної роботи: «Розробка застосунку для моніторингу навчального навантаження студентів. Спец-частина: Розробка Frontend-частини застосунку»
керівник кваліфікаційної роботи д.т.н., професор Ірина ЗАМРІЙ,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56.

2. Строк подання кваліфікаційної роботи «02» червня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про навчальне навантаження студенів, аналіз наявних рішень для моніторинг навантаження студентів та недоліків при їх використанні, огляд засобів реалізації з технічною документацією для використаних фреймворків та бібліотек.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Огляд та аналіз наявних рішень для спостереження за навантаженням студентів.
2. Проєктування застосунків для моніторингу навчального навантаження студентів.

3. Програмна реалізація та опис функціонування застосунків для створення, проходження та перегляду результатів оцінювань для студентів.
4. Тестування застосунків.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Діаграма варіантів використання.
5. Мапа застосунку адміністратора.
6. Мапа застосунку студента.
7. Екранні форми.
8. Апробація результатів дослідження.

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	25.02.-04.03.2025	
2	Аналіз та дослідження існуючих аналогів	05.03-12.03.2025	
3	Огляд існуючих аналогів для моніторингу навчального навантаження студентів	13.03-19.03.2025	
4	Проєктування застосунків для моніторингу навчального навантаження студентів	20.03-01.04.2025	
5	Програмна реалізація застосунків	02.04-25.04.2025	
6	Тестування застосунку	26.04-28.04.2025	
7	Оформлення роботи: вступ, висновки, реферат	29.04-11.05.2025	
8	Розробка демонстраційних матеріалів	12.05-18.05.2025	
9	Попередній захист роботи	19.05-30.05.2025	

Здобувач вищої освіти _____ Сергій РІДКОВЕЦЬ
(підпис)

Керівник
кваліфікаційної роботи _____ Ірина ЗАМРІЙ
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 64 стор., 2 табл., 44 рис., 20 джерел.

Мета роботи – спрощення моніторингу навчального навантаження студентів на основі використання програмного забезпечення для вирішення проблеми збору даних про завантаженість студентів.

Об'єкт дослідження – процес збору даних та моніторингу навчального навантаження студентів у закладах вищої освіти.

Предмет дослідження – програмне забезпечення для оцінювання та моніторингу навчального навантаження студентів.

Короткий зміст роботи: В роботі було проаналізовано проблему навчального навантаження студентів для знаходження її рішення. Оглянуто та проаналізовано наявні рішення цієї проблеми для знаходження недоліків: Google Forms, SurveyMonkey, Moodle, Canvas. Розроблено клієнтську частину на двох застосунках та було реалізовано ключові функціональні можливості, зокрема: імпортування інформації з силабусу, створення дисциплін та оцінювань, перехід за посиланням до групи оцінювання та одного оцінювання, проходження оцінювання та відправлення результатів, перегляд результатів та їх візуалізація за допомогою графіків. Проведено тестування застосунків за допомогою створених тестових сценаріїв. В роботі використано фреймворк .NET MAUI для створення застосунків, Refit для взаємодії з сервером, LiveCharts для візуалізації результатів.

Сферою використання застосунку є моніторинг навчального навантаження студентів через проходження оцінювань на витрачений час.

КЛЮЧОВІ СЛОВА: НАВЧАЛЬНЕ НАВАНТАЖЕННЯ, ЗАСТОСУНОК АДМІНІСТРАТОРІВ, ЗАСТОСУНОК СТУДЕНТІВ, СИЛАБУС, MVVM, .NET MAUI.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AI – Artificial Intelligence

API – Application Programming Interface

CLI – Command Line Interface

HTTP – Hypertext Transfer Protocol

IDE – Integrated Development Environment

LINQ – Language Integrated Query

LMS – Learning Management System

MAUI – Multi-platform App UI

MVVM – Model-View-ViewModel

REST – Representation State Transfer

SQL – Structured Query Language

UI – User Interface

XAML – Extensible Application Markup Language

ВСТУП

Останнім часом все більше студентів стикаються з проблемами під час навчання у вищих навчальних закладах. Основною проблемою є перенавантаження студентів по кількості та обсягу навчального матеріалу, що впливає на ефективне засвоєння матеріалу студентами та на їх психічний стан. Заклади освіти звісно проводять певні опитування, але переважно вони не націлені на виправлення проблем з навчальним навантаженням студентів. Через це зростає потреба в наявності цифрових інструментів, які будуть не лише вирішувати проблему з перевантаженням студентів, а ще порівнювати реальний витрачений час на завдання з зазначеним часом в силабусі.

Мета роботи – спрощення моніторингу навчального навантаження студентів на основі використання програмного забезпечення для вирішення проблеми збору даних про завантаженість студентів.

Об'єкт дослідження – процес збору даних та моніторингу навчального навантаження студентів у закладах вищої освіти.

Предмет дослідження – програмне забезпечення для оцінювання та моніторингу навчального навантаження студентів.

Для реалізації поставленої мети виділено такі задачі:

1. Проаналізувати наявні рішення для моніторингу навчального навантаження студентів.
2. Визначити функціональні та нефункціональні вимоги до застосунків.
3. Дослідити засоби реалізації, які можуть бути використані при розробці застосунків.
4. Розробити застосунок студентів для проходження оцінювання з можливістю вписувати витрачений час на дисципліну та відправляти результати.
5. Розробити застосунок адміністраторів для створення та перегляду результатів оцінювань з можливістю імпорту даних з документу про силабус.
6. Провести тестування розроблених застосунків.

Для розробки клієнтської частини застосунку було використано основний фреймворк .NET MAUI, що дозволяє створювати застосунки з єдиною кодовою базою. Мовою програмування виступає C#, яка забезпечує високу продуктивність та підтримує об’єктно-орієнтовану парадигму.

Розробка велася в середовищі JetBrains Rider, що добре підходить для роботи з подібними проєктами. При створенні кожного з застосунків, використовувався шаблон MVVM, що дозволяє розроблювати проєкт в команді на клієнтській та серверній частині відповідно. Також завдяки MVVM вдалося реалізувати інтерфейс для студентів і адміністраторів, з підтримкою двостороннього зв’язку між View та ViewModel.

В результаті було створено два застосунки, один з яких забезпечує студентам можливість фіксувати витрачений на дисципліни час та проходити оцінювання, а інший забезпечує адміністраторам можливість переглядати результати та здійснювати імпорт навчальних даних із силабусу.

Застосунки спростять моніторинг навчального навантаження та допоможуть вищим закладам приймати ефективні рішення при планування навчання в майбутньому.

1 АНАЛІЗ ПРОБЛЕМИ ТА ІСНУЮЧИХ ЗАСОБІВ ДЛЯ ЇЇ ВИРІШЕННЯ

1.1 Опис проблеми навчального навантаження студентів

У сучасному освітньому процесі важливу роль відіграє збалансоване розподілення навчального навантаження студентів. Надмірне навантаження може призвести до зниження якості засвоєння матеріалу, стресу, втрати мотивації, а недостатнє – до недоотримання необхідних знань і навичок. Водночас адміністрація вищих навчальних закладів має дотримуватись стандартів визначених силабусом, що є документом взаємовідносин викладачів та студентів [1, с. 72].

Задача моніторингу навчального навантаження полягає у зборі, обробці та аналізі інформації про кількість годин, які студенти реально витрачають на виконання завдань, передбачених навчальними дисциплінами. Традиційно такий моніторинг або не здійснюється взагалі, або ж виконується у формі звітності, яка не дозволяє отримати точні та динамічні дані. Крім того, відсутність інтерактивного інструменту ускладнює зворотній зв’язок між студентами та адміністраторами.

Особливістю даної задачі є суб’єктивність оцінки з боку студентів, різноманітність завдань за складністю та типом, а також потреба в адаптації процесу під конкретні навчальні програми. Складність полягає також у необхідності співставлення очікуваного навантаження, яке зазначається у силабусі, з фактичним – яке вказує студент. Таким чином, важливою стає можливість візуального порівняння теоретичних та реальних показників.

Загалом задача має міждисциплінарний характер, поєднуючи освітні, організаційні та технічні аспекти. Її вирішення можливе кількома способами:

- анкетування – стандартний підхід, при якому застосовуються анкети, де студенти заповнюють інформацію після завершення курсу, або під час його проходження [2, с. 186];

- використання електронних таблиць – цей спосіб автоматизовує процес збору даних, але не дозволяє обробляти та аналізувати їх;
- розробка програмного забезпечення – найбільш сучасний підхід, який дозволяє автоматизувати всі етапи збору та візуалізації даних про навчальне навантаження.

У рамках даної бакалаврської роботи було обрано третій підхід. Це розробка мобільного застосунку, який надає змогу створювати та проходити оцінювання. Основою аудиторією є:

- студенти, які отримують посилання на конкретне оцінювання або групу оцінювань, створені адміністратором на основі даних силабусу, і зазначають фактичну кількість годин, витрачених на виконання завдань;
- адміністратори, які формують оцінювання відповідно до силабусу, встановлюють очікуване навантаження (у годинах), надсилають студентам посилання та переглядають зібрані результати.

Основні вимоги цільової аудиторії до застосунку:

- мобільність та доступність – підтримка Android, адаптивний дизайн;
- інтерактивність – можливість вводити дані про кількість витраченого часу та створювати оцінювання;
- аналітика та звітність – побудова графіків з результатами та звітність з кількістю студентів що пройшли оцінювання.

Такий застосунок надаватиме адміністрації університетам інструмент, який допоможе з рівномірним навантаженням та зберегти стабільний емоційний та психічний стан студентів, який і так погіршується через війну в країні [3]. Існування цього застосунку зможе почати вирішувати проблеми в багатьох університетах з академічним навантаженням студентів. Бо результати показують, що при надмірному навантажені у студентів з’являються проблеми з подоланням негативних емоцій, тому університетам важливо приймати якісь дії для вирішення цієї проблеми [4, с. 418].

1.2 Аналіз наявних засобів для моніторингу навчального навантаження студентів

Одним з найпоширеніших способів моніторингу навчального навантаження студентів – це створення та розповсюдження опитувань за допомогою онлайн-платформ, такими як:

- Google Forms
- SurveyMonkey

Також деякі системи управління навчанням надають певні інструменти для відстежування рівня залученості студентів, що також може бути індикатором їхнього навчального навантаження. Одними з таких є:

- Moodle (Course Dedication)
- Canvas (Student Access Data)

1.2.1 Google Forms

Google Forms – це безкоштовний онлайн-інструмент від Google, який дозволяє створювати запитання різних типів та впорядковувати їх, а також надає змогу швидко налаштовувати форму відповідно до власних потреб та надсилати посилання на неї [5]. Основні можливості Google Forms включають:

- створення різних типів запитань, представлених на рисунку 1.1;
- автоматичне збереження результатів у Google Таблицях для подальшої обробки;
- гнучкі налаштування логіки опитування;
- обмеження доступу.

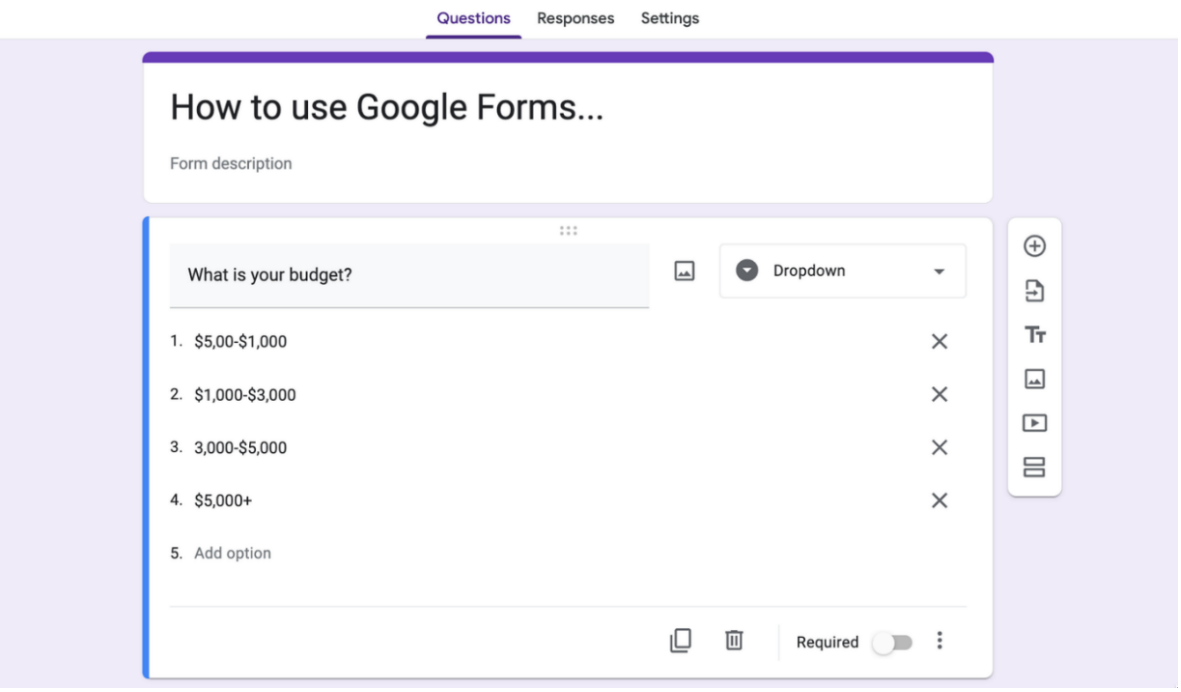


Рис. 1.1 Інтерфейс Google Forms

Google Forms є простим та доступним інструментом для збору інформації про навчальне навантаження студентів. Завдяки різноманітним типам запитань викладачі можуть отримувати зворотний зв'язок щодо обсягу завдань, рівня стресу та часу витраченого на навчання. Також вони можуть перегляди результати на графіках та ділитися посиланням на проходження опитувань.

Однак, Google Forms надає лише базові можливості автоматизації, тобто не підтримує імпорт файлів для створення опитувань. Тому, якщо виникає потреба здійснювати моніторинг по кожній окремій дисципліні, опитування доводиться створювати вручну для кожної з них. Це може бути доволі трудомістким процесом, особливо за умови великої кількості дисциплін і навчальних груп.

1.2.2 SurveyMonkey

SurveyMonkey – онлайн-платформа для створення опитувань, що дозволяє швидко та зручно ставити запитання у правильний спосіб та отримувати оперативні дані [6]. Основне призначення веб-сервісу – спрощення процесу

збору, аналізу та інтерпретації даних за допомогою створення анкет, опитувань та форм зворотного зв'язку, представлених на рисунку 1.2.

Платформа доступна для використання у веб-браузерах, а також як мобільний додаток для операційних систем Android та IOS. Вона також надає розширені можливості для бізнесу, освіти, досліджень ринку тощо. Можливості платформи наступні:

- створювати опитування із використанням шаблонів або повністю індивідуальних форм;
- використовувати різні типів запитань;
- автоматизувати аналіз відповідей із можливістю побудови графіків і таблиць;
- експортувати результати у формати CSV, XLS, PDF, та багато інших;
- проводити інтеграцію зі сторонніми сервісами, такими як Microsoft Teams, Mailchim та Salesforce;
- створювати форми опитувань за допомогою штучного інтелекту.

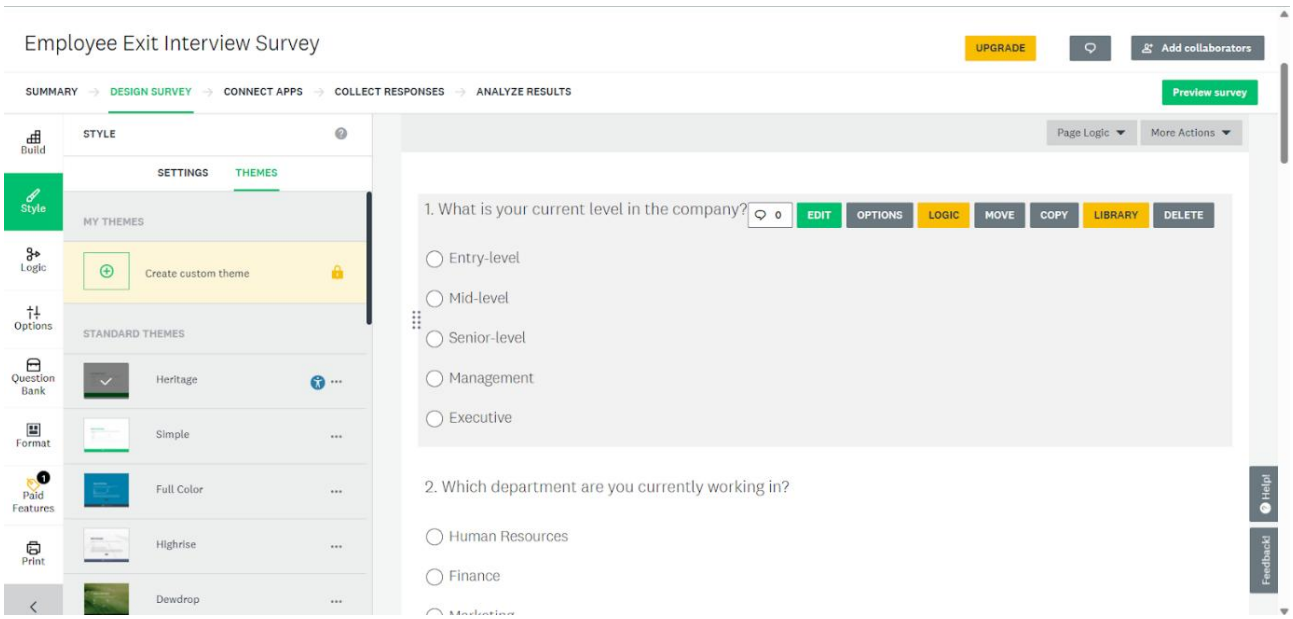


Рис. 1.2 Інтерфейс SurveyMonkey

SurveyMonkey є досить автоматизованим інструментом та може демонструвати результати на графіках, а створення запитань для опитувань можна суттєво спростити за допомогою AI.

Однак він не націлений конкретно на оцінювання навантаження студентів, через що йому бракую бажаного функціоналу як імпорт даних з силабусу. Відсутній прямий зв'язок між навчальною програмою дисципліни та форматом запитань, що обмежує можливості повної автоматизації їх генерації.

1.2.3 Moodle (Course Dedication)

Moodle – це одна з найпопулярніших гнучких та безпечних систем управління навчанням LMS яка налаштовується для будь-якої навчальної ініціативи, що дозволяє створювати навчальну платформу відповідно до потреб користувача [7]. Moodle дозволяє:

- організовувати онлайн-курси, зображених на рисунку 1.3;
- публікувати навчальні матеріали;
- контролювати навчальний прогрес;
- проводити тести, форуми, завантаження робіт.

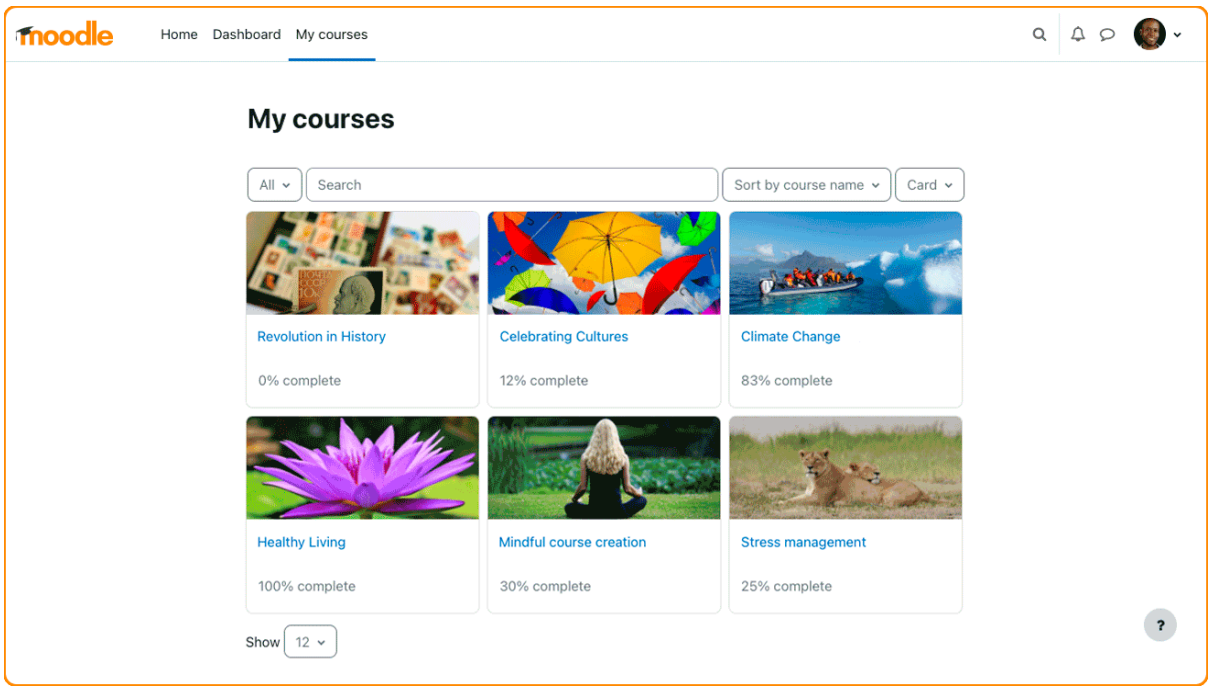


Рис. 1.3 Інтерфейс Moodle

Course Dedication – це плагін для Moodle, призначений для моніторингу та оцінки часу, який студенти реально проводять у курсі. Він надає викладачам та адміністраторам уявлення про рівень наявності студентів в онлайн-навчальному середовищі. Основні можливості плагіну включають:

- визначення загального часу перебування студентів у курсі, що можна переглянути на рисунку 1.4;
- відображення детальної статистики для кожного окремого студента;
- аналіз щоденної активності користувачів.

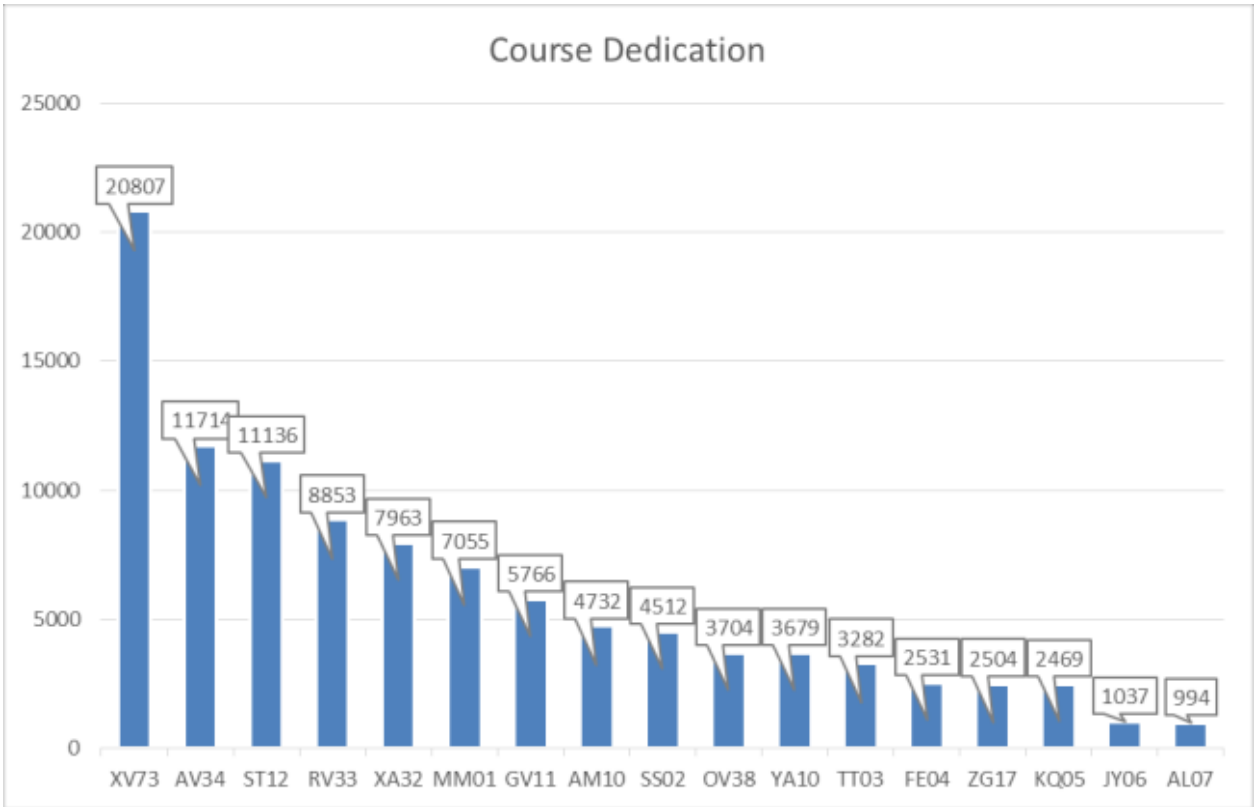


Рис. 1.4 Приклад даних з студентами завдяки Course Dedication

Використання Moodle разом із плагіном Course Dedication дає змогу здійснювати моніторинг навчального навантаження студентів у дистанційному та змішаному форматах навчання. Завдяки автоматичному збору даних про активність студентів у курсах, викладачі можуть отримувати інформацію про тривалість часу, який кожен студент приділяє опрацюванню навчальних матеріалів. Це допомагає виявляти як випадки низької активності, так і можливого перевантаження окремих студентів.

Водночас слід враховувати, що дані про залученість студентів у курсі не можна розглядати як об’єктивну оцінку навчального навантаження і не варто використовувати їх для визначення загального обсягу навантаження курсу. Можливості імпортувати дані з силабсу немає.

1.2.4 Canvas (Student Access Data)

Canvas – це сучасна система управління навчанням, що забезпечує якісне навчання та викладання для студентів та викладачів відповідно [8]. Розроблена компанією Instructure у 2011 році, вона позиціонується як інноваційне рішення для організації навчального процесу, зображеного на рисунку 1.5.

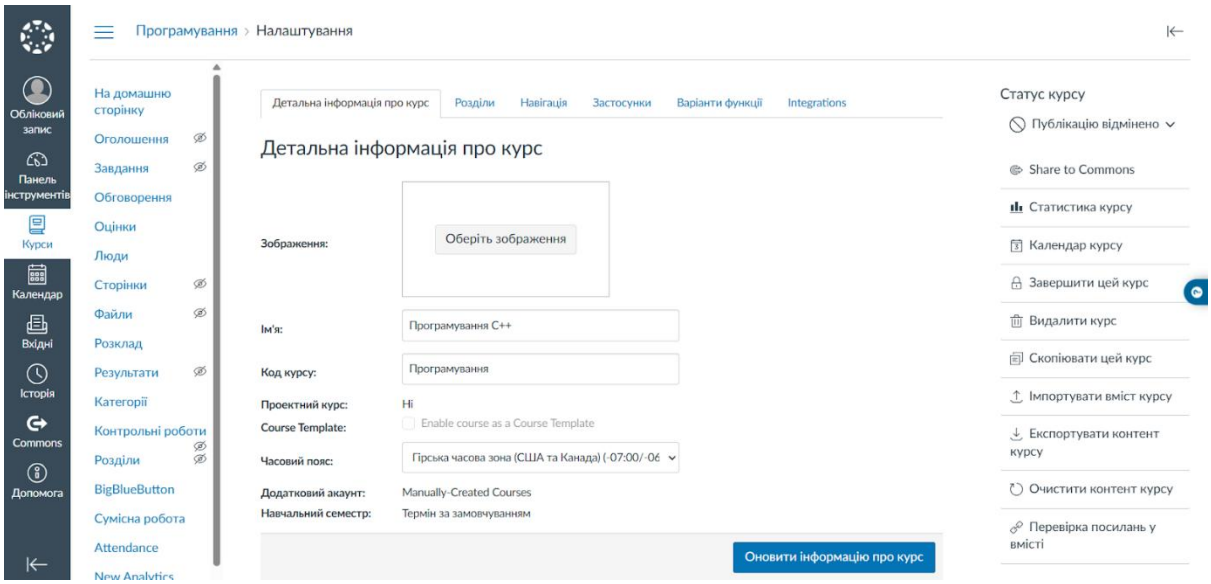


Рис. 1.5 Інтерфейс Canvas

Student Access Data – це інструмент Canvas, який надає викладачам і адміністраторам можливість відстежувати активність студентів на курсах та доступ студентів до навчальних матеріалів. Основні можливості цього інструменту включають:

- фіксацію часу, який студент витратив на платформі;
- логування дій студента, зокрема перегляд матеріалів, завантаження завдань та його участь на форумах;

- детальні звіти щодо кожного студента, що дозволяє оцінити рівень його залученості до навчального процесу.

Завдяки цьому інструменту, викладачі можуть отримати корисну інформацію про участь студентів на курсі, що допомагає коригувати підхід до навчання.

Однак він також не призначений для спостереженням за навантаженням студента, та результати не завжди точно відображають фактичне навчальне навантаження студента, та і можливості автоматично порівнювати це навантаження з теоретичним навантаженням в силабусі немає.

1.2.5 Зведені дані порівняння аналогів програмного забезпечення

Зведені результати аналізу наявних засобів для моніторингу навчального навантаження студентів наведено у таблиці 1.1.

Таблиця 1.1

Порівняння існуючих засобів-аналогів

Показник	**Google Forms**	**SurveyMonkey**	**Moodle (Course Dedication)**	**Canvas (Student Access Data)**	**TaskMon**
Форма збору даних про навантаження	Опитування студентів	Опитування студентів	На основі взаємодії з курсом	На основі взаємодії з курсом	Оцінювання студентів
Порівняння теоретичного навантаження з реальним	-	-	-	-	+
Візуалізація аналітики	Графіки, діаграми	Графіки, діаграми	Таблиця	Графіки, діаграми	Графіки, діаграми
Генерація посилань на оцінювання	+	+	-	-	+
Імпорт даних з силабусу	-	-	-	-	+

2 ПРОЄКТУВАННЯ ЗАСТОСУНКІВ

2.1 Функціональні та нефункціональні вимоги

Після порівняння аналогів було чітко зрозуміло, що прямих конкурентів які б вирішували основну проблему з навантаженням досі немає. Усі аналоги націлені на створення опитувань на різну тематику та не концентруються на основній проблемі, через що їм бракує функціоналу який би сильно допоміг при оцінені навантаження студентів. Тому після знаходження недоліків в наявних програмних забезпечень, було сформульовано вимоги, яким мають відповідати застосунки.

Функціональні вимоги:

- внесення даних про програму навчальної дисципліни. Застосунок адміністраторів повинен дозволяти адміністратору вводити інформацію про дисципліну та силабус, включаючи модулі, теми, завдання та теоретичну кількість витраченого часу на кожну з дисциплін;
- проведення оцінювання витраченого часу на вивчення дисципліни та окремих частин предмета студентами. Застосунок студентів надавати змогу вносити оцінку часу, який студенти витратили на вивчення конкретних тем або модулів, що дозволяє аналізувати їх реальне навчальне навантаження;
- створення оцінювань як для однієї дисципліни, так і для групи з дисциплінами. Застосунок адміністраторів має підтримувати створення оцінювань, які охоплюють як окрему дисципліну, так і групу дисциплін, що забезпечує гнучке оцінювання навчального процесу;
- порівняння очікуваного та реального навантаження студентів. В застосунку адміністраторів повинна бути реалізована функціональність виводу різниці між запланованим навчальним навантаженням згідно силабусу та фактично витраченим часом студентів, з можливістю візуального відображення цієї інформації за допомогою графіків;

- вивід інформації про кількість перенавантажених та недовантажених студентів. Застосунок адміністраторів повинен формувати діаграму про студентів, які витратили багато/мало часу, ніж було передбачено силабусом, що допомагає виявити незбалансованість у навантаженні;
- імпортування даних з документу про силабус. В застосунку адміністраторів має бути реалізована можливість автоматичного імпорту ключової інформації з документа силабусу, включаючи: модулі, теми, завдання та теоретичну кількість витрачених годин;
- створення посилань для проходження оцінювань. Застосунок адміністраторів повинен створювати унікальні URL-посилання на форму оцінювання, які можна надсилати студентам для збору даних про навчальне навантаження, де інший застосунок реагує на них.

Нефункціональні вимоги:

- завантаження сторінок не повинно перевищувати 3 секунди. Обидва застосунка повинні забезпечувати швидке завантаження даних з серверної частини;
- інтерфейс користувача має відповідати принципам Material Deign 3. Усі компоненти інтерфейсу повинні бути реалізовані згідно цього плагіну для Figma.

2.2 Діаграма варіантів використання

Далі було створено діаграму використання, яка відображає взаємодію основних користувачів системи з її функціональністю. Так як буде розроблюватися обидва застосунки для адміністратора та студента відповідно, то буде всього дві ролі (студент, адміністратор). Ця діаграма є важливою для розуміння загальної логіки роботи застосунків, що допомагає чітко визначити межі відповідальності кожного типу користувача та забезпечує візуалізацію функціональних вимог, що можна побачити на рисунку 2.1.

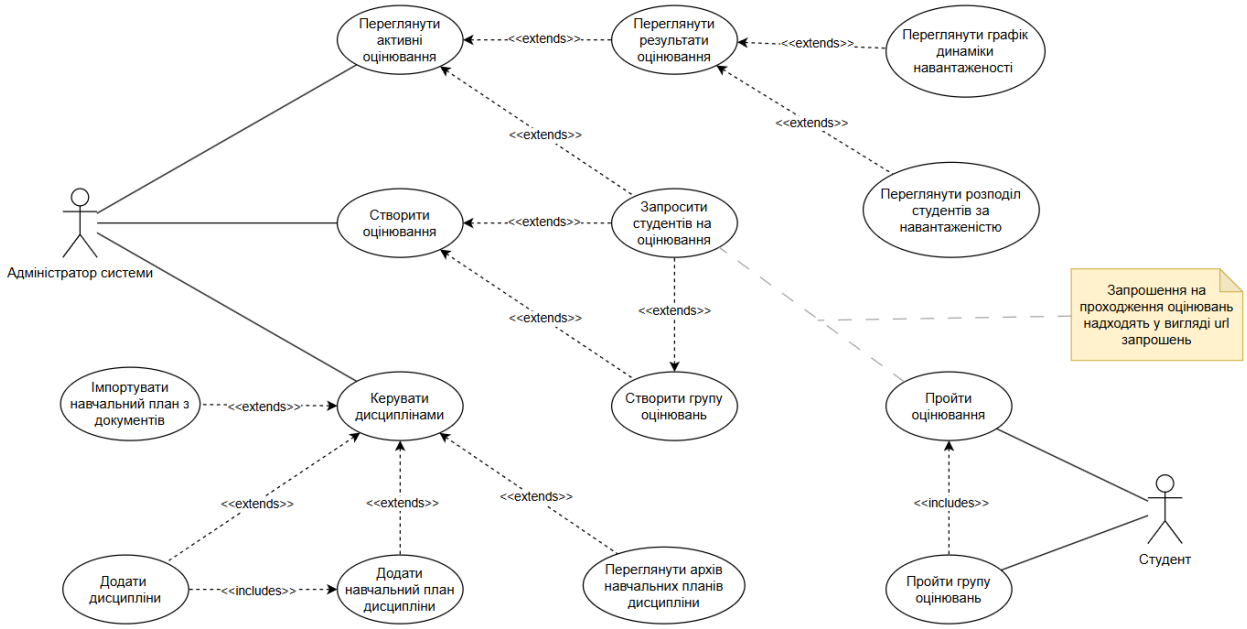


Рис. 2.1 Діаграма варіантів використання TaskMon

На діаграмі видно двох основних акторів, для кожного з яких розроблювався окремий застосунок. Розглянемо усю функціональність з якою взаємодіють актори:

- пройти групу оцінювань (студент): Дає змогу студенту обирати з списку одне з оцінювань на проходження. Після чого відображає статус проходження оцінювання, щоб студент міг побачити чи проходив він якесь з оцінювань чи ні. Також можна змінити групу оцінювань на іншу, перейшовши за новим посиланням з іншою групою опитувань;
- пройти оцінювання (студент): Дозволяє студентам проходити оцінювання, де вони можуть ввести кількість витраченого часу на кожне завдання та відправити ці результати, щоб адміністратори мали змогу їх переглядати. Якщо студенту потрібно пройти одне єдине опитування, то він може перейти за посиланням на нього;
- керувати дисциплінами (адміністратор): Адміністратор має змогу створювати, редагувати та видаляти дисципліни, які потрібні йому для створення оцінювань. Він може додавати навчальний план для кожної дисципліни вручну, заповнюючи поля з модулями, темами та завданнями, де він також може

керувати їх кількістю за потреби. Якщо навчальний план для дисципліни змінився, то він може для неї створити його знову, після чого старий силабус (навчальний план) архівується, щоб можна було подивитися в дисципліні архів із старих силабусів та новий активний силабус. Заповнювати дані вручну може бути тяжко, тому адміністратор може також імпортувати силабус, обравши PDF файл з силабусу та імпортувавши його, після чого статус файлу покаже йому коли можна його відкрити для перегляду усіх даних з силабусу;

- створити оцінювання (адміністратор): Можливість створювати оцінювання для його проходження студентами в майбутньому. Адміністратору просто потрібно обрати дисципліну по якій він хоче перевірити навантаження студентів та створити оцінювання, після чого він може копіювати посилання на його проходження та скинути його студентам;
- створити групу оцінювань (адміністратор): Дозволяє створювати не одне оцінювання, а цілу групу з ними. Адміністратор може вирішувати по скільком дисциплінам він хоче провести оцінювання та додати потрібну йому кількість, створивши групу оцінювань. Це також дає йому можливість отримати посилання виключно для групи з оцінюваннями та надіслати його студентам;
- переглянути активні оцінювання (адміністратор): Надає можливість адміністратору переглядати оцінювання в режимі моніторингу, щоб дізнатися кількість студентів які пройшли оцінювання, або ще проходять. Якщо він переглядає групу активних оцінювань, то він також може бачити кількість дисциплін що там знаходяться;
- переглянути результати оцінювання (адміністратор): Дозволяє отримати результати проходження оцінювань, де можна переглядати дані на графіках та діаграмах. Графік демонструє порівняння фактичної кількості годин та теоретичної з силабусу для кожної дисципліни, а кругова діаграма демонструє кількість студентів які справилися, або не справилися з навантаженням.

2.3 Мапа застосунку адміністратора

Далі було створено мапу застосунку адміністратора. Вона відображає навігацію між сторінками, де можна побачити кількість сторінок та що в кожній з них відбувається. Ця мапа є важливою для розуміння кожної сторінки, що допомагає зрозуміти, як орієнтуватися в застосунку. Її можна побачити на рисунку 2.2.

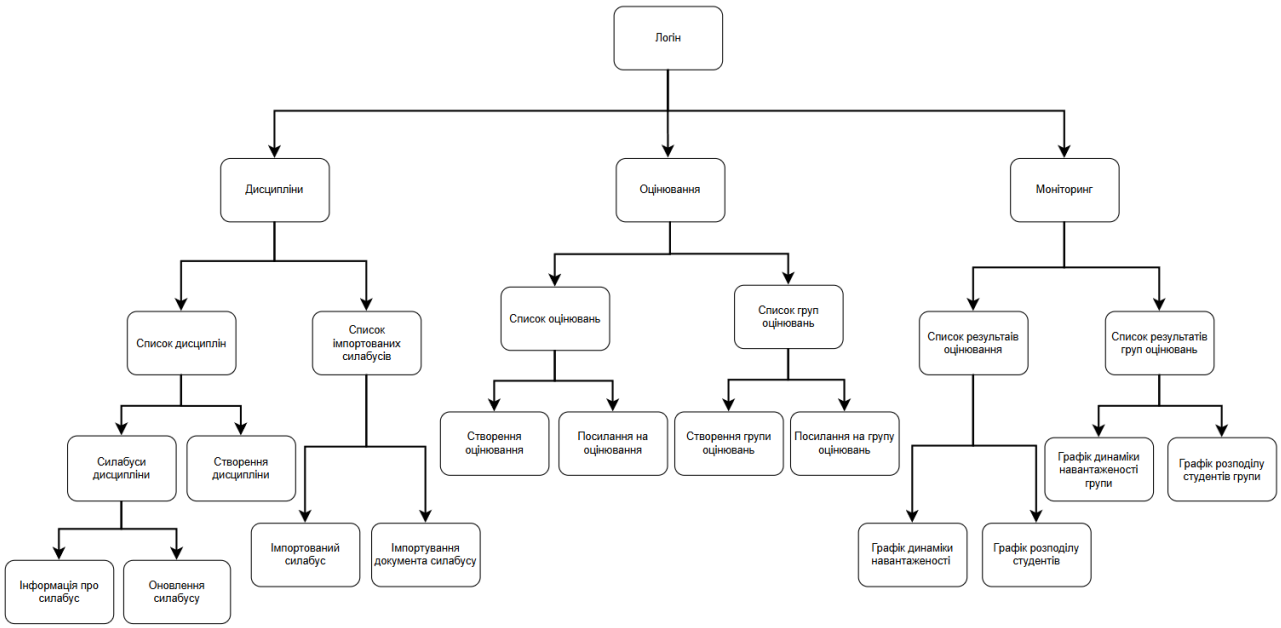


Рис. 2.2 Мапа TaskMon для адміністраторів

На мапі зображені усі основні сторінки, кожна з яких має свою функціональність та роль в роботі застосунку. Розглянемо усі сторінки цього застосунку:

- логін: Сторінка для автентифікації користувача, де після успішного входу він опиняється на трьох вкладках між якими може переключатися: дисципліни, оцінювання, моніторинг;
- дисципліни: Сторінка на якій можна переключатися між двома вкладками (список дисциплін, список імпортованих силабусів), для взаємодії з дисциплінами;
- список імпортованих силабусів: Сторінка з імпортованими файлами які можна переглянути;

- імпортований силабус: Користувач опиняється на ній після того як натиснув на один з імпортованих файлів, де може переглядати імпортовані дані;
- імпортування документа силабусу: Користувач опиняється на цій сторінці після того як вибрав опцію імпорту на сторінці з імпортованими файлами, де він може обрати файл силабусу та імпортувати його;
- список дисциплін: Сторінка для перегляду дисциплін з якими можна взаємодіяти;
- створення дисципліни: Користувач опиняється на цій сторінці після натиснення на кнопку створення дисципліни, де він створює дисципліну;
- силабуси дисципліни: Користувач опиняється на ній після того як натиснув на одну із дисциплін Тут відображаються архівовані силабуси дисципліни та один активний;
- інформація про силабус: Сторінка з відображенням даних про створений навчальний план дисципліни, яку обрав користувач;
- оновлення силабусу: Сторінка для створення силабусу вручну, після чого в дисципліни з’являється новий начальний план, а старий архівується;
- оцінювання: Сторінка на якій можна переключатися між двома вкладками (список оцінювань, список груп оцінювань), для взаємодії з оцінюваннями;
- список оцінювань: Сторінка з відображенням створених оцінювань користувачем;
- створення оцінювання: Користувач опиняється на ній після того як обрав опцію створення на попередній сторінці, де він може створити оцінювання;
- посилання на оцінювання: Користувач переходить на цю сторінку натиснувши на одне з оцінювань, де може копіювати посилання;
- список груп оцінювань: Сторінка з відображенням створених груп оцінювань користувачем;

- створення групи оцінювань: Користувач опиняється на цій сторінці після того як обрав опцію створення на попередній сторінці, де він може створити групу оцінювань;
- посилання на групу оцінювань: Користувач переходить на цю сторінку натиснувши на одну з груп оцінювань, де може копіювати посилання на цю групу;
- моніторинг: Сторінка на якій можна переключатися між двома вкладками (список результатів оцінювання, список результатів груп оцінювань), для перегляду результатів проходження оцінювань студентами;
- список результатів оцінювання: Сторінка на якій можна переключатися між двома графіками (графік динаміки навантаженості, графік розподілу студентів), для порівнювання отриманих даних з силабусом;
- графік динаміки навантаженості: Користувач опиняється на цій сторінці після того як натиснув на одне з оцінювань, де є графік для порівняння фактичних годин з годинами в силабусі;
- графік розподілу студентів: Користувач опиняється на ній також після натиснення на одне з оцінювань, де відображається графік з кількістю студентів що не справилися, або справилися з навантаженням;
- список результатів груп оцінювань: Сторінка на якій можна переключатися між двома графіками (графік динаміки навантаженості групи, графік розподілу студентів групи), для порівнювання отриманих даних з силабусом;
- графік динаміки навантаженості групи: Користувач опиняється на цій сторінці після того як натиснув на одне з груп оцінювань, де є графік для порівняння фактичних годин з годинами в силабусі для кожного з оцінювань, між якими може переключатися користувач;
- графік розподілу студентів групи: Користувач опиняється на ній також після натиснення на одне з груп оцінювань, де відображається графік з

кількістю студентів що не справилися, або справилися з навантаженням. Також можна переключатися між оцінюваннями для перегляду результатів кожного.

2.4 Мапа застосунку студента

Для застосунку студента також було створено мапу, на якій зображено всього 3 сторінки. Їх така мала кількість через те, що студентам достатньо просто проходити оцінювання, через що їх не потрібно плутати непотрібним функціоналом. Мапу для студентів продемонстровано на рисунку 2.3.

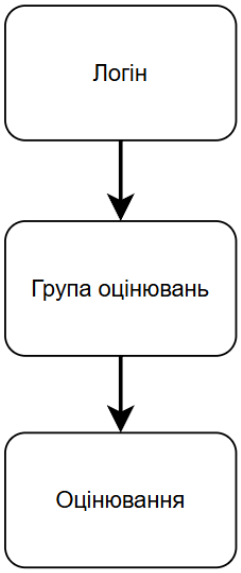


Рис. 2.3 Мапа TaskMon для студентів

Розглянемо усі сторінки застосунку для студентів:

- логін: Сторінка для автентифікації користувача, де після успішного входу він опиняється на сторінці групи оцінювань;
- група оцінювань: Сторінка з списком оцінювань, які може проходити студент. Після проходження, оцінювання позначається з статусом “Пройдено”;
- оцінювання: Сторінка з проходженням оцінювання, де можна відправити свої результати після їх проходження.

2.5 Проєктування інтерфейсу

Для проєктування інтерфейсу TaskMon було використано онлайн-сервіс Figma, що є ефективним інструментом для командної роботи над дизайном, який дозволяє генерувати ідеї, отримувати зворотний зв’язок, створювати інтерактивні прототипи та оптимізувати процес розробки продукту завдяки використанню дизайн-систем [9]. Особливістю роботи з інтерфейсом став плагін Material Design 3, що значно полегшує створення візуально збалансованого дизайну згідно з актуальними рекомендаціями від Google. Завдяки цьому плагіну можна швидко додавати готові компоненти, такі як кнопки, поля вводу, перемикачі та інші елементи. Це дає змогу підтримувати єдиний стиль у всьому додатку та економити час на візуальну адаптацію елементів. Figma у поєднанні з цим плагіном дає змогу сконцентруватися на створенні сторінок для користувацького досвіду, при цьому маючи інструменти для детального налаштування вигляду кожного елементу. Під час роботи дизайн легко оновлюється, компоненти можна змінювати централізовано, що особливо корисно для створення великого інтерфейсу. Таким чином, поєднання Figma з Material Design 3 стало ефективним рішенням для побудови сучасного та адаптивного інтерфейсу користувача.

Завдяки такій можливості я створив дизайн сторінок як для застосунку адміністраторів, так і для застосунку студентів використовуючи цей плагін. Подивитися дизайн сторінок можна на рисунках 2.4 та 2.5.

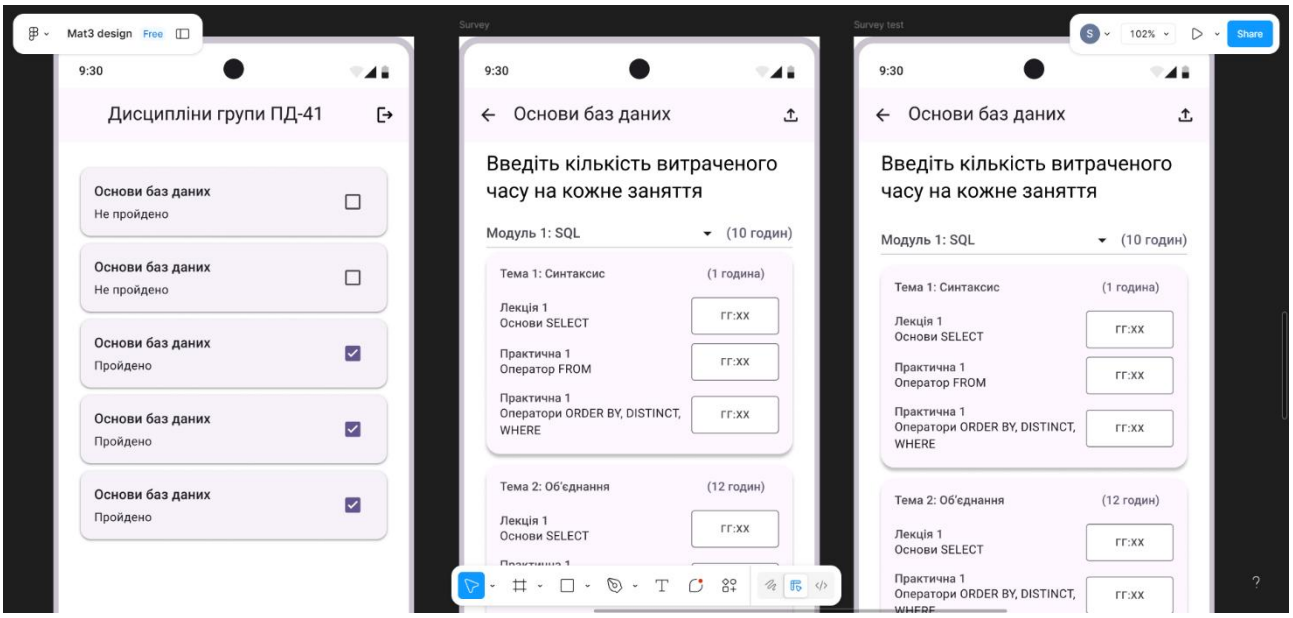


Рис. 2.4 Прототип сторінок для застосунку студентів

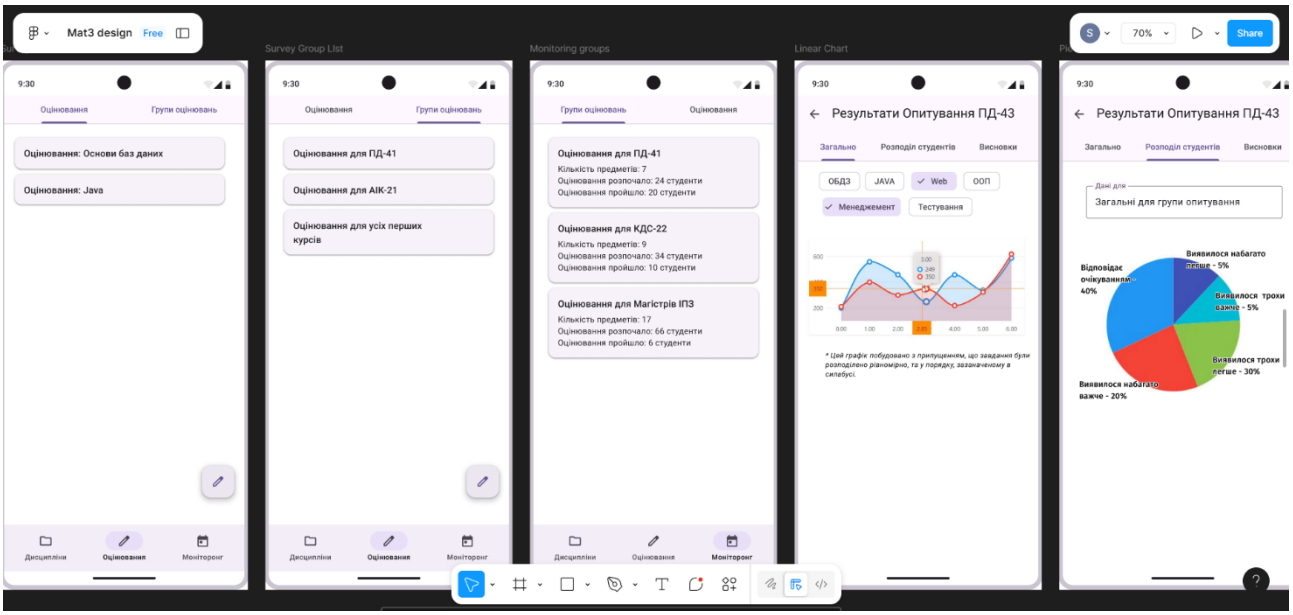


Рис. 2.5 Прототип сторінок для застосунку адміністраторів

2.6 Архітектура клієнтської частини застосунків

Під час розробки клієнтської частини для застосунків було обрано архітектурний шаблон MVVM, який є ефективним підходом для побудови структурованого, масштабованого та підтримуваного інтерфейсу користувача. Архітектуру цього шаблону можна побачити на рисунку 2.6.

У межах цього шаблону було чітко розділено логіку відображення (View) та логіку обробки даних і стану (ViewModel). View відповідає за візуалізацію інтерфейсу, взаємодію з користувачем та передачу подій до ViewModel. ViewModel містить всю бізнес-логіку клієнтської частини, обробку введених даних від користувача, керування станом інтерфейсу та взаємодію з моделлю.

Щодо моделі (Model), то вона реалізовувалась як набір об'єктів і сервісів, які забезпечували зв'язок із серверною частиною застосунку. Для цього використовувалася бібліотека Refit, що дозволяє створювати типізовані HTTP-клієнти. Завдяки їй можна було легко надсилати REST запити до API, що значно спрощувало обмін даними між клієнтом і сервером. Через ці інтерфейси клієнтська частина отримувала всю необхідну інформацію, яка потім оброблялась у ViewModel і передавалась у View для відображення.

Загалом така архітектура дозволила забезпечити чіткий розподіл обов’язків між компонентами та полегшила тестування і масштабування застосунку, а також значно покращила підтримку та повторне використання коду на стороні клієнта. MVVM у поєднанні з Refit став надійною основою для реалізації гнучкого та ефективного користувацького інтерфейсу.

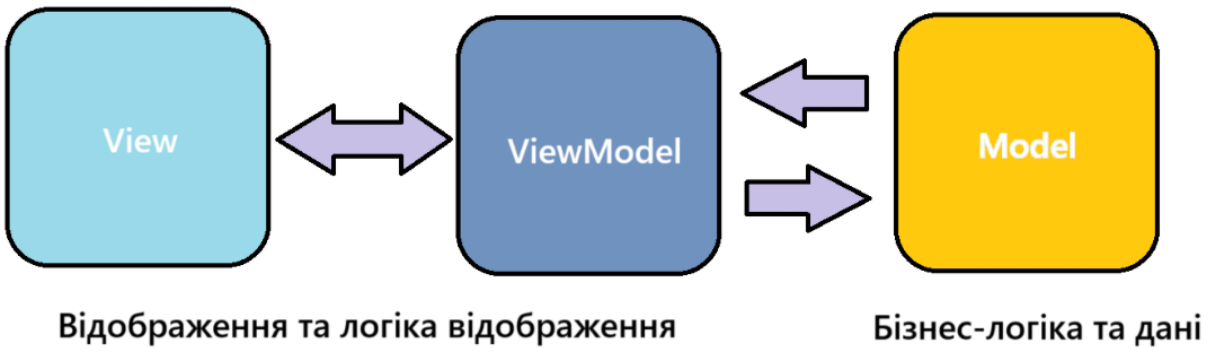


Рис. 2.6 Архітектурний шаблон MVVM

3 ОГЛЯД ЗАСОБІВ РЕАЛІЗАЦІЇ ЗАСТОСУНКІВ

Застосунки для моніторингу навчального навантаження студентів розроблювалися для платформи Android. Вони будуть використовувати клієнт-серверну архітектуру, через що мені потрібно було знайти та використати усі потрібні засоби реалізації для клієнтської частини. В результаті було обрано декілька програмних засобів для того щоб досягти поставлених цілей для створення двох застосунків.

3.1 Фреймворк .NET MAUI

.NET Multi-platform App UI (MAUI) – це сучасний кросплатформний фреймворк від Microsoft, який дозволяє розроблювати інтерфейси користувача для мобільних та десктопних застосунків використовуючи єдину базу коду на C# та XAML [10]. У порівнянні зі своїм попередником Xamarin.Forms, .NET MAUI пропонує покращену архітектуру, кращу інтеграцію з платформами, а також можливість повноцінно використовувати новітні можливості .NET 6 та його нові версії.

Розробка за допомогою .NET MAUI дозволяє скоротити витрати на розробку і обслуговування застосунків завдяки повторному використанню значної частини коду на всіх підтримуваних платформах. Інтерфейс користувача визначається за допомогою XAML, що дозволяє дизайнерам і розробникам ефективно співпрацювати. Платформа також забезпечує повну підтримку MVVM-архітектури, що підвищує масштабованість застосунку.

Ключовими перевагами .NET MAUI є: тісна інтеграція з .NET-екосистемою, повна підтримка різних середовищ розробки (Rider та Visual Studio), широке співтовариство розробників та багатий набір елементів керування, які можна розширювати. Завдяки MAUI, розробка B2B, B2C і B2E застосунків стає набагато ефективнішою, адже одне рішення може задовольнити вимоги різних платформ, не жертвуючи продуктивністю [11, с. 1].

3.2 Мова програмування C#

Мова програмування C# є основною мовою реалізації клієнтської частини в проєкті. Це сучасна мова програмування на платформі .NET, яка є безкоштовною, відкритою та підтримує розробку для різних операційних систем [12]. C# призначена для спрощення розробки Windows-додатків та корпоративного програмного забезпечення, пропонуючи широкий набір функцій, які сприяють швидкій розробці застосунків, об'єктно-орієнтованому проєктуванню та безпечному виконанню коду [13, с. 5743].

Однією з головних переваг C# є підтримка асинхронного програмування за допомогою ключових слів async/await, що дозволяє ефективно працювати з мережею, базами даних та іншими асинхронними ресурсами без блокування основного потоку. Також мова має вбудовану підтримку LINQ, що дозволяє писати потужні та зрозумілі запити до колекцій та баз даних у стилі SQL, без використання сторонніх бібліотек.

C# підтримує інкапсуляцію, успадкування, поліморфізм, а також має розвинуту систему типів. Завдяки цьому, код на C# є надійним, масштабованим і легко тестується. Управління пам’яттю автоматизоване завдяки збирачу сміття (Garbage Collector), що знижує кількість помилок і витоків пам’яті.

У контексті .NET MAUI, C# дозволяє реалізовувати не тільки UI-логіку, а й бізнес-логіку, взаємодію з API, обробку подій, моделі даних, що суттєво спрощує архітектуру застосунку та дозволяє тримати весь код в одному технологічному стеку.

3.3 Середовище розробки Rider

JetBrains Rider – це інтегроване середовище розробки IDE, яке ідеально підходить для .NET-розробників, що прагнуть підвищити ефективність роботи та спростити процес розробки [14]. Воно забезпечує високопродуктивне програмування на C# із розширеними інструментами для рефакторингу коду.

Для розробки проєктів на .NET MAUI, Rider забезпечує підтримку шаблонів MAUI, інтеграцію з .NET CLI та підтримку емуляторів Android/IOS. IDE має вбудований Git, який дозволяє зручно керувати гілками, конфліктами та злиттям. Rider підтримує роботу з NuGet-пакетами без необхідності переходу до окремих інструментів, що спрощує встановлення й оновлення бібліотек.

Інтерфейс Rider можна налаштовувати під особисті потреби, що дозволяє максимально оптимізувати робочий процес. Завдяки можливості використання плагінів, середовище розширюється під конкретні задачі розробника. У порівнянні з Visual Studio, Rider споживає менше ресурсів системи та надає гнучкі інструменти для аналізу коду, що підвищує загальну ефективність розробки.

3.4 Бібліотеки для розширення можливостей розробки

Під час створення клієнтської частини застосунку, особливу увагу було приділено використанню сторонніх бібліотек, які значно розширюють функціональні можливості базового фреймворку .NET MAUI. Завдяки цим інструментам вдалося реалізувати сучасний, гнучкий та зручний для користувача інтерфейс, забезпечивши ефективну взаємодію з серверною частиною, а також допомогло підтримати високий рівень візуалізації даних.

3.4.1 MVVM Toolkit

MVVM Toolkit – це модульна бібліотека для шаблону MVVM, що входить до складу .NET Community Toolkit [15]. Вона значно спрощує реалізацію шаблону MVVM у застосунках на .NET MAUI. Цей підхід дозволяє розділити логіку інтерфейсу користувача та бізнес-логіку.

Бібліотека надає атрибути та базові класи, які автоматично генерують необхідний шаблонний код. Використовуючи [ObservableProperty], розробник може швидко створити властивість, яка підтримує повідомлення про зміни (INotifyPropertyChanged), без необхідності вручну писати десятки рядків коду.

Також бібліотека включає в себе атрибут [RelayCommand], що дозволяє легко створювати команди для зв’язку між ViewModel та View.

Ця бібліотека підтримує асинхронні команди, генерацію подій та властивостей, локалізацію, навігацію та інші поширені завдання у .NET MAUI. Завдяки цьому, розробникам не потрібно концентрувати увагу на шаблонному коді, що значно пришвидшує розробку та зменшує кількість помилок.

3.4.2 Uranium UI

Uranium UI – це бібліотека користувацьких інтерфейсів, що має відкритий та безкоштовний набір компонентів для .NET MAUI, що включає елементи управління та утиліти для розробки сучасних додатків [16]. Вона орієнтована на тих розробників, які хочуть точно реалізувати дизайн-макети з Figma, зберігаючи при цьому високу продуктивність застосунку. Також є наявність документації, де можна ознайомитися з усіма елементами UI та як їх використовувати. Цю документацію можна переглянути на рисунку 3.1.

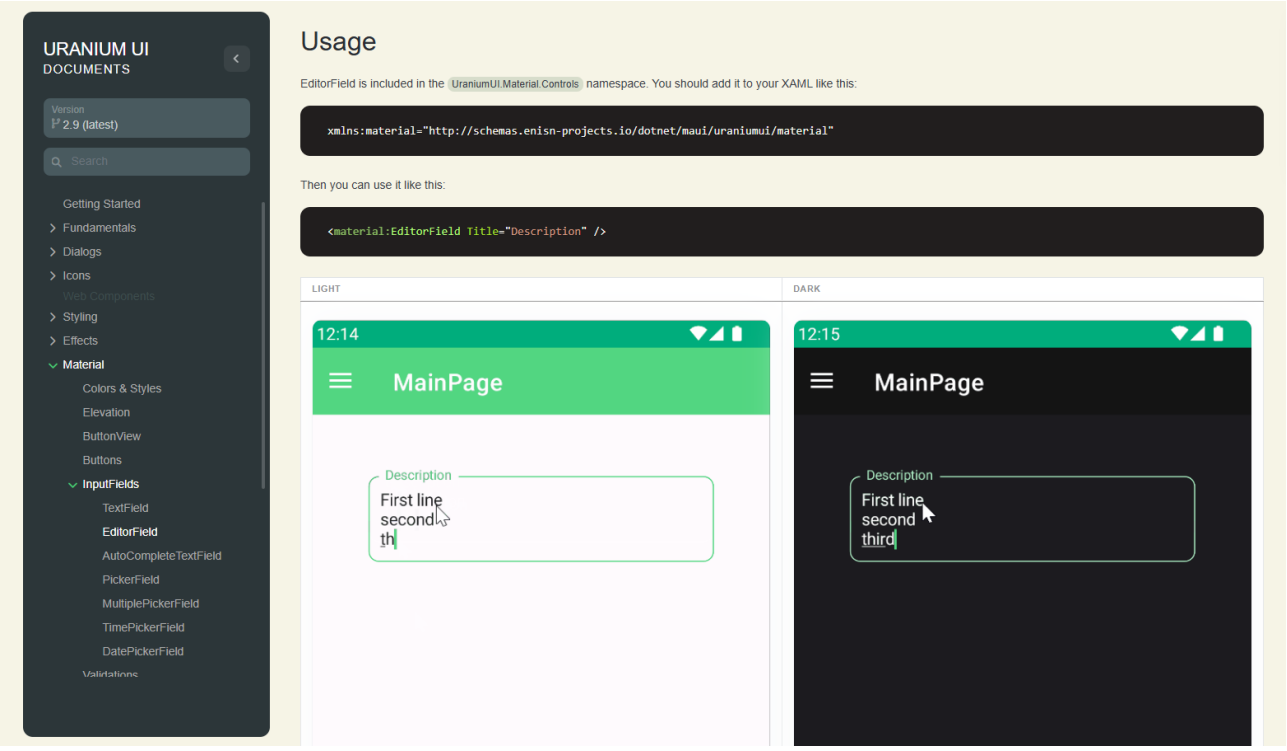


Рис. 3.1 Документація елементів з Uranium UI

Цей UI додає та вдосконалює стандартні елементи керування .NET MAUI, такі як: PickerField, TabView, EditorField, Elevation та інші. Бібліотека також підтримує теми, стилі та адаптивність до розмірів екрану. Це дає змогу створювати інтерфейси без потреби робити кастомні елементи UI вручну.

Використання Uranium UI дозволяє скоротити час розробки інтерфейсів та досягти кращої візуальної відповідності до дизайнерських макетів. Бібліотека активно підтримується, добре документована й інтегрується з MAUI без конфліктів, що робить її ідеальним вибором для застосунків з акцентом на зовнішній вигляд.

3.4.3 Refit

Refit – це бібліотека для .NET Core, Xamarin та .NET, яка забезпечує автоматичну генерацію REST-запитів з підтримкою безпеки типів [17]. Вона дозволяє розробникам визначати API як інтерфейси з використанням наступних атрибутів: Get, Post, Put, Delete. На основі цих інтерфейсів Refit автоматично генерує реалізації клієнтів, які обробляють HTTP-запити та відповіді.

Однією з ключових переваг Refit є безпечність типів, а завдяки використанню інтерфейсів, компілятор перевіряє правильність викликів API на етапі компіляції, що зменшує кількість помилок під час їх виконання. Також Refit автоматично перетворює передані дані з клієнта в JSON-дані, що спрощує обробку відповідей від сервера.

При використанні Refit, можна легко інтегрувати клієнтську частину застосунку з серверними API. Це особливо корисно при розробці застосунків на різних платформах, де необхідно забезпечити однакову логіку взаємодії з сервером на різних платформах.

3.4.4 Auth0

Auth0 – це зручна платформа з можливістю налаштовування, яка забезпечує автентифікацію й авторизацію та легко інтегрується у застосунки

[18]. Вона підтримує різні протоколи автентифікації, включаючи OpenID Connect та OAuth, що дозволяє легко інтегрувати її з .NET MAUI застосунками.

Інтеграція з Auth0 забезпечує безпечний та зручний механізм входу для користувачів. Розробники можуть налаштувати соціальні входи (Google, Facebook), а також традиційні методи автентифікації з використанням електронної пошти та пароля. Auth0 також надає можливість керувати сесіями користувачів, зберігати токени доступу та оновлення, що спрощує доступ до захищених ресурсів.

Крім того, Auth0 пропонує розширені функції безпеки: багатофакторна автентифікація, виявлення підозрілої активності та управління ролями користувачів. Це дозволяє забезпечити високий рівень захисту даних та відповідність сучасним стандартам безпеки.

3.4.5 LiveCharts

LiveCharts – це бібліотека для побудови графіків у .NET, що підтримує роботу на кількох платформах та у різних середовищах розробки [19]. Вона дозволяє візуалізувати дані в реальному часі, що особливо корисно для відображення статистики, аналітики та інших динамічних даних. Також є наявність документації, де можна ознайомитися з усіма графіками та як їх використовувати. Переглянути цю документацію можна на рисунку 3.2.

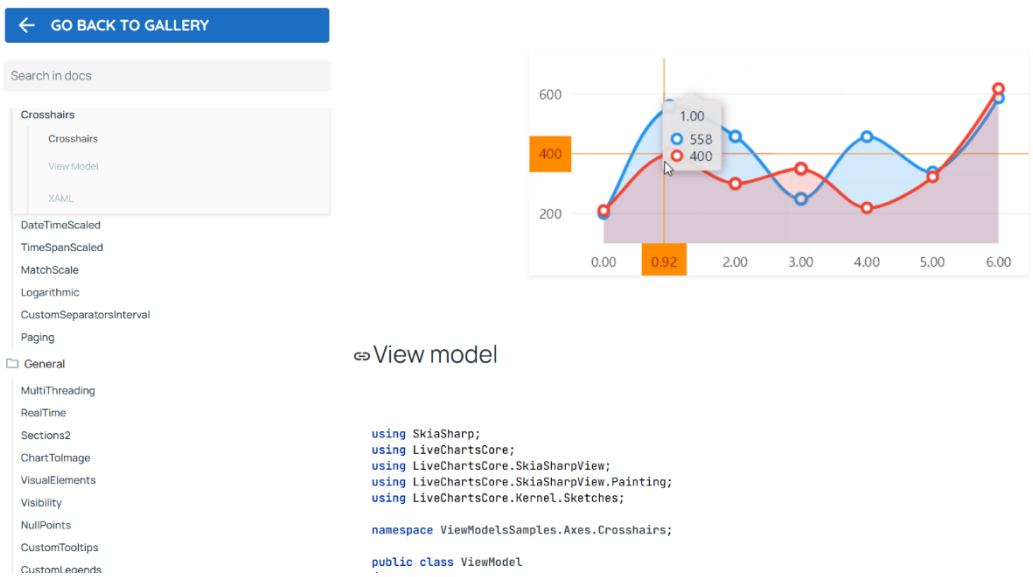


Рис. 3.2 Документація графіків з LiveCharts

Бібліотека підтримує різні типи графіків, такі як лінійні, стовпчикові, кругові, а також більш складні варіанти, включаючи теплові карти та географічні карти. LiveCharts забезпечує плавну анімацію при зміні даних, що покращує користувацький досвід.

У контексті .NET MAUI, LiveCharts дозволяє створювати адаптивні графіки, які автоматично підлаштовуються під розмір екрану та орієнтацію пристрою. Це забезпечує однаковий вигляд графіків на різних пристроях. Також бібліотека підтримує теми оформлення, що дозволяє легко інтегрувати графіки в загальний дизайн застосунку.

Завдяки LiveCharts, розробники можуть ефективно представити складні дані у візуально привабливій формі, що сприяє кращому розумінню та аналізу інформації користувачами.

4 РЕАЛІЗАЦІЯ КЛІЄНТСЬКОЇ ЧАСТИНИ ЗАСТОСУНКІВ ДЛЯ МОНІТОРИНГУ НАВЧАЛЬНОГО НАВАНТАЖЕННЯ СТУДЕНТІВ

4.1 Структура проєкту в Rider

Наявність структури дуже важлива при розробці будь-якого програмного забезпечення, оскільки вона дозволяє чітко організувати код для забезпечення розуміння його логіки та спрощує навігацію в проєкті як для самого розробника, так і для інших людей що мають доступ до проєкту. Добре продумана структура – це основа масштабованості та підтримки роботи застосунку в майбутньому. У середовищі розробки JetBrains Rider, дана організація проєкту дозволяє легко орієнтуватися між модулями, ресурсами, файлами конфігурацій та іншими складовими проєкту.

У моїй роботі структура проєкту передбачає створення двох окремих проєктів: “TaskMonMobile” та “TaskMonAdmin”, зображених на рисунку 4.1. Такий підхід забезпечує чітке розподілення функціональності між застосунком для студентів та застосунком для адміністраторів.

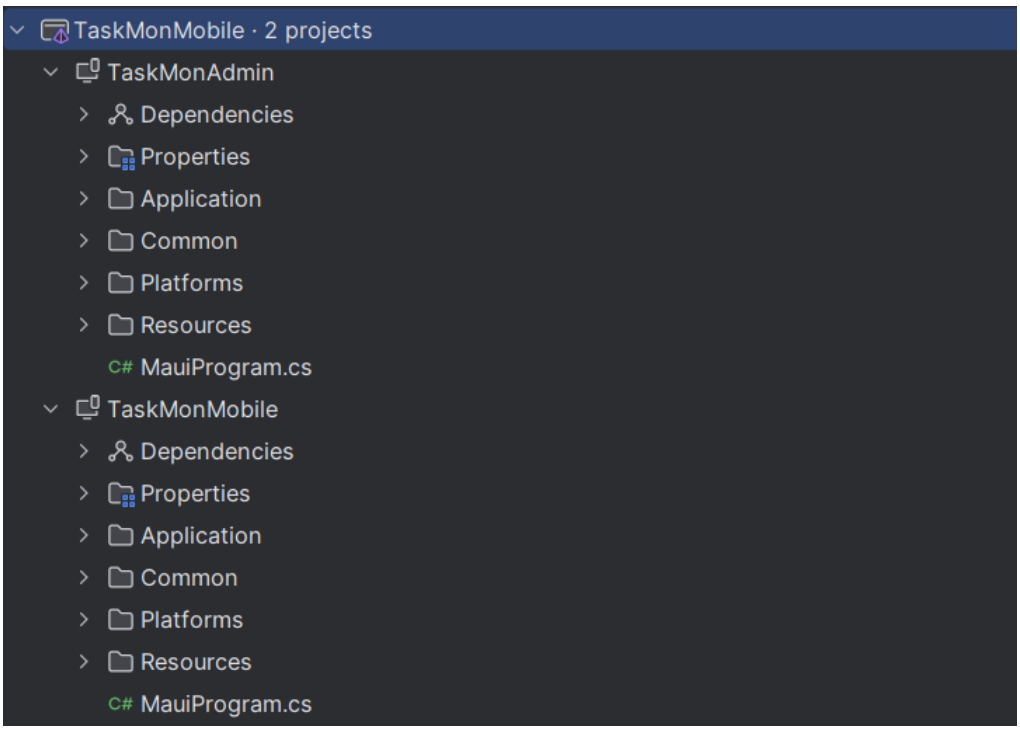


Рис. 4.1 Два окремих проєкта

У кожному з цих проєктів, структура дотримується єдиного підходу, що забезпечує зручність у розробці та повторне використання коду. Основу цієї структури складають чотири ключові папки: Application, Common, Platforms та Resources. Кожна з них виконує свою важливу роль у побудові застосунку, що ви можете побачити на рисунку 4.2:

- Application містить в собі файли App та AppShell для налаштування усіх сторінок загалом та змінюючи основний вигляд застосунку. В App також розписується логіка в момент запуску застосунку, тобто відбуваються певні дії в момент запуску застосунку. В AppShell можна взаємодіяти з розташуванням створених сторінок та робити для них навігацію. Також тут знаходяться папки в кожній з яких зберігаються основні сторінки користувацького інтерфейсу у форматі .xaml, а також відповідні їм файли з логікою на C# та ViewModel. Саме тут реалізовується архітектурний шаблон MVVM, який дозволяє відокремити візуальну частину від бізнес-логіки, покращуючи написання коду;
- Common відповідає за спільну логіку, яку можна використовувати у різних частинах застосунку. Тут реалізовано процес авторизації, зберігання токенів доступу, а також створення кастомних компонентів інтерфейсу. Папка є свого роду бібліотекою набором своїх елементів, які можна використовувати всередині застосунку;
- Platforms містить специфічні налаштування для різних платформ, де особливу увагу про приділено Android. Тут можна налаштувати поведінку при запуску застосунку. Це дозволяє зробити застосунок більш адаптивним до конкретної операційної системи, зберігаючи при цьому основну логіку кросплатформною;
- Resources зберігає в собі всі ресурси застосунку, які використовуються для оформлення інтерфейсу: зображення, іконки, стилі. Вона забезпечує централізоване управління зовнішнім виглядом застосунку, що особливо зручно при необхідності швидко змінити тему чи якусь іконку в застосунку.

Також в кожному з двох проєктів є файл MauiProgram, де підключаються компоненти з усіх встановлених бібліотек та створенні сторінки. Саме в цьому файлі відбувається підключення до серверної частини, до якої підключення відбувається через мікросервіси.

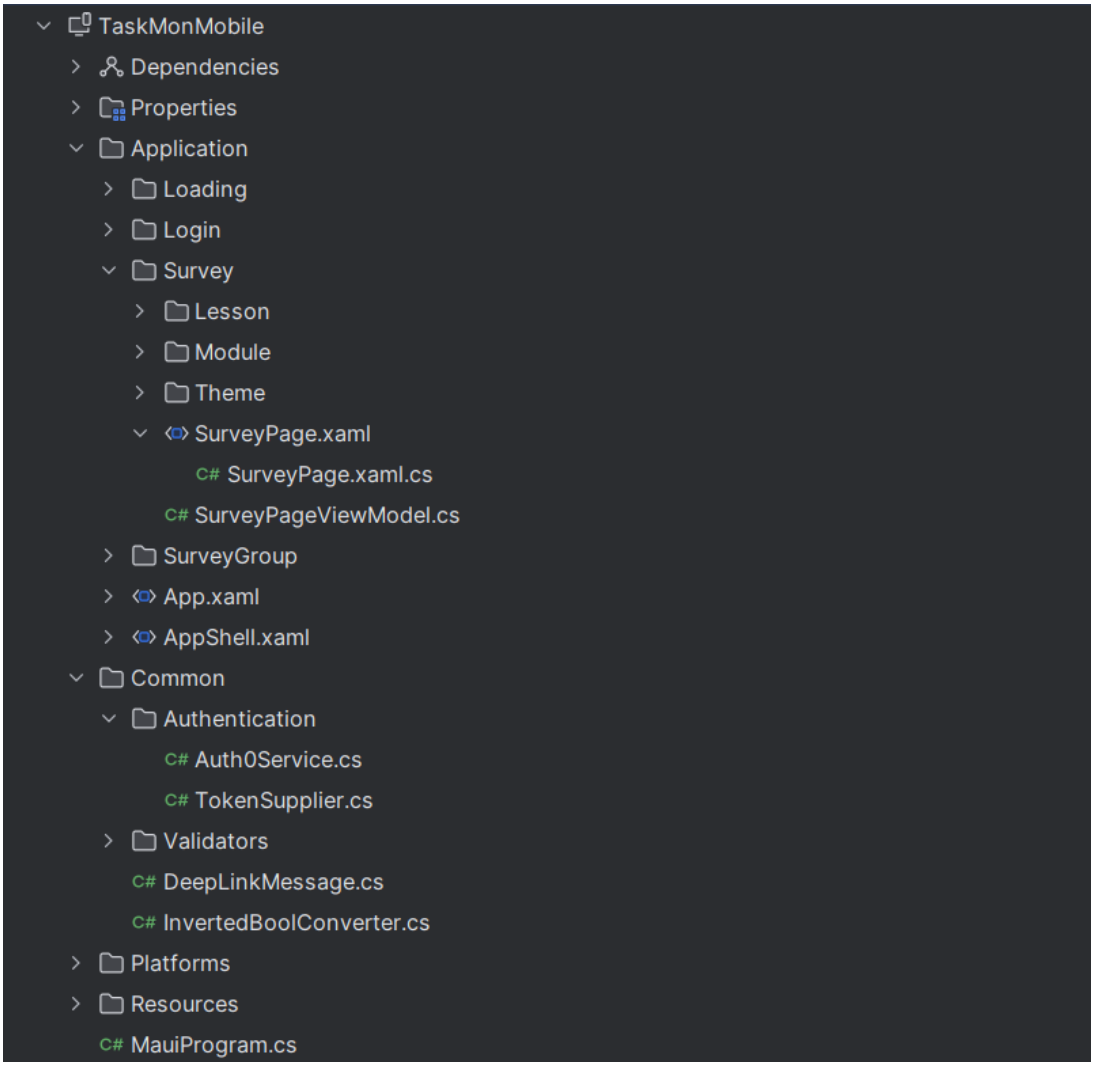


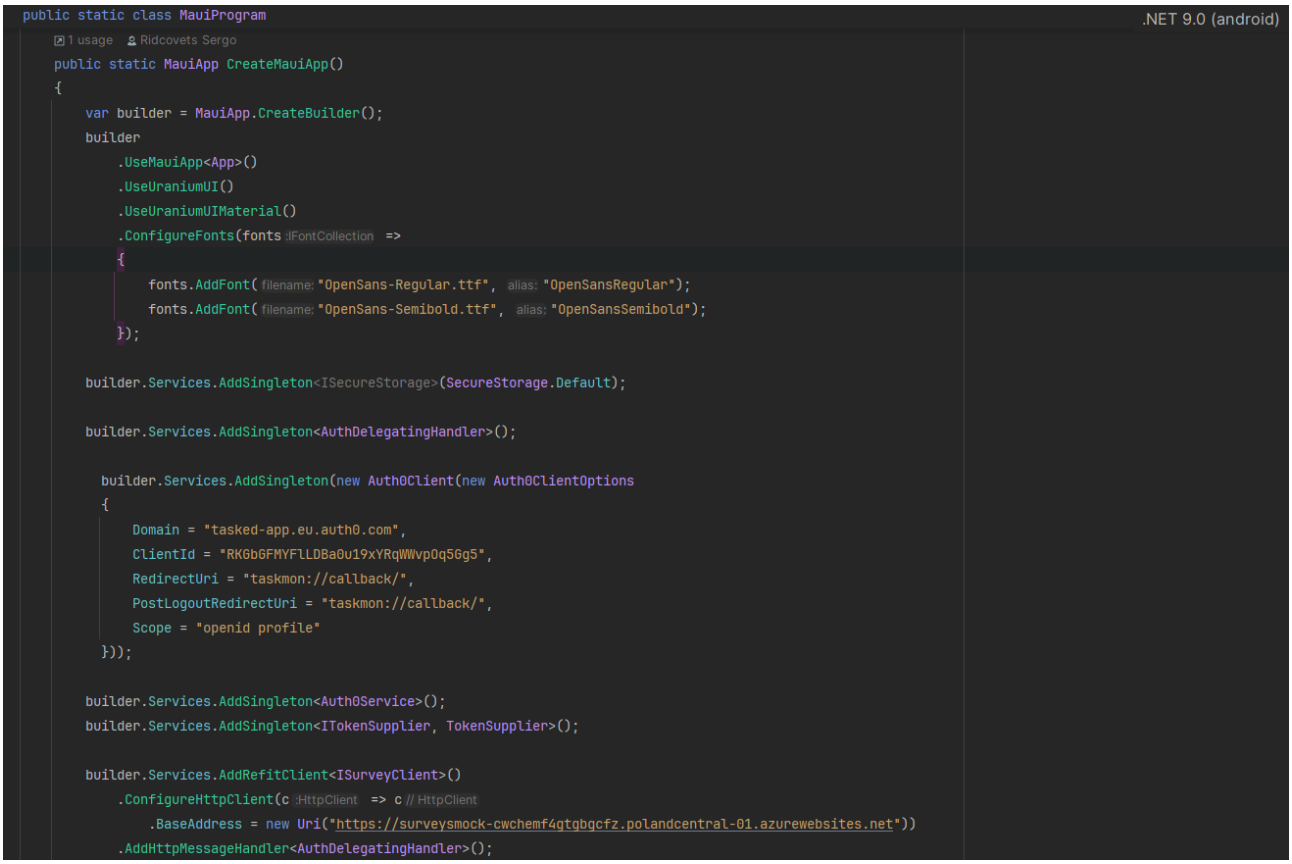
Рис. 4.2 Структура кожного проєкту

4.2 Застосунок для студентів

У проєкті “TaskMonMobile” було реалізовано застосунок для студентів, де вони можуть проходити оцінювання, для перевірки їх навчального навантаження. Основна мета розробки полягає в наданні студентам зручного інструменту для можливості вписувати витрачений час на кожне завдання конкретних дисциплін. Завдяки інтерактивному інтерфейсу та підтримці

мобільної платформи Android, застосунок дозволяє користувачам взаємодіяти з оцінюваннями та відправляти результати їх проходження. Тож розглянемо основні файли, що забезпечують роботу цього застосунку.

Спершу перейдемо до класу MauiProgram, який відповідає за налаштування та ініціалізацію застосунку для студентів, що продемонстрований на рисунку 4.3. У методі CreateMauiApp відбувається конфігурація основних компонентів та сервісів додатку.




```
public static class MauiProgram
    1 usage   Ridcovets Sergo
    public static MauiApp CreateMauiApp()
    {
        var builder = MauiApp.CreateBuilder();
        builder
            .UseMauiApp<App>()
            .UseUraniumUI()
            .UseUraniumUIMaterial()
            .ConfigureFonts(fonts IFontCollection =>
            {
                fonts.AddFont( filename: "OpenSans-Regular.ttf",  alias: "OpenSansRegular");
                fonts.AddFont( filename: "OpenSans-Semibold.ttf",  alias: "OpenSansSemibold");
            });

        builder.Services.AddSingleton<ISecureStorage>(SecureStorage.Default);

        builder.Services.AddSingleton<AuthDelegatingHandler>();

          builder.Services.AddSingleton(new Auth0Client(new Auth0ClientOptions
          {
              Domain = "tasked-app.eu.auth0.com",
              ClientId = "RKGbGFMYFlLDBa0u19xYRqWWvpOq5Gg5",
              RedirectUri = "taskmon://callback/",
              PostLogoutRedirectUri = "taskmon://callback/",
              Scope = "openid profile"
          }));

        builder.Services.AddSingleton<Auth0Service>();
        builder.Services.AddSingleton<ITokenSupplier, TokenSupplier>();

        builder.Services.AddRefitClient<ISurveyClient>()
            .ConfigureHttpClient(c :HttpClient  => c // HttpClient
                .BaseAddress = new Uri("https://surveysmock-cwchemf4gtgbgcfz.polandcentral-01.azurewebsites.net"))
            .AddHttpMessageHandler<AuthDelegatingHandler>();
```

Рис. 4.3 Файл MauiProgram

Спочатку створюється базовий будівельник додатку за допомогою MauiApp.CreateBuilder(). Додаток конфігурується для використання користувацького інтерфейсу Uranium UI та його матеріального стилю через виклики UseUraniumUI() та UseUraniumUIMaterial(). Також налаштовуються шрифти для застосунку.

Далі відбувається реєстрація залежностей у контейнері. Зареєстровано сервіс ISecureStorage для безпечного зберігання даних та обробник

AuthDelegatingHandler для управління HTTP-запитами, а також клієнт Auth0Client для авторизації користувачів. Для Auth0Client налаштовані домен "tasked-app.eu.auth0.com", ідентифікатор клієнта, URI перенаправлення після логіну та виходу, а також визначено необхідні області доступу.

Також реєструються Auth0Service для роботи з авторизацією та ITokenSupplier для управління токенами. За допомогою RefitR налаштовується HTTP-клієнт для взаємодії з API сервісу проходження оцінювань. Також було додано обробник повідомлень для автоматичного додавання токенів авторизації.

Вкінці основних налаштувань, реєструються також основні сторінки додатку: SurveyPage та SurveyGroupPage. Ці сторінки ми розглянемо окремо з наявністю екранних форм.

Далі йде клас MainActivity, що налаштовується в платформі Android. Там реалізована система обробки глибоких посилань (Deep Link).

Клас MainActivity налаштований так, щоб відповідати на посилання за допомогою фільтрів Intent, зображених на рисунку 4.4. Фільтри налаштовані для обробки HTTP та HTTPS-посилань, які відповідають шаблонам "taskmon.com/surveys/invite" та "taskmon.com/groups/invite".

```
[Activity(Theme = "@style/Maui.SplashTheme", MainLauncher = true, LaunchMode = LaunchMode.SingleTop,
    ConfigurationChanges = ConfigChanges.ScreenSize | ConfigChanges.Orientation | ConfigChanges.UiMode |
                           ConfigChanges.ScreenLayout | ConfigChanges.SmallestScreenSize | ConfigChanges.Density)]


[IntentFilter( actions: new[] { Intent.ActionView },
    Categories = new[] { Intent.CategoryDefault, Intent.CategoryBrowsable },
    DataScheme = "http",
    DataHost = "taskmon.com",
    DataPathPrefix = "/surveys/invite" )]

[IntentFilter( actions: new[] { Intent.ActionView },
    Categories = new[] { Intent.CategoryDefault, Intent.CategoryBrowsable },
    DataScheme = "http",
    DataHost = "taskmon.com",
    DataPathPrefix = "/groups/invite" )]

[IntentFilter( actions: new[] { Intent.ActionView },
    Categories = new[] { Intent.CategoryDefault, Intent.CategoryBrowsable },
    DataScheme = "https",
    DataHost = "taskmon.com",
    DataPathPrefix = "/surveys/invite" )]

[IntentFilter( actions: new[] { Intent.ActionView },
    Categories = new[] { Intent.CategoryDefault, Intent.CategoryBrowsable },
    DataScheme = "https",
    DataHost = "taskmon.com",
    DataPathPrefix = "/groups/invite" )]
```

Рис. 4.4 Фільтри Intent в MainActivity

Коли застосунок отримує такі посилання, метод ProcessIntent аналізує їх та визначає тип посилання та його ідентифікатор, що зображено на рисунку 4.5. Залежно від типу посилання (одне оцінювання, група оцінювань), додаток зберігає відповідну інформацію в Preferences для подальшого використання та розсилає повідомлення через WeakReferenceMessenger. Це дозволяє іншим компонентам реагувати на отримані глибокі посилання. Також перед відправкою повідомлення відбувається перевірка, чи є ідентифікатор дійсним GUID.

```
private void ProcessIntent(Intent intent)
    if (uri != null)
        if (!string.IsNullOrEmpty(path))
            if (path.StartsWith("/surveys/invite/"))
            {
                string surveyId = path.Replace("/surveys/invite/", "");
                if (Guid.TryParse(surveyId, out _))
                {
                    Preferences.Set("PendingDeepLinkType", (int)DeepLinkType.Survey);
                    Preferences.Set("PendingDeepLinkId", surveyId);
                    Preferences.Set("HasPendingDeepLink", true);

                    WeakReferenceMessenger.Default.Send(new DeepLinkMessage(DeepLinkType.Survey, surveyId));
                }
                else
                {
                    WeakReferenceMessenger.Default.Send(new DeepLinkMessage(DeepLinkType.Survey,  id: string.Empty));
                }
            }
            else if (path.StartsWith("/groups/invite/"))
            {
                string groupId = path.Replace("/groups/invite/", "");
                if (Guid.TryParse(groupId, out _))
                {
                    Preferences.Set("PendingDeepLinkType", (int)DeepLinkType.Group);
                    Preferences.Set("PendingDeepLinkId", groupId);
                    Preferences.Set("HasPendingDeepLink", true);

                    WeakReferenceMessenger.Default.Send(new DeepLinkMessage(DeepLinkType.Group, groupId));
                }
                else
```

Рис. 4.5 Метод ProcessIntent в MainActivity

Тепер перейдемо до реалізації авторизації користувачів. було створено клас Auth0Service, що забезпечує взаємодію з сервісом Auth0 для автентифікації. Він керує логікою входу, виходу та токенами доступу.

У конструкторі класу ініціалізуються: Auth0Client, безпечне сховище ISecureStorage та логер для реєстрації помилок, зображених на рисунку 4.6. Метод LoginAsync виконує вхід користувача через Auth0, передаючи необхідні параметри аудиторії, що видно на рисунку 4.7. У разі успішного входу токен доступу зберігається в захищеному сховищі за допомогою ключа.

```
Ridcovets Sergo
public Auth0Service(Auth0Client auth0Client, ISecureStorage secureStorage, ILogger<Auth0Service> logger)
{
    _auth0Client = auth0Client;
    _secureStorage = secureStorage;
    _logger = logger;
}
```

Рис. 4.6 Конструктор класу Auth0Service

```
1 usage  Ridcovets Sergo
public async Task<bool> LoginAsync()
{
    try
    {
        var loginResult = await _auth0Client.LoginAsync( extraParameters: new
        {
            audience = "https://taskmon.com"
        }); // Task<LoginResult>

        if (loginResult.IsError)
        {
            _logger.LogError($"Не вийшло залогінитись: {loginResult.Error}");
            return false;
        }

        await _secureStorage.SetAsync(AccessTokenKey, loginResult.AccessToken);

        return true;
    }
    catch (Exception ex)
    {
        _logger.LogError(ex,  message: "Помилка під час логіну");
        return false;
    }
}
```

Рис. 4.7 Метод LoginAsync в Auth0Service

Метод LogoutAsync видаляє збережений токен доступу та викликає вихід через Auth0Client. Методи GetAccessTokenAsync та IsAuthenticatedAsync дозволяють отримати поточний токен доступу та перевірити, чи автентифікований користувач. Ці три методи можна побачити на рисунку 4.8. Ці методи забезпечують простий інтерфейс для роботи з авторизацією в інших частинах додатку.

```
1 usage  Ridcovets Sergo
public async Task<bool> LogoutAsync()
{
    try
    {
        _secureStorage.Remove(AccessTokenKey);

        await _auth0Client.LogoutAsync();
        return true;
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, message: "Помилка під час виходу з акаунту");
        return false;
    }
}

2 usages  Ridcovets Sergo
public async Task<string?> GetAccessTokenAsync()
{
    return await _secureStorage.GetAsync(AccessTokenKey);
}

2 usages  Ridcovets Sergo
public async Task<bool> IsAuthenticatedAsync()
{
    var token :string? = await GetAccessTokenAsync();
    return !string.IsNullOrEmpty(token);
}
```

Рис. 4.8 Інші методи в Auth0Service

Реалізація Auth0Service забезпечує обробку помилок під час входу та виходу, записуючи інформацію про них у журнал, що полегшує діагностику та усунення потенційних проблем.

Тепер повернемося до двох основних сторінок для роботи з оцінюваннями: сторінка “Групи оцінювань” та сторінка “Оцінювання”. Кожна з них представлена відповідними ViewModel, які відповідають за логіку, управління даними та взаємодію з користувачем.

Сторінка “Групи оцінювань” представлена класом SurveyGroupPageViewModel. Цей клас взаємодіє з серверним API через інтерфейс ISurveyClient для отримання даних про групи оцінювань. Основна функціональність цього класу полягає у завантаженні та відображенні списку доступних опитувань у межах певної групи.

При завантаженні даних групи через метод LoadSurveyGroupDataAsync виконується запит до серверного API, що зображено на рисунку 4.9, після чого отримані дані обробляються методом LoadFromModel. У випадку помилки при завантаженні групи встановлюється відповідне повідомлення. Важливою функцією є LoadCompletedSurveyStatus, яка перевіряє локально збережену інформацію про вже пройдені оцінювання, що можна побачити на рисунку 4.10.

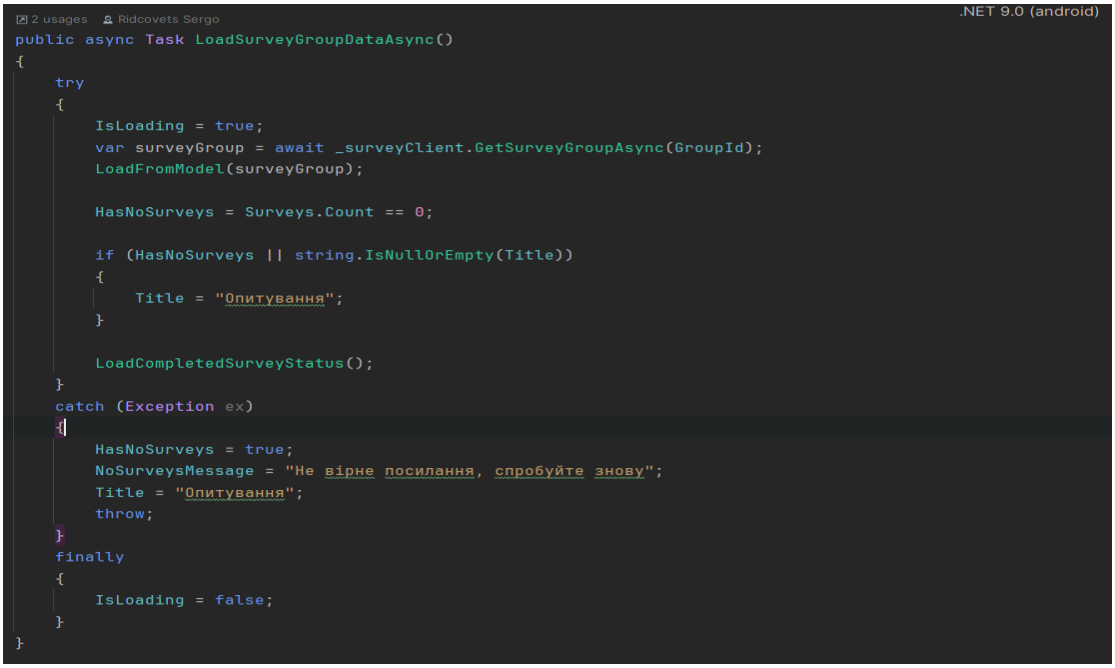


Рис. 4.9 Метод LoadSurveyGroupDataAsync

Рис. 4.10 Методи LoadFromModel та LoadCompletedSurveyStatus

Елементи списку оцінювань представлені класом SurveyItemViewModel, зображеного на рисунку 4.11, який зберігає ідентифікатор, заголовок та стан завершеності опитування. Він містить обчислювальну властивість CompletionStatus, яка показує статус проходження опитування. Також клас містить команду SelectSurveyCommand, яка дозволяє переходити до незавершених оцінювань через головний ViewModel групи.

```
12 usages  Ridcovets Sergo
public partial class SurveyItemViewModel : ObservableObject
{
    [ObservableProperty]
    private Guid _id;

    [ObservableProperty]
    private string _title;

    [ObservableProperty]
    private bool _isCompleted;

    2 usages
    public SurveyGroupPageViewModel ParentViewModel { get; set; }

    1 usage  Ridcovets Sergo
    public string CompletionStatus => IsCompleted ? "Пройдено" : "Не пройдено";

    1 usage  Ridcovets Sergo
    public ICommand SelectSurveyCommand => new Command( execute: () =>
    {
        if (!IsCompleted)
        {
            ParentViewModel.NavigateToSurveyCommand.Execute(Id);
        }
    });
}
```

Рис. 4.11 Клас SurveyItemViewModel

Наступна сторінка "Оцінювання" була представлена класом SurveyPageViewModel. Цей клас відповідає за роботу з конкретним оцінюванням і його структурою.

При завантаженні даних оцінювання через метод LoadSurveyDataAsync виконується запит до серверного API, після чого дані оброблюються методом LoadFromModel, що показано на рисунку 4.12. Найважливішою функцією є SubmitSurvey, яка збирає інформацію про витрачений час для кожного уроку з усіх модулів та тем, формує об'єкт відправки і надсилає його на сервер, що можна

побачити на рисунку 4.13. Після успішного відправлення, оцінювання позначається як завершене в локальному сховищі групи.

```
2 usages  Ridcovets Sergo
public async Task LoadSurveyDataAsync()
{
    try
    {
        var survey = await _surveyClient.GetSurveyAsync(Id);
        LoadFromModel(survey);
    }
    catch (Exception ex)
    {
        await Application.Current.MainPage.DisplayAlert( title: "Помилка",  message: $"Виникла помилка:
    }
}
```

Рис. 4.12 Метод LoadSurveyDataAsync

```
    private async Task SubmitSurvey()
        try
            foreach (var module in Modules)
            {
                foreach (var theme in module.Themes)
                {
                    foreach (var lesson in theme.Lessons)
                    {
                        if (lesson.TimeSpend > 0)
                        {
                            var assessment = new Assessment(
                                lesson.Id,
                                lesson.TimeSpend
                            );
                            assessments.Add(assessment);
                        }
                    }
                }
            }

            if (assessments.Count == 0)
            {
                await Application.Current.MainPage.DisplayAlert( title: "Не відправлено",  message: "Вкажіть час хоча б для одног
                return;
            }

            Submission submission = GroupId.HasValue
                ? new Submission(assessments, GroupId.Value)
                : new Submission(assessments);

            await _surveyClient.SubmitSurveyAsync(Id, submission);
```

Рис. 4.13 Метод SubmitSurvey

Структура оцінювання представлена ієрархією класів: ModuleViewModel, ThemeViewModel та LessonViewModel. Кожен з цих класів оброблює свої специфічні дані та обчислює загальний час, витрачений на відповідному рівні.

ModuleViewModel відображає модуль оцінювання з його ідентифікатором, заголовком та часом, витраченим на весь модуль, що продемонстровано на рисунку 4.14. Він також має властивість IsExpanded для керування відображенням вмісту модуля та команду ToggleExpanded для зміни цього стану.

```
[RelayCommand]
2 usages  Ridcovets Sergo
private void ToggleExpanded()
{
    IsExpanded = !IsExpanded;
}

1 usage  Ridcovets Sergo
public static ModuleViewModel FromModel(Module module)
{
    var viewModel = new ModuleViewModel
    {
        Id = module.Id,
        Title = module.Title,
        TimeSpend = 0,
        Themes = new ObservableCollection<ThemeViewModel>(),
        IsExpanded = true
    };

    foreach (var theme in module.Themes)
    {
        viewModel.Themes.Add(ThemeViewModel.FromModel(theme, viewModel));
    }

    return viewModel;
}
```

Рис. 4.14 Клас ModuleViewModel

ThemeViewModel представляє тему в межах модуля та містить метод RecalculateTimeSpend для підрахунку загального часу, витраченого на всі завдання в темі, що зображено на рисунку 4.15.

```
1 usage  Ridcovets Sergo                                                    .NET 9.0 (android)
public void RecalculateTimeSpend()
{
    TimeSpend = Lessons.Sum(l :LessonViewModel => l.TimeSpend);
}

1 usage  Ridcovets Sergo
partial void OnTimeSpendChanged(float value)
{
    OnPropertyChanged(nameof(ThemeTimeSpendDisplay));
}

1 usage  Ridcovets Sergo
public static ThemeViewModel FromModel(Theme theme, ModuleViewModel parentModule)
{
    var viewModel = new ThemeViewModel
    {
        Id = theme.Id,
        Title = theme.Title,
        TimeSpend = 0,
        Lessons = new ObservableCollection<LessonViewModel>()
    };

    foreach (var lesson in theme.Lessons)
    {
        viewModel.Lessons.Add(LessonViewModel.FromModel(lesson, parentModule, viewModel));
    }

    return viewModel;
}
```

Рис. 4.15 Клас ThemeViewModel

LessonViewModel є найнижчим рівнем ієрархії та представляє конкретне завдання. Він дозволяє користувачу вводити час, витрачений на нього, та автоматично оновлює загальний час у батьківських об'єктах (темі та модулі) за допомогою методу UpdateParentTimeSpend, що зображено на рисунку 4.16.

Рис. 4.16 Клас LessonViewModel

У всіх цих класах використовується атрибут [ObservableProperty] з бібліотеки CommunityToolkit.Mvvm для автоматичного створення властивостей, що підтримують сповіщення про зміни, а також атрибути [RelayCommand] для створення команд. Це дозволяє ефективно зв'язувати дані з інтерфейсом користувача за допомогою механізму прив'язки даних.

В висновку було створено застосунок для студентів, в якому користувачі можуть проходити оцінювання та групу оцінювань переходячи за посиланням. Де користувачі концентруватимуться переважно на цих двох сторінках, зображених на рисунках 4.17 та 4.18.

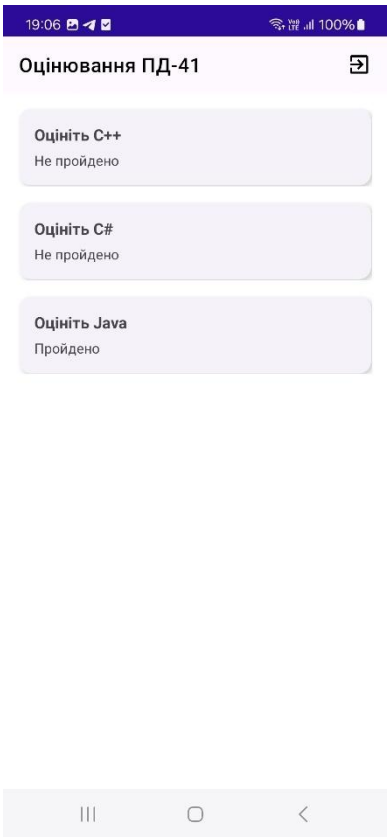


Рис. 4.17 Сторінка групи оцінювань

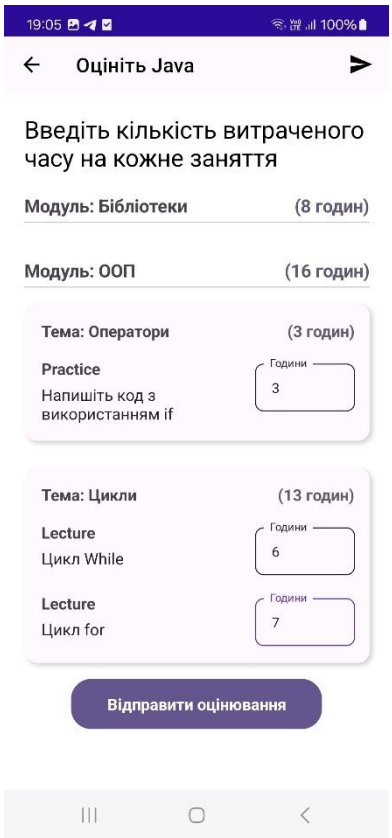


Рис. 4.18 Сторінка оцінювань

4.3 Застосунок для адміністраторів

У проєкті “TaskMonAdmin” було реалізовано застосунок для адміністратора, де він може робити наступне: створювати дисципліни, створювати силабус вручну та імпортувати його з документу, спостерігати за результатами проходження оцінювань за допомогою графіків. Основна мета розробки полягає в наданні адміністраторам зручного інструменту для можливості створювати оцінювання по конкретним дисциплінам та переглядати результати стосовно витраченого часу на кожне завдання конкретних дисциплін. Застосунок для адміністраторів переважно також має схожі за написанням коду сторінки, які були в застосунку для студентів. Тому основну увагу буде приділено новим функціям як імпорт силабусу та відображення графіків. Тож розглянемо деякі файли, що розширюють функціонал застосунку для адміністраторів.

На початку було створено власний елемент UI, якого не існувало серед запропонованих. Було реалізовано кастомний CustomChip, який представляє собою інтерактивний компонент для відображення вибраних елементів. Цей компонент розширює стандартний ContentView і додає до нього специфічну функціональність та візуальний стиль.

CustomChip реалізований як самостійний елемент інтерфейсу з можливістю вибору та візуальною індикацією стану. Він має декілька ключових властивостей Bindable Properties, які дозволяють контролювати його вигляд та поведінку, зображених на рисунку 4.19. До них належать:

- TextProperty – властивість для встановлення текстового вмісту компонента, який відображається користувачу;
- IsSelectedProperty – властивість для відстеження стану вибору компонента;
- IconSourceProperty – властивість для встановлення зображення іконки, яка відображається коли чіп вибрано.

```
public static readonly BindableProperty TextProperty = BindableProperty.Create(
    nameof(Text), typeof(string), typeof(CustomChip), defaultValue: string.Empty, propertyChanged: OnTextChanged);

public static readonly BindableProperty IsSelectedProperty = BindableProperty.Create(
    nameof(IsSelected), typeof(bool), typeof(CustomChip), defaultValue: false, BindingMode.TwoWay, propertyChanged: OnIsSelectedChanged);

public static readonly BindableProperty IconSourceProperty = BindableProperty.Create(
    nameof(IconSource), typeof(string), typeof(CustomChip), defaultValue: "check.png", propertyChanged: OnIconSourceChanged);
```

Рис. 4.19 Bindable Properties в класі CustomChip

Візуально CustomChip складається з кількох вкладених елементів:

- Border – основний контейнер з округленими кутами, що забезпечує зовнішній вигляд чіпа;
- StackLayout – горизонтальне розташування для організації внутрішніх елементів;
- Label – текстова мітка для відображення вмісту властивості Text;
- Image – іконка для індикації стану вибору, яка за замовчуванням є зображенням галочки.

При ініціалізації компонента створюються всі необхідні елементи з відповідними стилями та встановлюється TapGestureRecognizer для обробки дотиків, що продемонстровано на рисунку 4.20. Цей обробник жестів відповідає за зміну стану вибору чіпа при натисканні на нього.

```
var tapGestureRecognizer = new TapGestureRecognizer();
tapGestureRecognizer.Tapped += OnChipTapped;
GestureRecognizers.Add(tapGestureRecognizer);

UpdateVisualState();
```

Рис. 4.20 Bindable Properties в класі CustomChip

Методи OnTextChanged, OnIsSelectedChanged та OnIconSourceChanged викликаються автоматично при зміні відповідних властивостей, що можна побачити на рисунку 4.21. Вони оновлюють відповідні внутрішні елементи компонента. Тобто OnTextChanged оновлює текст у мітці, а OnIconSourceChanged змінює джерело зображення іконки.

```
1 usage  Ridcovets Sergo
private static void OnTextChanged(BindableObject bindable, object oldValue, object newValue)
{
    var control = (CustomChip)bindable;
    control._label.Text = newValue?.ToString();
}

1 usage  Ridcovets Sergo
private static void OnIsSelectedChanged(BindableObject bindable, object oldValue, object newValue)
{
    var control = (CustomChip)bindable;
    control.UpdateVisualState();
}

1 usage  Ridcovets Sergo
private static void OnIconSourceChanged(BindableObject bindable, object oldValue, object newValue)
{
    var control = (CustomChip)bindable;
    control._checkImage.Source = (string)newValue;
}
```

Рис. 4.21 Методи для змін в класі CustomChip

Завдяки використанню системи прив'язування даних MAUI, CustomChip легко інтегрується з архітектурою MVVM додатку. Властивість IsSelected підтримує двосторонню прив'язку, що дозволяє як оновлювати стан вибору з ViewModel, так і реагувати на зміни стану, що відбуваються в результаті взаємодії користувача з компонентом.

Цей кастомний елемент інтерфейсу надає унікальний та послідовний спосіб для відображення вибраних опцій або фільтрів у всьому додатку, покращуючи загальний досвід користувача через візуально зрозумілі та інтерактивні компоненти.

Далі йде реалізація роботи з графіком. Було реалізовано систему візуалізації даних для аналізу результатів оцінювань через клас DiagramsGroupPageViewModel, який відповідає за відображення графіка часової шкали. Ця функціональність базується на бібліотеці LiveCharts для побудови інтерактивних графіків.

Основна візуалізація даних представлена лінійним графіком, який відображає фактичні та прогнозовані значення часових витрат для вибраних опитувань. Графік конфігурується через певні властивості, одна з яких містить набір даних для їх відображення на діаграмі.

Для налаштування осей графіка використовуються властивості XAxes та YAxes, зображених на рисунку 4.22. Осі мають спеціальні налаштування для відображення перехресної лінії з відповідним форматуванням та кольорами. В осі Y для властивості CrosshairSnapEnabled встановлено значення true, що дозволяє перехресній лінії прив'язуватися до найближчої точки даних.

```
1 usage
public ICartesianAxis[] XAxes { get; set; } = [
    new Axis
    {
        CrosshairLabelsBackground = SKColors.DarkOrange.AsLvcColor(),
        CrosshairLabelsPaint = new SolidColorPaint(SKColors.DarkRed),
        CrosshairPaint = new SolidColorPaint(SKColors.DarkOrange, 1),
        Labeler = value :double => value.ToString( format: "N0")
    }
];

1 usage
public ICartesianAxis[] YAxes { get; set; } = [
    new Axis
    {
        CrosshairLabelsBackground = SKColors.DarkOrange.AsLvcColor(),
        CrosshairLabelsPaint = new SolidColorPaint(SKColors.DarkRed),
        CrosshairPaint = new SolidColorPaint(SKColors.DarkOrange, 1),
        Labeler = value :double => value.ToString( format: "N1"),
        CrosshairSnapEnabled = true
    }
];
```

Рис. 4.22 Налаштування осей графіка

В класі присутній метод LoadSurveyData, позначений атрибутом [RelayCommand], який відповідає за асинхронне завантаження даних з сервера, який можна побачити на рисунку 4.23. При виконанні цього методу відображається індикатор завантаження в інтерфейсі. Після отримання даних через GetSurveyGroupResultsTimeline, метод обробляє їх і створює колекцію SurveyCheckBoxItems, яка містить інформацію про кожне опитування з доданим елементом “Загальні дані”, який обирається за замовчуванням.

```
public async Task LoadSurveyData()
    try
        foreach (var surveyItem :SurveyTimeline in _surveyResults.Statistics)
            SurveyCheckBoxes.Add(new SurveyCheckBoxItem
                Name = TruncateName(surveyItem.Survey.SurveyName, maxLength: 20),
                IsSelected = false,
                Statistics = surveyItem.Timeline,
            });
        }

        SurveyCheckBoxes.Add(new SurveyCheckBoxItem
        {
            Name = TruncateName("Загальні дані", maxLength: 20),
            IsSelected = false,
            Statistics = _surveyResults.GlobalTimeline,
        });

        SurveyCheckBoxes.Last().IsSelected = true;

        _categorizationResults = await _statisticsClient.GetSurveyGroupResultsCategorization(GroupId);

        DistributionItems.Clear();

        DistributionItems.Add(new DistributionPickerItem
        {
            Name = "Загальні дані",
            Distribution = _categorizationResults.Global.Distribution,
            IsSelected = true
        });
```

Рис. 4.23 Метод LoadSurveyData

Наступний метод UpdateChart є ключовим для оновлення графіка на основі вибраних опитувань, що зображено на рисунку 4.24. Він отримує вибрані опитування за допомогою фільтрації колекції SurveyCheckBoxes за властивістю IsSelected і для кожного з них створює дві серії даних:

- серія для фактичних даних з суцільною лінією, маркерами у вигляді геометричних фігур та заливкою напівпрозорим кольором.
- серія для прогнозованих даних з пунктирною лінією без маркерів і без заливки.

```
private void UpdateChart()
    foreach (var surveyStatistic SurveyCheckBoxItem in surveyStatistics)
            new LineSeries<float>
            {
                Values = actualPoints,
                Name = $"{surveyName}: Факт",
                Stroke = new SolidColorPaint()
                {
                    Color = currentColor,
                    StrokeThickness = 2
                },
                Fill = new SolidColorPaint(currentColor.WithAlpha(80)),
                GeometrySize = 8,
                GeometryStroke = new SolidColorPaint(currentColor, 2),
                GeometryFill = new SolidColorPaint(SKColors. ■ White)
            });

        lineSeries.Add(
            new LineSeries<float>
            {
                Values = predictedPoints,
                Name = $"{surveyName}: Прогноз",
                Stroke = new SolidColorPaint()
                {
                    Color = currentColor,
                    StrokeThickness = 1,
                    PathEffect = effect,
                },
                Fill = new SolidColorPaint(SKColors.   Transparent),
                GeometrySize = 0
```

Рис. 4.24 Метод UpdateChart

Клас SurveyCheckBoxItem використовується для представлення окремого оцінювання в списку вибору, зображеного на рисунку 4.25. Він містить назву опитування, стан вибору та дані статистики часової шкали.

```
13 usages  Ridcovets Sergo
public partial class SurveyCheckBoxItem : ObservableObject
{
    [ObservableProperty]
    private string _name;

    [ObservableProperty]
    private bool _isSelected;

    [ObservableProperty]
    private Timeline _statistics;
}
```

Рис. 4.25 Клас SurveyCheckBoxItem

Графік дозволяє користувачам інтерактивно вибирати за допомогою кастомних chip, які оцінювання відображати, та візуально порівнювати фактичні дані з прогнозованими. Таке представлення даних дає змогу ефективно аналізувати відхилення у часових витратах для різні оцінювання, що особливо корисно для адміністраторів, які відстежують прогрес та навантаження студентів.

І остання цікава функціональність це імпорт даних з силабусу. Було створено клас ImportSyllabusPageViewModel, що відповідає за обробку вибору та завантаження PDF-файлу силабусу в застосунку.

Спочатку користувач обирає PDF-файл за допомогою методу PickAndImportFile, продемонстрованого на рисунку 4.26. Цей метод використовує FilePicker для відображення діалогового вікна вибору файлу з обмеженням на тип файлу. Після обирання файлу, оновлюються властивості FileName та IsFileSelected, які відображаються в інтерфейсі користувача.

```
[RelayCommand]
2 usages  Ridcovets Sergo
private async Task PickAndImportFile()
{
    try
    {
        _selectedFile = await FilePicker.PickAsync(new PickOptions
        {
            PickerTitle = "Оберіть PDF файл",
            FileTypes = FilePickerFileType.Pdf
        });

        if (_selectedFile != null)
        {
            FileName = _selectedFile.FileName;
            IsFileSelected = true;
        }
        else
        {
            FileName = "Файл не обрано";
            IsFileSelected = false;
        }
    }
    catch (Exception ex)
    {
        await Application.Current.MainPage.DisplayAlert( title: "Помилка",  message: $"Помилка при виборі файлу: {ex.Message}",
    }
}
```

Рис. 4.26 Метод PickAndImportFile

Після вибору файлу користувач може перейти до процесу імпорту завдяки методу ImportSyllabus, зображеного на рисунку 4.27. Спочатку він перевіряє наявність обраного файлу, якщо файл обрано, то метод виконує наступне: відкриває потік для читання файлу, копіює дані в масив байтів, створює об'єкт ByteArrayPart з даними файлу та передає його серверу через команду ImportSyllabus з інтерфейсу IImportClient.

```
[RelayCommand]
2 usages  Ridcovets Sergo
private async Task ImportSyllabus()
{
    if (_selectedFile == null)
    {
        await Application.Current.MainPage.DisplayAlert( title: "Помилка",  message: "Будь ласка, оберіть файл перед імпортом",
        return;
    }

    try
    {
        using var stream = await _selectedFile.OpenReadAsync();
        byte[] fileBytes;
        using (var memoryStream = new MemoryStream())
        {
            stream.CopyTo(memoryStream);
            fileBytes = memoryStream.ToArray();
        }

        var fileToSend = new ByteArrayPart(fileBytes, _selectedFile.FileName,  contentType: "application/pdf");

        await _importClient.ImportSyllabus(fileToSend);

        await Application.Current.MainPage.DisplayAlert( title: "Успіх",  message: "Сілабус успішно імпортовано",  cancel: "OK");

        await Shell.Current.GoToAsync( state:  "..");
    }
```

Рис. 4.27 Метод ImportSyllabus

Важливим компонентом є використання ByteArrayPart, що є частиною бібліотеки Refit, для створення відповідного формату даних для передачі на сервер. Цей об'єкт оброблює бінарні дані файлу разом з метаданими, що дозволяє серверу правильно інтерпретувати отримані дані.

В результаті було створено сторінки для імпорту даних з силабусу, які можна побачити на рисунках 4.28 та 4.29. А також було створено сторінки відображення результатів через графік обраного оцінювання, що продемонстровано на рисунках 4.30 та 4.31.

Рис. 4.28 Сторінка імпорту

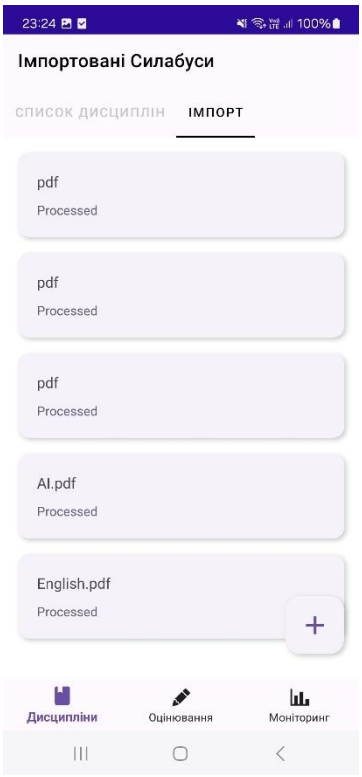


Рис. 4.29 Сторінка імпортованих файлів

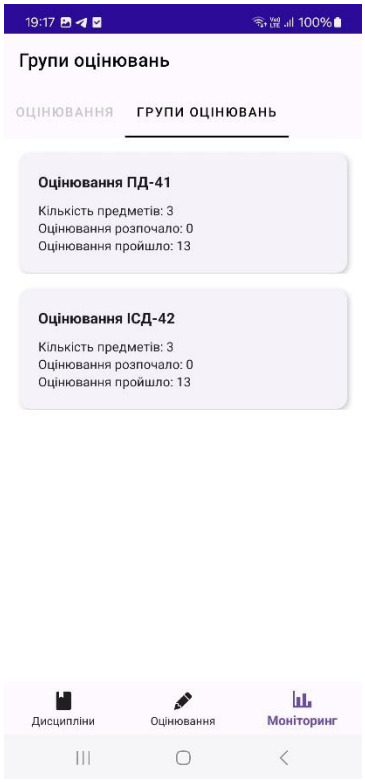


Рис. 4.30 Групи оцінювань

Рис. 4.31 Діаграми групи оцінювань

4.4 Тестування застосунків

Тестування є важливою складовою процесу розробки програмного забезпечення, оскільки дозволяє виявити помилки, перевірити відповідність функціоналу вимогам та забезпечити стабільну роботу застосунку. Особливу роль у цьому процесі відіграють тестові сценарії, які описують послідовність дій користувача та очікувану реакцію системи. Вони дають змогу системно перевірити основні функції програми, забезпечують відтворюваність тестування та підвищують якість проєктів. В результаті було створено тестові сценарії, розроблені для перевірки ключових можливостей застосунку, з метою підтвердження його коректної роботи та відповідності поставленим вимогам. Тестовий сценарій являє собою опис дій користувача, що імітує типову ситуацію використання застосунку. Він подається стисло, здебільшого в одному реченні, та охоплює основну функціональність з погляду кінцевого користувача [20]. Переглянути ці тестові сценарії можна в таблиці 4.1.

Таблиця 4.1

Тестові сценарії для тестування застосунків TaskMon

№ сценарію	**Назва сценарію**	**Кроки тестування**	**Очікувані результати**	**Фактичні результати**
1	Авторизація та перенаправлення студента	1. Запустити застосунок 2. Натиснути кнопку авторизації 3. Підтвердити вхід	1. Користувач успішно авторизований 2. Перевіряється наявність останньої відкритої групи	1. Користувач успішно авторизований 2. Перевіряється наявність останньої відкритої групи
2	Перехід за посиланням на групу оцінювань	1. Відкрити посилання 2. Обрати застосунок для студентів 3. Авторизуватися	1. Застосунок зберігає ID групи 2. На сторінці відображаються оцінювання групи з посилання	1. Застосунок зберігає ID групи 2. На сторінці відображаються оцінювання групи з посилання

Продовження таблиці 4.1

Тестові сценарії для тестування застосунків TaskMon

№ сценарію	**Назва сценарію**	**Кроки тестування**	**Очікувані результати**	**Фактичні результати**
3	Перегляд та заповнення оцінювання	1. Натиснути на доступне оцінювання 2. Натиснути кнопку “Відправити”	1. Результати відправляються 2. Оцінювання отримує статус “Пройдено”	1. Результати відправляються 2. Оцінювання отримує статус “Пройдено”
4	Вихід з облікового запису	Натиснути кнопку виходу з облікового запису	Користувач перенаправляється на сторінку авторизації	Користувач перенаправляється на сторінку авторизації
5	Імпорт силабусу	1. Перейти до розділу імпорту 2. Натиснути кнопку для імпортування 3. Обрати файл для імпорту 4. Натиснути кнопку “Імпортувати”	1. Файл надсилається на сервер 2. Файл з’являється в розділі імпорту 3. Після імпортування файл має статус “Processed”	1. Файл надсилається на сервер 2. Файл з’являється в розділі імпорту 3. Після імпортування файл має статус “Processed”
6	Зміна даних відображення в діаграмі	1. Перейти на вкладку моніторингу 2. Натиснути на групу оцінювань 3. Обрати декілька міток	1. Діаграма оновлюється при виборі мітки 2. Діаграма відображає одночасно декілька міток	1. Діаграма оновлюється при виборі мітки 2. Діаграма відображає одночасно декілька міток
7	Створення нової дисципліни	1. Перейти на вкладку дисциплін 2. Натиснути кнопку створення дисципліни 3. Створити дисципліну	1. Дисципліна успішно створюється 2. Створена дисципліна додалася до списку дисциплін	1. Дисципліна успішно створюється 2. Створена дисципліна додалася до списку дисциплін

ВИСНОВКИ

Під час роботи над проєктом було проведено дослідження проблеми, аналіз наявних рішень та реалізовано два застосунка для моніторингу навчального навантаження студентів. Було враховано як потреби кінцевих користувачів, так і сучасні підходи до розробки програмного забезпечення. В результаті було виконано наступні задачі:

1. Проаналізовано наявні рішення (Google Forms, SurveyMonkey, Moodle, Canvas) для моніторингу навчального навантаження студентами та знайдено основні недоліки. Виявилося, що основним недоліком переважно є відсутність деяких функцій, наявність яких спростила б моніторинг навчального навантаження студентів. Це дозволило виявити основний функціонал для розробки власних застосунків.

2. Визначено основні функціональні вимоги та нефункціональні вимоги для розробки обох застосунків. До основних функціональних вимог належать: можливість створювати оцінювання, проходження оцінювань студентами, відображення результатів графіками та імпорт даних з силабусу. Серед нефункціональних вимог основною є дотримування інтерфейсу з Material Design 3.

3. Досліджено сучасні засоби реалізації та обґрунтовано їх вибір. Було проаналізовано інструменти для клієнтської частини застосунку, серед яких основними є фреймворк .NET MAUI та середовище розробки Rider. Фреймворк добре продемонстрував себе як основа для мобільного застосунку, а середовище розробки дозволили комфортніше працювати з ним.

4. Розроблено застосунок студентів для проходження оцінювань. В ньому передбачено авторизацію користувача, відображення оцінювань при переході за посиланням та можливість проходити активні оцінювання. Завдяки цьому було реалізовано інструмент для збору інформації стосовно навчального навантаження.

5. Розроблено застосунок адміністраторів для створення та перегляду результатів оцінювання. В ньому можна створювати дисципліни, створювати силабус вручну та імпортувати його, створювати групу оцінювань вибираючи наявні дисципліни, переглядати результати за допомогою графіків на надсилати посилання на проходження оцінювань. Цей інструмент допоможе збирати інформацію про навантаження студентів, щоб допомогти формулювати нові навчальні плани на майбутнє.

6. Протестовано обидва застосунка для перевірки взаємодії клієнта з сервером. Для цього було створені тестові сценарії, за якими тестувався застосунок, в результаті яких усі сценарії відповідали очікуваним результатам. Це підтвердило відповідність застосунків заявленим вимогам, через що тепер їх можна використовувати в реальному навчальному процесі.

Кваліфікаційна робота пройшла апробацію на двох конференціях та було опубліковано наступні дві тези:

1. Рідковець С.В., Замрій. І.В. Переваги фреймворку .NET MAUI в створенні додатків. VI Міжнародна науково-технічна конференція «Сучасний стан та перспективи розвитку IoT», 15 квітня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій (подано до друку).

2. Рідковець С.В., Замрій І.В. Організація архітектури додатку за допомогою шаблону MVVM на прикладі фреймворку .NET MAUI. VI Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ», 24 квітня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С.125–127.

ПЕРЕЛІК ПОСИЛАНЬ

1. Батечко Н., Панталієнко Л. Силабуси навчальних дисциплін: сучасні підходи до формування змісту підготовки фахівців інженерних спеціальностей. *Освітологічний дискурс*. 2020. URL: https://www.od.kubg.edu.ua/index.php/journal/article/view/747/602.

2. Кириленко О. І. Визначення навчального навантаження студентів за допомогою план-форм. *Збірник наукових праць Міжнародної науково-практичної конференції «Сучасна освіта та наука: проблеми, перспективи, інновації»*. 2021. URL: https://enpuir.npu.edu.ua/bitstream/handle/123456789/34764/Kyrylenko.pdf

3. Прокопенко О. А. Емоційне виснаження студентів та втома під час війни. *Проблеми сучасних трансформацій. Серія: педагогіка та психологія*. 2025. Т. 7. URL: https://www.reicst.com.ua/pmtp/article/view/2025-7-10-01/2025-7-10-01.

4. Well-being and academic workload: Perceptions of Science and technology students / K. Yangdon та ін. *Educational Research and Reviews*. 2021. Т. 16. URL: https://academicjournals.org/journal/ERR/article-full-text-pdf/316F97B68217.

5. Google Workspace. Google Forms. URL: https://workspace.google.com/products/forms.

6. SurveyMonkey. URL: https://www.surveymonkey.com.

7. Moodle. URL: https://moodle.com.

8. Instructure. Canvas. URL: https://www.instructure.com/canvas.

9. Figma. Figma Design. URL: https://www.figma.com/design.

10. Learn Microsoft. .NET MAUI. URL: https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui?view=net-maui-9.0.

11. Stonis M. Enterprise Application Patterns Using .NET MAUI. Microsoft Corporation, 2022. 99 с. URL: https://learn.microsoft.com/ru-ru/dotnet/architecture/maui.

12. Learn Microsoft. Tour of C#. URL: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/overview.

13. Comparative Study of C++ and C# Programming Languages / V. V. Ponggawa та ін. *Jurnal Syntax Admiration*. 2024. Т. 5. URL: https://www.jurnalsyntaxadmiration.com/index.php/jurnal/article/view/1926/1961.

14. JetBrains. Rider. URL: https://www.jetbrains.com/rider.

15. Learn Microsoft. MVVM Toolkit Introduction. URL: https://learn.microsoft.com/uk-ua/dotnet/communitytoolkit/mvvm.

16. EnisnProjects. Uranium UI. URL: https://enisn-projects.io/docs/en/uranium.

17. Reactive UI. Refit. URL: https://reactiveui.github.io/refit.

18. Auth0. URL: https://auth0.com.

19. LiveCharts2. URL: https://livecharts.dev.

20. PractiTest. Test Scenario vs Test Case. URL: https://www.practitest.com/resource-center/article/test-scenario-vs-test-case.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка застосунку для моніторингу навчального навантаження студентів. Спец-частина: Розробка Frontend-частини застосунку

Виконав студент 4 курсу
групи ПД-41
Рідковець Сергій Вікторович
Керівник роботи
д.т.н., проф., завідувач кафедри ІПЗ Замрій Ірина Вікторівна

Київ – 2025

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи:** спрощення моніторингу навчального навантаження студентів на основі використання програмного забезпечення для вирішення проблеми збору даних про завантаженість студентів.
- **Об'єкт дослідження:** процес збору даних та моніторингу навчального навантаження студентів у закладах вищої освіти.
- **Предмет дослідження:** програмне забезпечення для оцінювання та моніторингу навчального навантаження студентів.

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Проаналізувати наявні рішення для моніторингу навчального навантаження студентів.
2. Визначити функціональні та нефункціональні вимоги до застосунків.
3. Дослідити засоби реалізації, які можуть бути використані при розробці застосунків.
4. Розробити застосунок студентів для проходження оцінювання з можливістю вписувати витрачений час на дисципліну та відправляти результати.
5. Розробити застосунок адміністраторів для створювання та перегляду результатів оцінювань з можливістю імпорту даних з документу про силабус.
6. Провести тестування розроблених застосунків.

3

АНАЛІЗ АНАЛОГІВ

Показник	**Google Forms**	**SurveyMonkey**	**Moodle (Course Dedication)**	**Canvas (Student Access Data)**	**TaskMon**
Форма збору даних про навантаження	Опитування студентів	Опитування студентів	На основі взаємодії з курсом	На основі взаємодії з курсом	Оцінювання студентів
Порівняння теоретичного навантаження з реальним	-	-	-	-	+
Візуалізація аналітики	Графіки, діаграми	Графіки, діаграми	Таблиця	Графіки, діаграми	Графіки, діаграми
Генерація посилань на оцінювання	+	+	-	-	+
Імпорт даних з силабусу	-	-	-	-	+

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:

1. Внесення даних про програму навчальної дисципліни.
2. Проведення оцінювання витраченого часу на вивчення дисципліни та окремих частин предмета студентами.
3. Створення оцінювань як для однієї дисципліни, так і для групи з дисциплінами.
4. Порівняння очікуваного та реального навантаження студентів.
5. Вивід інформації про кількість перенавантажених та недовантажених студентів.
6. Імпортування даних з документу про силабус.
7. Створення посилань для проходження оцінювань.

Нефункціональні вимоги:

1. Завантаження сторінок не повинно перевищувати 3 секунди.
2. Інтерфейс користувача має відповідати принципам Material Design 3.

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Rider

Uranium UI

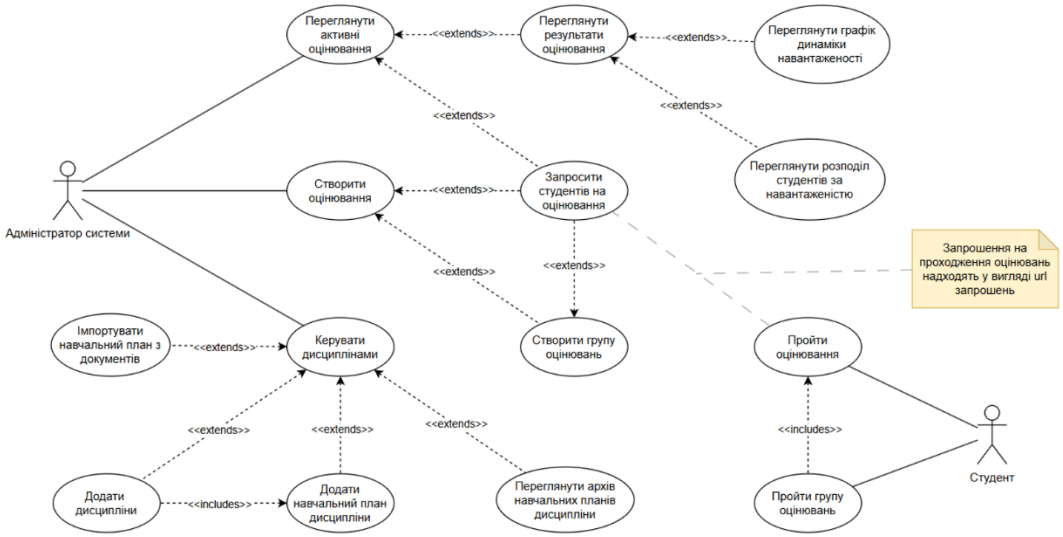
C#

Refit

MVVM Toolkit

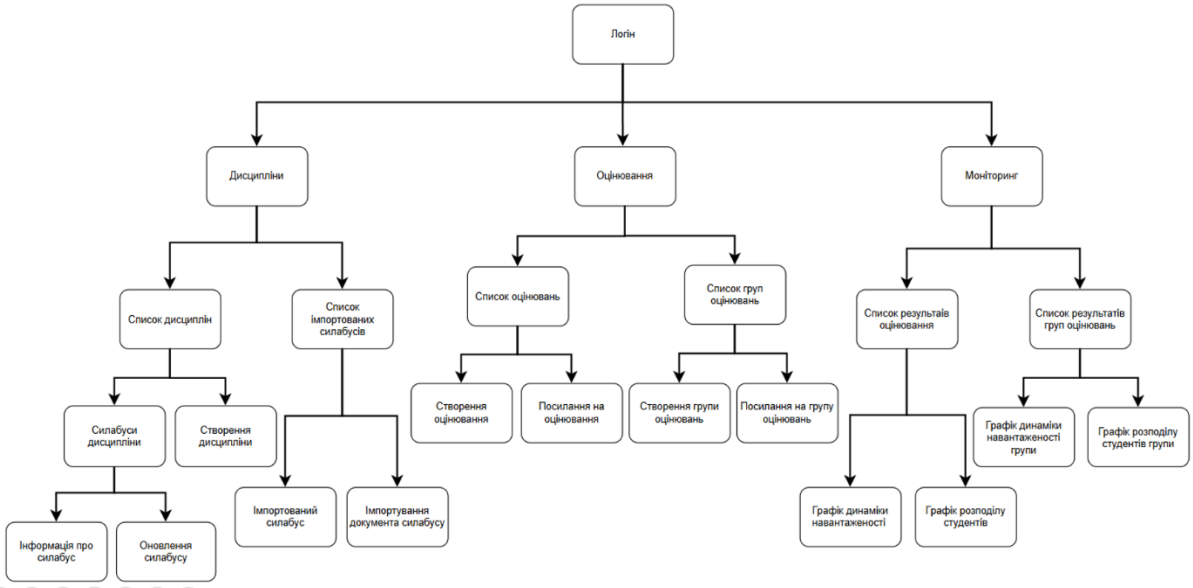
6

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



7

МАПА ЗАСТОСУНКУ АДМІНІСТРАТОРА

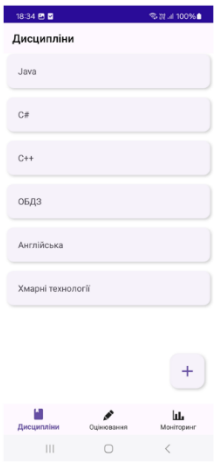


8

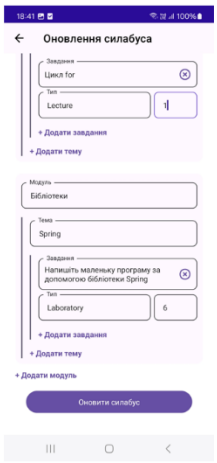
МАПА ЗАСТОСУНКУ СТУДЕНТА

9

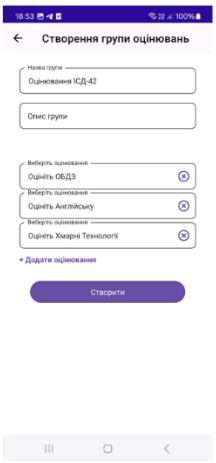
ЕКРАННІ ФОРМИ



Сторінка дисциплін (адміністратор)



Сторінка створення силабуса (адміністратор)

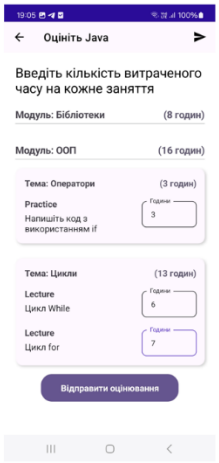


Сторінка створення групи оцінювань (адміністратор)

Сторінка груп оцінювань (адміністратор)

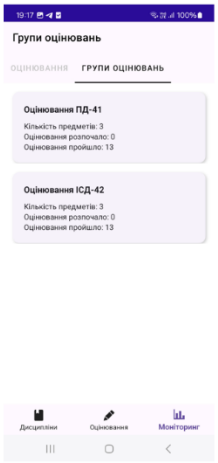
10

ЕКРАННІ ФОРМИ

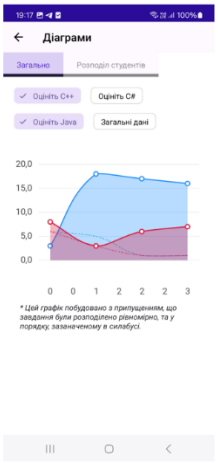


Сторінка проходження оцінювання (студент)

Сторінка проходження групи оцінювань (студент)



Сторінка моніторингу груп оцінювань (адміністратор)



Сторінка результатів групи оцінювань (адміністратор)

11

ЕКРАННІ ФОРМИ

Сторінка імпортування силабусу (адміністратор)

Сторінка імпортованих силабусів (адміністратор)

Сторінка імпортованого силабусу (адміністратор)

12

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Рідковець С.В., Замрій. І.В. Переваги фреймворку .NET MAUI в створенні додатків. VI Міжнародна науково-технічна конференція «Сучасний стан та перспективи розвитку IoT», 15 квітня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій (подано до друку).
2. Рідковець С.В., Замрій І.В. Організація архітектури додатку за допомогою шаблону MVVM на прикладі фреймворку .NET MAUI. VI Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в ІКТ», 24 квітня 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С.125–127.

13

ВИСНОВКИ

1. Проаналізовано наявні рішення для моніторингу навчального навантаження студентами (Google Forms, SurveyMonkey, Moodle, Canvas), що дозволило виявити основний функціонал для розробки власного застосунку.
2. Визначено основні функціональні та нефункціональні вимоги для розробки обох застосунків, серед яких основними є: можливість створення, проходження та перегляду результатів оцінювань.
3. Досліджено сучасні засоби реалізації застосунків (.NET MAUI, Rider) та обґрунтовано їх вибір для розробки.
4. Розроблено застосунок студентів для проходження оцінювань, що дає змогу збирати інформацію про навантаження.
5. Розроблено застосунок адміністраторів для створення та перегляду результатів оцінювань, що допоможе сформулювати висновки про навантаження студентів.
6. Протестовано обидва застосунки, що підтвердило стабільний зв’язок між клієнтом і сервером.

14

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

MauiProgram.cs (student app):

```
public static class MauiProgram
{
   public static MauiApp CreateMauiApp()
   {
      var builder = MauiApp.CreateBuilder();
      builder
         .UseMauiApp<App>()
         .UseUraniumUI()
         .UseUraniumUIMaterial()
         .ConfigureFonts(fonts =>
         {
            fonts.AddFont("OpenSans-
Regular.ttf", "OpenSansRegular");
            fonts.AddFont("OpenSans-
Semibold.ttf", "OpenSansSemibold");
         });


builder.Services.AddSingleton<ISecureStorag
e>(SecureStorage.Default);


builder.Services.AddSingleton<AuthDelegati
ngHandler>();

        builder.Services.AddSingleton(new
Auth0Client(new Auth0ClientOptions
        {
            Domain = "tasked-
app.eu.auth0.com",
            ClientId =
"RKGbGFMYFlLDBa0u19xYRqWWvpOq5
Gg5",


            RedirectUri = "taskmon://callback/",
            PostLogoutRedirectUri =
"taskmon://callback/",
            Scope = "openid profile"
        }));


builder.Services.AddSingleton<Auth0Service
>();

builder.Services.AddSingleton<ITokenSuppli
er, TokenSupplier>();


builder.Services.AddRefitClient<ISurveyClie
nt>()
          .ConfigureHttpClient(c => c
             .BaseAddress = new
Uri("https://surveysmock-
cwchemf4gtgbgcfz.polandcentral-
01.azurewebsites.net"))

.AddHttpMessageHandler<AuthDelegatingH
andler>();


builder.Services.AddSingleton<LoadingPage
>();

builder.Services.AddSingleton<LoginPage>()
;

builder.Services.AddSingleton<SurveyPage>(
);

builder.Services.AddSingleton<SurveyGroup
Page>();

#if DEBUG
      builder.Logging.AddDebug();
#endif

      return builder.Build();
   }
}
```

MainActivity.cs (student app):

```
[Activity(Theme =
"@style/Maui.SplashTheme", MainLauncher
= true, LaunchMode =
LaunchMode.SingleTop,
   ConfigurationChanges =
```

```
ConfigChanges.ScreenSize |
ConfigChanges.Orientation |
ConfigChanges.UiMode |

ConfigChanges.ScreenLayout |
ConfigChanges.SmallestScreenSize |
ConfigChanges.Density)]


[IntentFilter(new[] { Intent.ActionView },
   Categories = new[] {
Intent.CategoryDefault,
Intent.CategoryBrowsable },
   DataScheme = "http",
   DataHost = "taskmon.com",
   DataPathPrefix = "/surveys/invite" )]

[IntentFilter(new[] { Intent.ActionView },
   Categories = new[] {
Intent.CategoryDefault,
Intent.CategoryBrowsable },
   DataScheme = "http",
   DataHost = "taskmon.com",
   DataPathPrefix = "/groups/invite" )]

[IntentFilter(new[] { Intent.ActionView },
   Categories = new[] {
Intent.CategoryDefault,
Intent.CategoryBrowsable },
   DataScheme = "https",
   DataHost = "taskmon.com",
   DataPathPrefix = "/surveys/invite" )]

[IntentFilter(new[] { Intent.ActionView },
   Categories = new[] {
Intent.CategoryDefault,
Intent.CategoryBrowsable },
   DataScheme = "https",
   DataHost = "taskmon.com",
   DataPathPrefix = "/groups/invite" )]
public class MainActivity :
MauiAppCompatActivity
{
   protected override void OnCreate(Bundle
savedInstanceState)
   {
      base.OnCreate(savedInstanceState);
      ProcessIntent(Intent);
   }

   protected override void
OnNewIntent(Intent intent)
   {
      base.OnNewIntent(intent);
      ProcessIntent(intent);
   }

   private void ProcessIntent(Intent intent)
   {
      var uri = intent?.Data;
      if (uri != null)
      {
         string path = uri.Path;
         if (!string.IsNullOrEmpty(path))
         {
            if
(path.StartsWith("/surveys/invite/"))
            {
               string surveyId =
path.Replace("/surveys/invite/", "");
               if (Guid.TryParse(surveyId, out
_))
               {

Preferences.Set("PendingDeepLinkType",
(int)DeepLinkType.Survey);

Preferences.Set("PendingDeepLinkId",
surveyId);

Preferences.Set("HasPendingDeepLink",
true);


WeakReferenceMessenger.Default.Send(new
DeepLinkMessage(DeepLinkType.Survey,
surveyId));
               }
               else
               {

WeakReferenceMessenger.Default.Send(new
DeepLinkMessage(DeepLinkType.Survey,
string.Empty));
               }
            }
```

```
            else if
(path.StartsWith("/groups/invite/"))
            {
                string groupId =
path.Replace("/groups/invite/", "");
                if (Guid.TryParse(groupId, out
_))
                {

Preferences.Set("PendingDeepLinkType",
(int)DeepLinkType.Group);

Preferences.Set("PendingDeepLinkId",
groupId);

Preferences.Set("HasPendingDeepLink",
true);


WeakReferenceMessenger.Default.Send(new
DeepLinkMessage(DeepLinkType.Group,
groupId));
                }
                else
                {

WeakReferenceMessenger.Default.Send(new
DeepLinkMessage(DeepLinkType.Group,
string.Empty));
                }
            }
        }
    }
  }
}
```

Auth0Service.cs (student app):

```
namespace TaskMonMobile.Services
{
  public class Auth0Service
  {
    private readonly Auth0Client
_auth0Client;
    private readonly
ILogger<Auth0Service> _logger;
    private readonly ISecureStorage
_secureStorage;
    private const string AccessTokenKey =
"auth0_access_token";

    public Auth0Service(Auth0Client
auth0Client, ISecureStorage secureStorage,
ILogger<Auth0Service> logger)
    {
      _auth0Client = auth0Client;
      _secureStorage = secureStorage;
      _logger = logger;
    }

    public async Task<bool> LoginAsync()
    {
      try
      {
        var loginResult = await
_auth0Client.LoginAsync(new
        {
          audience = "https://taskmon.com"
        });

        if (loginResult.IsError)
        {
          _logger.LogError($"Не вийшло
залогінитись: {loginResult.Error}");
          return false;
        }

        await
_secureStorage.SetAsync(AccessTokenKey,
loginResult.AccessToken);

        return true;
      }
      catch (Exception ex)
      {
        _logger.LogError(ex, "Помилка
під час логіну");
        return false;
      }
    }

    public async Task<bool> LogoutAsync()
    {
      try
      {
```

```
_secureStorage.Remove(AccessTokenKey);

            await _auth0Client.LogoutAsync();
            return true;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Помилка
під час виходу з акаунту");
            return false;
        }
    }

    public async Task<string?>
GetAccessTokenAsync()
    {
        return await
_secureStorage.GetAsync(AccessTokenKey);
    }

    public async Task<bool>
IsAuthenticatedAsync()
    {
        var token = await
GetAccessTokenAsync();
        return !string.IsNullOrEmpty(token);
    }
  }
}
```

SurveyGroupPageViewModel.cs (student app):

```
namespace TaskMonMobile.ViewModels
{
  public partial class
SurveyGroupPageViewModel :
ObservableObject
  {
    private readonly ISurveyClient
_surveyClient;

    [ObservableProperty]
    private Guid _id;

    [ObservableProperty]
    private Guid _groupId;

    [ObservableProperty]
    private string _title = "Опитування";

    [ObservableProperty]
    private
ObservableCollection<SurveyItemViewMode
l> _surveys;

    [ObservableProperty]
    private bool _hasNoSurveys;

    [ObservableProperty]
    private string _noSurveysMessage = "У
вас поки немає опитувань";

    [ObservableProperty]
    private bool _isLoading;

    [ObservableProperty]
    private bool _isRefreshing;

    public ICommand
NavigateToSurveyCommand { get; }

    public
SurveyGroupPageViewModel(ISurveyClient
surveyClient)
    {
        _surveyClient = surveyClient;
        Surveys = new
ObservableCollection<SurveyItemViewMode
l>();
        NavigateToSurveyCommand = new
Command<Guid>(async (surveyId) => await
NavigateToSurvey(surveyId));
        HasNoSurveys = true;
    }

    [RelayCommand]
    private async Task RefreshDataAsync()
    {
        IsRefreshing = true;
        try
        {
            await
LoadSurveyGroupDataAsync();
        }
        finally
```

```
        {
            IsRefreshing = false;
        }
    }

    public async Task
LoadSurveyGroupDataAsync()
    {
        try
        {
            IsLoading = true;
            var surveyGroup = await
_surveyClient.GetSurveyGroupAsync(GroupI
d);
            LoadFromModel(surveyGroup);

            HasNoSurveys = Surveys.Count ==
0;

            if (HasNoSurveys ||
string.IsNullOrEmpty(Title))
            {
                Title = "Опитування";
            }

            LoadCompletedSurveyStatus();
        }
        catch (Exception ex)
        {
            HasNoSurveys = true;
            NoSurveysMessage = "Не вірне
посилання, спробуйте знову";
            Title = "Опитування";
            throw;
        }
        finally
        {
            IsLoading = false;
        }
    }

    private void
LoadFromModel(SurveyGroup surveyGroup)
    {
        Id = surveyGroup.Id;

        Title =
!string.IsNullOrEmpty(surveyGroup.Title) ?
surveyGroup.Title : "Опитування";

        Surveys.Clear();

        foreach (var survey in
surveyGroup.Surveys)
        {
            Surveys.Add(new
SurveyItemViewModel
            {
                Id = survey.Id,
                Title = survey.Title,
                ParentViewModel = this
            });
        }
    }

    private void
LoadCompletedSurveyStatus()
    {
        foreach (var survey in Surveys)
        {
            string key =
$"CompletedSurvey_{GroupId}_{survey.Id}
";
            bool isCompleted =
Preferences.Get(key, false);
            survey.IsCompleted = isCompleted;
        }
    }

    public void
MarkSurveyAsCompleted(Guid surveyId)
    {
        var survey = Surveys.FirstOrDefault(s
=> s.Id == surveyId);
        if (survey != null)
        {
            survey.IsCompleted = true;
            string key =
$"CompletedSurvey_{GroupId}_{surveyId}"
;
            Preferences.Set(key, true);
        }
    }

    private async Task
NavigateToSurvey(Guid surveyId)
```

```
        {
            await
Shell.Current.GoToAsync($"SurveyPage?sur
veyId={surveyId}&groupId={GroupId}");
        }
    }
}
```

SurveyItemViewModel.cs (student app):

```
namespace TaskMonMobile.ViewModels
{
    public partial class SurveyItemViewModel
: ObservableObject
    {
        [ObservableProperty]
        private Guid _id;

        [ObservableProperty]
        private string _title;

        [ObservableProperty]
        private bool _isCompleted;

        public SurveyGroupPageViewModel
ParentViewModel { get; set; }

        public string CompletionStatus =>
IsCompleted ? "Пройдено" : "Не пройдено";

        public ICommand
SelectSurveyCommand => new Command(()
=>
        {
            if (!IsCompleted)
            {

ParentViewModel.NavigateToSurveyComma
nd.Execute(Id);
            }
        });
    }
}
```

CustomChip.cs (admin app):

```
public class CustomChip : ContentView
{
    public static readonly BindableProperty
TextProperty = BindableProperty.Create(
        nameof(Text), typeof(string),
typeof(CustomChip), string.Empty,
propertyChanged: OnTextChanged);

    public static readonly BindableProperty
IsSelectedProperty =
BindableProperty.Create(
        nameof(IsSelected), typeof(bool),
typeof(CustomChip), false,
BindingMode.TwoWay, propertyChanged:
OnIsSelectedChanged);

    public static readonly BindableProperty
IconSourceProperty =
BindableProperty.Create(
        nameof(IconSource), typeof(string),
typeof(CustomChip), "check.png",
propertyChanged: OnIconSourceChanged);

    public event EventHandler<bool>
SelectedChanged;

    public string Text
    {
        get => (string)GetValue(TextProperty);
        set => SetValue(TextProperty, value);
    }

    public bool IsSelected
    {
        get =>
(bool)GetValue(IsSelectedProperty);
        set => SetValue(IsSelectedProperty,
value);
    }

    public string IconSource
    {
        get =>
(string)GetValue(IconSourceProperty);
        set => SetValue(IconSourceProperty,
value);
    }

    private readonly Border _container;
    private readonly Label _label;
```

```
    private readonly Image _checkImage;
    private readonly StackLayout
_contentStack;

    public CustomChip()
    {
        _label = new Label
        {
            TextColor = Colors.Black,
            VerticalOptions =
LayoutOptions.Center,
            Margin = new Thickness(5, 0)
        };

        _checkImage = new Image
        {
            Source = IconSource,
            HeightRequest = 16,
            WidthRequest = 16,
            IsVisible = false,
            VerticalOptions =
LayoutOptions.Center
        };

        _contentStack = new StackLayout
        {
            Orientation =
StackOrientation.Horizontal,
            Spacing = 5,
            Children = { _checkImage, _label }
        };

        _container = new Border
        {
            Padding = new Thickness(8, 5),
            StrokeShape = new RoundRectangle
            {
                CornerRadius = new
CornerRadius(8)
            },
            Stroke = Colors.LightGray,
            BackgroundColor = Colors.White,
            Content = _contentStack
        };

        Content = _container;

        var tapGestureRecognizer = new
TapGestureRecognizer();
        tapGestureRecognizer.Tapped +=
OnChipTapped;

GestureRecognizers.Add(tapGestureRecogniz
er);

        UpdateVisualState();
    }

    private static void
OnTextChanged(BindableObject bindable,
object oldValue, object newValue)
    {
        var control = (CustomChip)bindable;
        control._label.Text =
newValue?.ToString();
    }

    private static void
OnIsSelectedChanged(BindableObject
bindable, object oldValue, object newValue)
    {
        var control = (CustomChip)bindable;
        control.UpdateVisualState();
    }

    private static void
OnIconSourceChanged(BindableObject
bindable, object oldValue, object newValue)
    {
        var control = (CustomChip)bindable;
        control._checkImage.Source =
(string)newValue;
    }

    private void OnChipTapped(object sender,
EventArgs e)
    {
        IsSelected = !IsSelected;
        SelectedChanged?.Invoke(this,
IsSelected);
    }

    private void UpdateVisualState()
    {
        if (IsSelected)
        {
```

```
            _container.BackgroundColor =
Color.FromArgb("#E8DEF8");
            _container.Stroke =
Colors.Transparent;
            _label.TextColor =
Color.FromArgb("#6750A4");
            _checkImage.IsVisible = true;
        }
        else
        {
            _container.BackgroundColor =
Colors.White;
            _container.Stroke = Colors.LightGray;
            _label.TextColor = Colors.Black;
            _checkImage.IsVisible = false;
        }
    }
}
```

DiagramsGroupPageViewModel (admin app):

```
public partial class
DiagramsGroupPageViewModel :
ObservableObject
{
    private readonly IStatisticsClient
_statisticsClient;
    private SurveyGroupResultsTimeline
_surveyResults;
    private SurveyGroupResultsCategorization
_categorizationResults;

    [ObservableProperty]
    private
ObservableCollection<SurveyCheckBoxItem
> _surveyCheckBoxes = [];

    [ObservableProperty]
    private
ObservableCollection<DistributionPickerItem
> _distributionItems = [];

    [ObservableProperty]
    private DistributionPickerItem
_selectedDistributionItem;

    [ObservableProperty]
    private ISeries[] _series;

    [ObservableProperty]
    private IEnumerable<ISeries> _pieSeries;

    [ObservableProperty]
    private bool _isLoading;

    [ObservableProperty]
    private Guid _groupId;

    private readonly
Dictionary<WorkloadCategories, SKColor>
_categoryColors = new()
    {
        { WorkloadCategories.OnTrack,
SKColors.ForestGreen },
        { WorkloadCategories.Overload20,
SKColors.Orange },
        { WorkloadCategories.Overload50,
SKColors.OrangeRed },
        { WorkloadCategories.CriticalOverload,
SKColors.Crimson },
        { WorkloadCategories.Underload20,
SKColors.SkyBlue },
        { WorkloadCategories.Underload35,
SKColors.RoyalBlue },
        {
WorkloadCategories.CriticalUnderload,
SKColors.DarkBlue }
    };

    private readonly
Dictionary<WorkloadCategories, string>
_categoryNames = new()
    {
        { WorkloadCategories.OnTrack, "В
нормі" },
        { WorkloadCategories.Overload20,
"Перевантажені на 20%" },
        { WorkloadCategories.Overload50,
"Перевантажені на 50%" },
        { WorkloadCategories.CriticalOverload,
"Критично перевантажені" },
        { WorkloadCategories.Underload20,
"Недовантажені на 20%" },
        { WorkloadCategories.Underload35,
"Недовантажені на 35%" },
        {
```

```
WorkloadCategories.CriticalUnderload,
"Критично недовантажені" }
    };

    public ICartesianAxis[] XAxes { get; set; }
= [
        new Axis
        {
            CrosshairLabelsBackground =
SKColors.DarkOrange.AsLvcColor(),
            CrosshairLabelsPaint = new
SolidColorPaint(SKColors.DarkRed),
            CrosshairPaint = new
SolidColorPaint(SKColors.DarkOrange, 1),
            Labeler = value =>
value.ToString("N0")
        }
    ];

    public ICartesianAxis[] YAxes { get; set; }
= [
        new Axis
        {
            CrosshairLabelsBackground =
SKColors.DarkOrange.AsLvcColor(),
            CrosshairLabelsPaint = new
SolidColorPaint(SKColors.DarkRed),
            CrosshairPaint = new
SolidColorPaint(SKColors.DarkOrange, 1),
            Labeler = value =>
value.ToString("N1"),
            CrosshairSnapEnabled = true
        }
    ];

    public
DiagramsGroupPageViewModel(IStatisticsCl
ient statisticsClient)
    {
        _statisticsClient = statisticsClient;
        Series = [];
        PieSeries = [];
    }

    [RelayCommand]
    public async Task LoadSurveyData()
    {
        try
        {
            IsLoading = true;

            _surveyResults = await
_statisticsClient.GetSurveyGroupResultsTime
line(GroupId);

            SurveyCheckBoxes.Clear();

            foreach (var surveyItem in
_surveyResults.Statistics)
            {
                SurveyCheckBoxes.Add(new
SurveyCheckBoxItem
                {
                    Name =
TruncateName(surveyItem.Survey.SurveyNa
me, 20),
                    IsSelected = false,
                    Statistics = surveyItem.Timeline,
                });
            }

            SurveyCheckBoxes.Add(new
SurveyCheckBoxItem
            {
                Name = TruncateName("Загальні
дані", 20),
                IsSelected = false,
                Statistics =
_surveyResults.GlobalTimeline,
            });

            SurveyCheckBoxes.Last().IsSelected
= true;

            _categorizationResults = await
_statisticsClient.GetSurveyGroupResultsCate
gorization(GroupId);

            DistributionItems.Clear();

            DistributionItems.Add(new
DistributionPickerItem
            {
                Name = "Загальні дані",
                Distribution =
_categorizationResults.Global.Distribution,
```

```
            IsSelected = true
        });

        foreach (var survey in
_categorizationResults.Surveys)
        {
            DistributionItems.Add(new
DistributionPickerItem
            {
                Name =
TruncateName(survey.Survey.SurveyName,
20),
                Distribution =
survey.Categorization.Distribution,
                IsSelected = false
            });
        }

        SelectedDistributionItem =
DistributionItems.First();

        UpdateChart();
        UpdatePieChart();
      }
      catch (Exception ex)
      {
        await
Application.Current.MainPage.DisplayAlert("
Помилка",
            $"Не вдалося завантажити дані
опитувань: {ex.Message}", "OK");
      }
      finally
      {
        IsLoading = false;
      }
   }

   private string TruncateName(string name,
int maxLength)
   {
      if (string.IsNullOrEmpty(name) ||
name.Length <= maxLength)
        return name;

      return name.Substring(0, maxLength - 3)
+ "...";
   }

   [RelayCommand]
   private void UpdateChart()
   {
      IEnumerable<SurveyCheckBoxItem>
surveyStatistics =
SurveyCheckBoxes.Where(cb =>
cb.IsSelected);
      List<LineSeries<float>> lineSeries = [];

      var strokeDashArray = new float[] { 3, 2,
3, 2, 1, 2 };
      var effect = new
DashEffect(strokeDashArray);

      var colors = new[]
      {
        SKColors.DodgerBlue,
        SKColors.Crimson,
        SKColors.ForestGreen,
        SKColors.Orange,
        SKColors.Purple,
        SKColors.Teal,
        SKColors.Brown,
        SKColors.DarkGoldenrod
      };

      int colorIndex = 0;

      foreach (var surveyStatistic in
surveyStatistics)
      {
        var surveyName =
surveyStatistic.Name.Substring(0,
Math.Min(10, surveyStatistic.Name.Length));
        var actualPoints =
surveyStatistic.Statistics.DataPoints;
        var predictedPoints =
surveyStatistic.Statistics.PredictedPoints;

        var currentColor = colors[colorIndex
% colors.Length];
        colorIndex++;

        lineSeries.Add(
          new LineSeries<float>
          {
            Values = actualPoints,
```

```
                Name = $"{surveyName}:
Факт",
                Stroke = new SolidColorPaint()
                {
                    Color = currentColor,
                    StrokeThickness = 2
                },
                Fill = new
SolidColorPaint(currentColor.WithAlpha(80)
),
                GeometrySize = 8,
                GeometryStroke = new
SolidColorPaint(currentColor, 2),
                GeometryFill = new
SolidColorPaint(SKColors.White)
            });

        lineSeries.Add(
            new LineSeries<float>
            {
                Values = predictedPoints,
                Name = $"{surveyName}:
Прогноз",
                Stroke = new SolidColorPaint()
                {
                    Color = currentColor,
                    StrokeThickness = 1,
                    PathEffect = effect,
                },
                Fill = new
SolidColorPaint(SKColors.Transparent),
                GeometrySize = 0
            });
        }

        Series = lineSeries.ToArray();
        OnPropertyChanged(nameof(Series));
    }

    [RelayCommand]
    private void UpdatePieChart()
    {
        var distribution =
SelectedDistributionItem.Distribution;

        if (distribution.Count == 0)
        {
            PieSeries = [];
OnPropertyChanged(nameof(PieSeries));
            return;
        }

        var values = new List<ISeries>();

        foreach (var item in distribution)
        {
            if (item.Value > 0)
            {
                var category = item.Key;
                var count = item.Value;
                var categoryName =
_categoryNames.TryGetValue(category, out
var name) ? name : category.ToString();
                var categoryColor =
_categoryColors.TryGetValue(category, out
var color) ? color : SKColors.Gray;

                values.Add(new PieSeries<int>
                {
                    Values = new[] { count },
                    Name = categoryName,
                    Fill = new
SolidColorPaint(categoryColor)
                });
            }
        }

        PieSeries = values;
        OnPropertyChanged(nameof(PieSeries));
    }

    partial void
OnSelectedDistributionItemChanged(Distribu
tionPickerItem value)
    {
        if (value != null)
        {
            UpdatePieChart();
        }
    }
}

public partial class SurveyCheckBoxItem :
ObservableObject
{
```

```
    [ObservableProperty]
    private string _name;

    [ObservableProperty]
    private bool _isSelected;

    [ObservableProperty]
    private Timeline _statistics;
}

public partial class DistributionPickerItem :
ObservableObject
{
    [ObservableProperty]
    private string _name;

    [ObservableProperty]
    private IDictionary<WorkloadCategories,
int> _distribution;

    [ObservableProperty]
    private bool _isSelected;

    public override string ToString()
    {
        return Name;
    }
}
```

ImportSyllabusPageViewModel.cs (admin app):

```
public partial class
ImportSyllabusPageViewModel :
ObservableObject
{
    private readonly IImportClient
_importClient;
    private FileResult _selectedFile;

    [ObservableProperty]
    private string _fileName;

    [ObservableProperty]
    private bool _isFileSelected;

    public
ImportSyllabusPageViewModel(IImportClien
t importClient)
    {
        _importClient = importClient;
        IsFileSelected = false;
        FileName = "Файл не обрано";
    }

    [RelayCommand]
    private async Task PickAndImportFile()
    {
        try
        {
            _selectedFile = await
FilePicker.PickAsync(new PickOptions
            {
                PickerTitle = "Оберіть PDF файл",
                FileTypes = FilePickerFileType.Pdf
            });

            if (_selectedFile != null)
            {
                FileName =
_selectedFile.FileName;
                IsFileSelected = true;
            }
            else
            {
                FileName = "Файл не обрано";
                IsFileSelected = false;
            }
        }
        catch (Exception ex)
        {
            await
Application.Current.MainPage.DisplayAlert("
Помилка", $"Помилка при виборі файлу:
{ex.Message}", "OK");
        }
    }

    [RelayCommand]
    private async Task ImportSyllabus()
    {
        if (_selectedFile == null)
        {
            await
Application.Current.MainPage.DisplayAlert("
Помилка", "Будь ласка, оберіть файл перед
```

```
імпортом", "OK");
            return;
        }

        try
        {
            using var stream = await
_selectedFile.OpenReadAsync();
            byte[] fileBytes;
            using (var memoryStream = new
MemoryStream())
            {
                stream.CopyTo(memoryStream);
                fileBytes =
memoryStream.ToArray();
            }

            var fileToSend = new
ByteArrayPart(fileBytes,
_selectedFile.FileName, "application/pdf");

            await
_importClient.ImportSyllabus(fileToSend);

            await
Application.Current.MainPage.DisplayAlert("
Успіх", "Силабус успішно імпортовано",
"OK");

            await Shell.Current.GoToAsync("..");
        }
        catch (Exception ex)
        {
            await
Application.Current.MainPage.DisplayAlert("
Помилка", $"Помилка при імпорті
силабусу: {ex.Message}", "OK");
        }
    }
}
```

AppShell.xaml.cs (admin app):

```
public partial class AppShell : Shell
{
    public AppShell()
    {
        InitializeComponent();
        Routing.RegisterRoute("LoginPage",
typeof(LoginPage));

Routing.RegisterRoute("CreateCoursePage",
typeof(CreateCoursePage));

Routing.RegisterRoute("SyllabusGroupPage",
typeof(SyllabusGroupPage));
        Routing.RegisterRoute("SyllabusPage",
typeof(SyllabusPage));

Routing.RegisterRoute("CreateSyllabusPage"
, typeof(CreateSyllabusPage));

Routing.RegisterRoute("UpdateCoursePage",
typeof(UpdateCoursePage));

Routing.RegisterRoute("CreateSurveyGroupP
age", typeof(CreateSurveyGroupPage));

Routing.RegisterRoute("CreateSurveyPage",
typeof(CreateSurveyPage));
        Routing.RegisterRoute("LinkPage",
typeof(LinkPage));
        Routing.RegisterRoute("DiagramsPage",
typeof(DiagramsPage));

Routing.RegisterRoute("DiagramsGroupPage
", typeof(DiagramsGroupPage));

Routing.RegisterRoute("ImportSyllabusPage"
, typeof(ImportSyllabusPage));

Routing.RegisterRoute("ImportedSyllabusesP
age", typeof(ImportedSyllabusesPage));

Routing.RegisterRoute("SyllabusImportedPag
e", typeof(SyllabusImportedPage));
    }
}
```