Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система для організації практичних занять в автошколі. Mobile Application
(тема)

Виконав:
студент 4 курсу, групи ПЗПІ 20-4

Панасенко І.С.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник ст.викл. кафедри ПІ Олійник О.В.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Панасенку Ігорю Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Програмна система для організації практичних занять в автошколі. Mobile Application

Затверджена наказом по університету від 20.05. 2024р. № 471 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 20.06.2024

3. Вихідні дані до роботи Розробити мобільний додаток для комплексної програмної системи, яка призначена для організації практичних занять в автошколі. Використовувати мову програмування C# та фреймворк .NET MAUI.

4. Перелік питань, що потрібно опрацювати в роботі

Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, перелік джерел посилань, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	21.05.2024	*виконано*
2	Створення специфікації ПЗ	22.05.2024	*виконано*
3	Проектування ПЗ	24.05.2024	*виконано*
4	Розробка ПЗ	08.05.2024	*виконано*
5	Тестування ПЗ	09.05.2024	*виконано*
6	Оформлення пояснювальної записки	11.06.2024	*виконано*
7	Підготовка презентації та доповіді	13.06.2024	*виконано*
8	Попередній захист	14.06.2024	*виконано*
9	Нормоконтроль, рецензування	15.06.2024	*виконано*
10	Здача роботи у електронний архів	16.06.2024	*виконано*
11	Допуск до захисту у зав. кафедри	19.06.2024	*виконано*

Дата видачі завдання 21 травня 2024р.

Студент (ка)_____ Панасенко І.С.

(підпис)

Керівник роботи _____ ст.викл. кафедри ПІ Олійник О.В.

(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 93 стор., 35 рис., 1 табл., 10 джерел.

АВТОМАТИЗАЦІЯ, АВТОШКОЛА, ІНСТРУКТОРИ, МОБІЛЬНИЙ ЗАСТОСУНОК, НАВЧАННЯ, СТУДЕНТИ АВТОШКОЛИ, .NET MAUI, ANDROID, IOS, MVVM, SQLITE.

Об’єкт розробки – мобільний додаток для автоматизації проведення практичних занять в автошколі.

Мета розробки – створення зручного та ефективного інструменту на мобільній платформі для планування, ведення та аналізу практичних занять в автошколі, який задовольняє потреби як інструкторів, так і учнів.

Методи рішення – використання фреймворку .NET MAUI, архітектурного підходу MVVM, менеджерів стану бази даних SQLite для зберігання та керування даними, а також взаємодію з бекенд частиною шляхом надсилання запитів по протоколу HTTP для синхронізації та оновлення інформації.

У результаті розробки створено мобільний застосунок з простим та зрозумілим інтерфейсом, здатним оптимізувати та спростити процес проведення практичних занять в автошколі для всіх учасників навчального процесу.

AUTOMATION, DRIVING SCHOOL, INSTRUCTORS, MOBILE APPLICATION, EDUCATION, DRIVING SCHOOL STUDENTS, .NET MAUI, ANDROID, IOS, MVVM, SQLITE.

The development object is a mobile application for automating practical driving lessons at a driving school.

The development goal is to create a convenient and efficient tool on the mobile platform for planning, conducting, and analyzing practical driving lessons at the driving school, which satisfies the needs of both instructors and students.

The methods of solving the task involve using the .NET MAUI framework, the MVVM architectural approach, SQLite database state managers for storing and managing data, as well as interacting with the backend by sending requests via the HTTP protocol for synchronization and updating of information.

The result of the work is a created mobile application with a simple and intuitive interface capable of optimizing and simplifying the process of conducting practical driving lessons at the driving school for all participants of the educational process.

Я, Панасенко Ігор Сергійович, студент гр. ПЗПІ-20-4, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система для організації практичних занять в автошколі. Mobile», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

У сучасному світі технології переплітаються з усіма сферами нашого життя, надаючи нові можливості для покращення різних аспектів щоденного існування. Однією з галузей, яка постійно вдосконалюється завдяки інноваціям, є освіта.

Для отримання водійського посвідчення в Україні обов’язкове проходження практичної підготовки в автошколі, що визначено у положенні про порядок видачі посвідчень водія та допуску громадян до керування транспортним засобом (постанова Кабінета Міністрів України від 08.05.1993 № 340) [1].

Сучасні машини представляють собою не тільки засіб для транспортування, але й гаджет, який пропонує різні розумні функції. Однією з таких функцій є автопілот, що в перспективі має дозволити машині перевозити людину з однієї точки в іншу без необхідності безпосереднього керування водієм. Хоч така технологія активно розвивається та вже існують такі концепти автомобілів, що можуть рухатися самостійно, дана функція ще далека від масового впровадження й всі сучасні автомобілі що випускаються, потребують присутність людини, яка вміє ними керувати. До того ж, функції автопілоту можливі лише на машинах з автоматичними коробками передач, тоді як транспортні засоби з механічною коробкою передач ніколи не зможуть позбутися водіїв. Таким чином навчання в автошколах залишиться актуальним ще досить тривалий час.

Наразі більшість українських автошкіл не мають свого мобільного застосунку, в такому випадку інструктори друкують на листочках розклад занять, куди потім записують студентів, а також мають окремий листочок для кожного студента для відслідковування прогресу навчання. Студенти, в свою чергу, для уточнення будь-яких питань вимушені писати або дзвонити інструктору, відволікаючи його від іншого заняття, а перегляд свого розкладу та контроль прогресу по навичкам для них недоступний. Після аналізу тих небагатьох застосунків, що присутні на ринку українських автошкіл було виявлено, що вони не мають достатнього функціоналу для задоволення мети користувачів, як з точки зору студента, так і з точки зору інструктора.

Відповідно до описаних вище проблем, мета розробки програмної системи полягає у тому, щоб створити мобільний застосунок, який дозволить автоматизувати як процес навчання для студентів, так і роботу з розкладом для інструкторів, а також надати можливість для спілкування через чат інструктора та учня.

Під час виконання роботи спочатку було проведено аналіз предметної галузі та спроектована архітектура системи, після чого було розроблено функціонал застосунку, побудовано візуальну частину й безпосередньо реалізовано сам мобільний додаток.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної галузі

Тема навчання в автошколі є актуальною. Отримати водійське посвідчення бажають, як більшість серед молоді, так і люди вже старшого віку, які раніше, за якихось причин, ще не отримали водійське посвідчення. Аналізуючи статистику отримання водійських посвідчень можна стверджувати, що ця послуга не втрачає своєї актуальності та популярності серед українців, за 2022 рік 430 тисяч водіїв отримали такий документ вперше, а за перші три місяці 2024 року було видано близько 57 тисяч посвідчень водія після складання практичного іспиту [2].

Кожна автошкола прагне надати своїм студентам гарний досвід навчання, що включає у собі як професійних спеціалістів, так і надання зручних способів для керування розкладом, оплати занять, обрання інструктора, можливість спілкуватися зі своїм інструктором, а також можливості розуміти свій прогрес проходження навчання. Якщо за гарних спеціалістів відповідає адміністрація автошколи та людина яка приймає інструкторів з практичних занять на роботу, то інші пункти, що допоможуть залишити найкраще враження від навчання, можуть бути надані у мобільному додатку. Зараз майже кожна людина має смартфон і постійно носить його з собою, тому учні зможуть у будь-який момент записатися на заняття, або відмінити майбутнє заняття, якщо виникла така потреба, або написати інструктору для уточнення необхідних питань.

Процес навчання практичним навичкам водіння в автошколах можна описати так:

- обрати автошколу для проходження навчання;
- подати необхідні документи для зарахування на курс, включно з довідкою про успішне складання іспиту з теоретичної підготовки в сервісному центрі МВС;
- очікувати поки адміністратори автошколи закріплять вас за інструктором з яким буде відбуватися навчання;

- практичні заняття з інструктором. Рекомендована кількість годин, яку учень має провести за кермом перед здачею іспитів у сервісному центра становить 40 годин, зважаючи на прогрес кожного учня цей час може бути змінено;
- складання внутрішнього іспиту в автошколі та отримання довідки про успішне завершення практичного навчання в автошколі;
- складання практичного іспиту в сервісному центра МВС.

Зібравши інформацію у людей, які нещодавно отримали водійське посвідчення, а також, беручи до уваги власний досвід, можемо стверджувати, що у більшості автошкіл процес навчання не є автоматизованим. Майбутній водій не може обрати на якій машині він хоче проходити навчання (звичайно окрім типу коробки передач якою буде обладнана машина), або який інструктор йому більше до вподоби, єдиний спосіб якось висловити свої побажання це сказати їх адміністратору при подачі документів. Розклад занять інструктора є тільки у нього, роздрукований на листочку, куди він здійснює запис, тому для того, щоб записатися на заняття потрібно зв’язатися з інструктором, що є незручним як для учнів, так і для самих інструкторів. Прогрес по вивченню навичок також фіксується в інструктора на роздрукованому листочку, переглянути що ще потрібно опанувати студент може тільки під час заняття, або попередньо сфотографувавши цей листок. Зв’язок з інструктором, найчастіше, відбувається через дзвінки по телефону, хоча також можливе спілкування через месенджери, проте обидва способи доставляють інструкторам незручності через те, що кожного дня вони повинні відповідати студентам стосовно розкладу.

Переглянувши застосунки у Google Play та App Store ми визначили, що на українському ринку дуже мало автошкіл, які б мали свій мобільний застосунок. А навіть ті які мають не покривають увесь необхідний функціонал для студентів та інструкторів. Мобільні додатки автошкіл можуть не мати такого функціоналу як: можливість вибрати інструктора, оплата занять, чат з інструктором, та інших.

Для полегшення роботи працівників автошкіл, а також для надання кращого досвіду навчання для учнів рішенням є створення програмної системи, яка зможе

автоматизувати процес проходження практичного навчання в автошколах і задовольнити всі потреби як студентів, так і інструкторів.

1.2 Аналіз ринку

На ринку мобільних додатків автошкіл представлена досить невелика кількість застосунків. У таких магазинах як google play market та AppStore більшість при спробі знайти застосунки для автошкіл у першу чергу ми побачимо ігри для навчання водіння, де потрібно керуючи машиною за допомогою елементів на екрані керувати машиною на дорозі. Також існують додатки, що дозволяють вивчати теоретичну частину навчального курсу, вони надають необхідні матеріали, а також дозволяють проходити тести по вивченим темам. Проте нам все таки вдалося знайти декілька застосунків, що реалізують функціонал пов’язаний з автоматизацією практичних занять під час навчання. Аналіз цих застосунків й було проведено.

При проведенні аналізу було приділено увагу дизайну, зручності користування, наявності функціоналу, що покриває потреби користувачів, а також на наявність додаткових “фішок”, що можуть сподобатися учням. Також ми визначили недоліки наших конкурентів, щоб обов’язково звернути на них увагу при реалізації нашої системі, що дозволить їй виглядати більш привабливо в очах користувачів.

Перший додаток, який було розглянуто в якості аналога створюваної системи називається “Автошкола Карат”.

Мобільний застосунок київської автошколи “Карат” можна знайти у Google Play Market та App Store. Згідно з даними застосунку SensorTower, додаток автошколи знаходиться на 371 місці у категорії “спосіб життя” серед застосунків в Україні. Загальна ж кількість завантажень за весь час існування додатку перебуває в межах від 5 до 10 тисяч. Рейтинг застосунку автошколи складає 4.4 зірки на основі 286 відгуків у Google Play Market.

У опису застосунку зазначено, що додаток створений для учнів автошколи “Карат” та призначений для онлайн-запису на практичні заняття по водінню.

Учень зможе обрати інструктора та час для заняття, з можливістю змінити графік. Також зазначається, що користувачу буде надіслано нагадування про наближення заняття.

Дизайн додатку на прикладі скріншотів декількох сторінок наведено на рисунках 1.1 та 1.2.

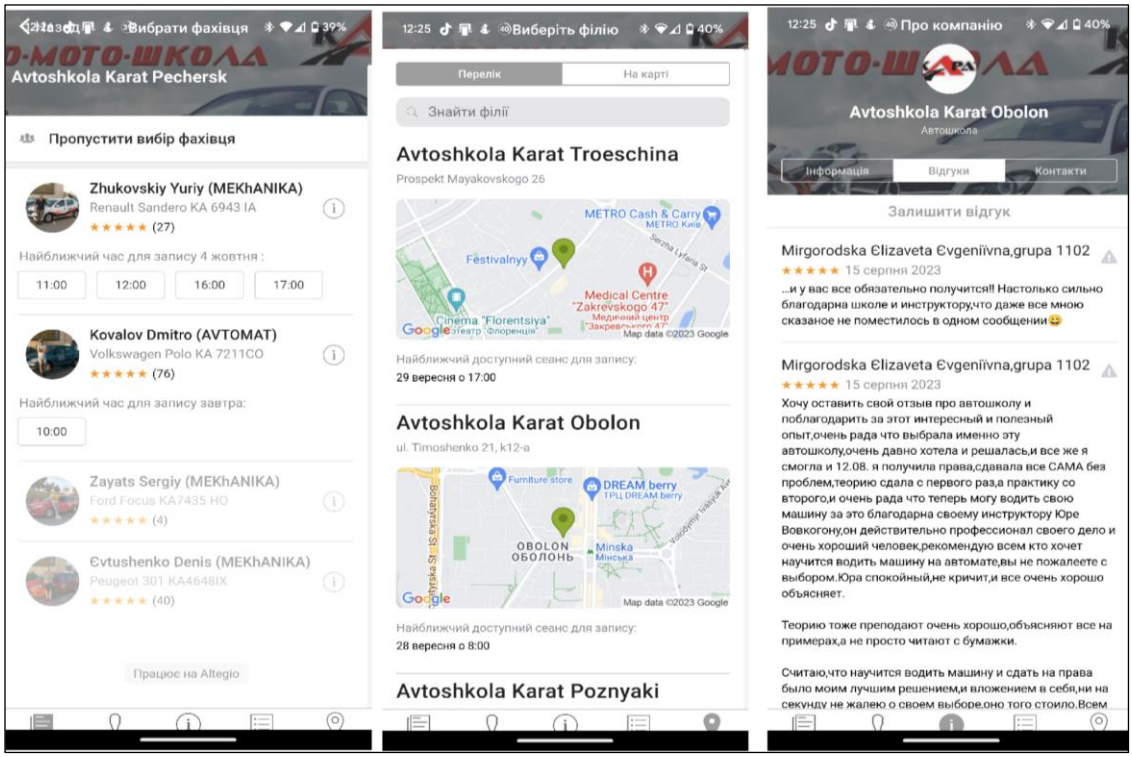


Рисунок 1.1 – Дизайн додатку Автошкола Карат

Після аналізу функціоналу додатку ми виділили його плюси та мінуси.

Переваги:

- зрозумілий інтерфейс;
- інтуїтивний запис на заняття;
- повідомлення про наближення часу заняття;
- широкий вибір інструкторів;
- можливість переглядати відгуки інших користувачів;
- попередження про скасування заняття за 24 години до його початку, або необхідність повної оплати;
- список майбутніх занять та історія проведених занять.

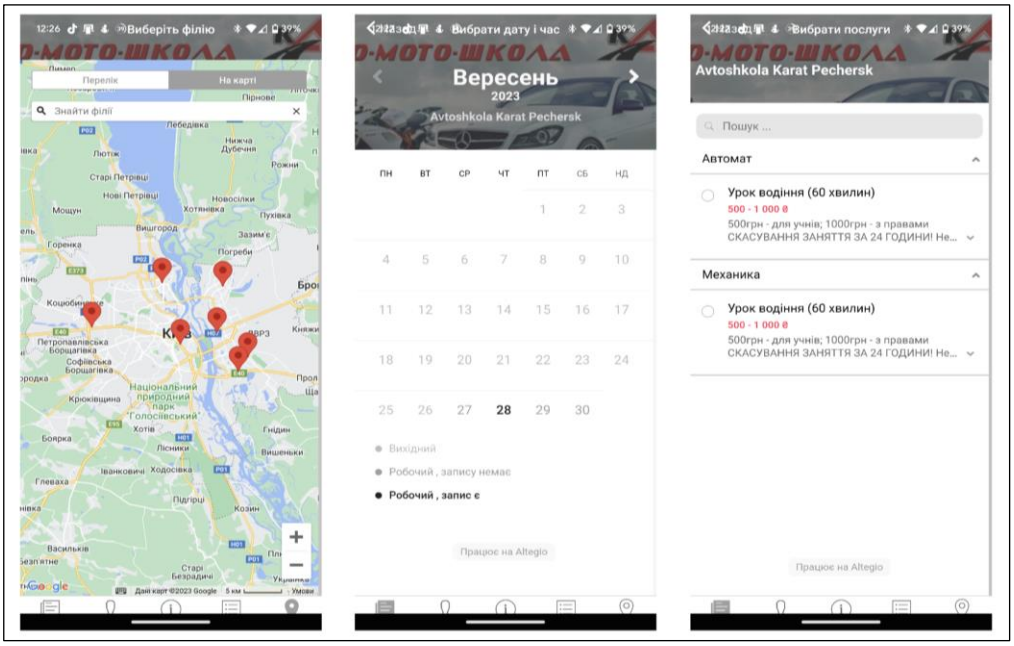


Рисунок 1.2 – Дизайн додатку Автошкола Карат

Незважаючи на досить велику кількість переваг даний мобільний застосунок також має ряд недоліків:

– неправильне відображення нижнього меню, що значно ускладнює навігацію;
– незрозуміло як виконати реєстрацію;
– відсутня можливість входу через сторонні сервіси, такі як Google та Facebook;
– немає відгуків про конкретних інструкторів та їх рейтинг;
– немає відгуків про машини та їх стан;
– відсутня можливість відстежувати прогрес занять;
– застосунок розроблений тільки для учнів і ніяк не допомагає інструкторам з водіння.

Другий додаток, який було розглянуто в якості аналога створюваної системи називається «Автошкола-контроль».

Мобільний застосунок «Автошкола-Контроль» можна знайти у Google Play Market та App Store. Згідно з даними сайту SensorTower, додаток автошколи знаходиться на 80 місці у категорії “навчання”. Він має близько 10000

завантажень у березні 2024 року, загальна ж кількість завантажень перебуває в межах від 100 до 500 тисяч. Перший реліз застосунку випущено. Мінімальна версія операційної системи для запуску застосунку Android 5.0. Рейтинг застосунку автошколи складає 3.5 зірки на основі 1,31 тисячі відгуків у Google Play Market.

У опису застосунку зазначено, що він надає особистий кабінет у автошколі. Також розробниками додатку визначається наступний опис: додаток створений для учнів та співробітників, що використовують сервіс “Автошкола-Контроль”.

Даний додаток надає такі можливості для учнів учнів: розклад та запис на водіння та теоретичні заняття; онлайн-навчання (якщо надається автошколою); тренування по актуальним білетам та темам ПДР; розрахунок часу та кількості занять з керування; історія занять; маршрути; отримання сповіщень: чати з автошколою; інформація про автошколу.

Також надаються такі можливості для інструкторів: створення особистого розкладу; профіль кожного учня зі всіма даними; маршрути занять; запис та відміна занять; отримання сповіщень; чати з іншими інструкторами та учнями.

Дизайн додатку на прикладі декількох сторінок наведено на рисунках 1.3 та 1.4.

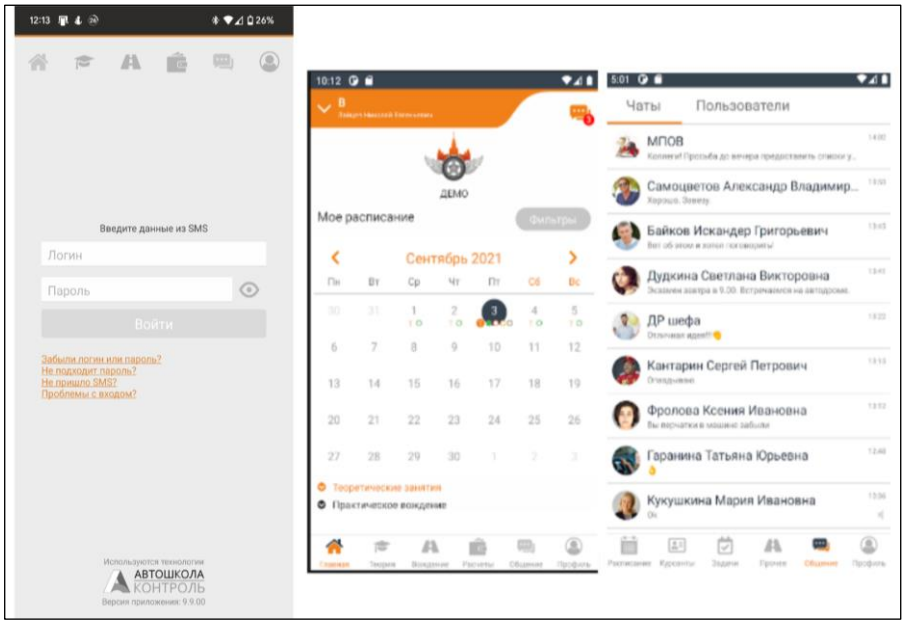


Рисунок 1.3 – Дизайн додатку Автошкола Контроль

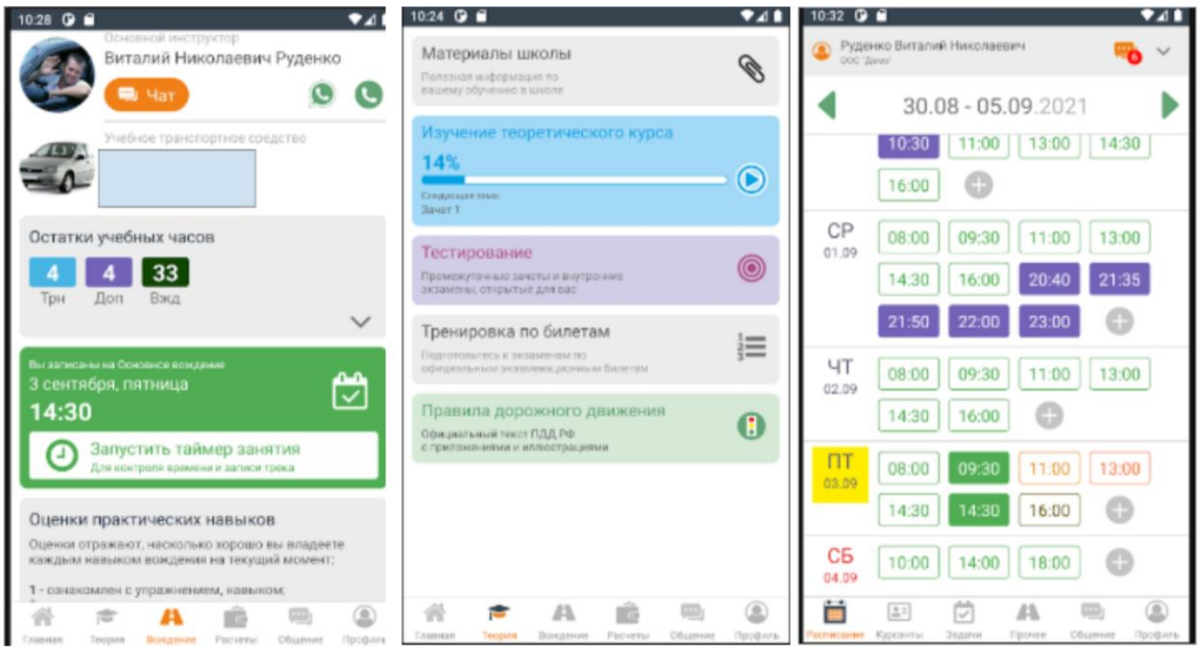


Рисунок 1.4 – Дизайн додатку автошкола Контроль

В результаті аналізу роботи застосунку ми змогли виділили такі його плюси та мінуси.

Переваги:

– забезпечення безпеки даних користувачів шляхом недопущення до користування людей, що не проходять навчання;
– можливість переглянути розклад занять у виді календаря;
– можливість записатися на заняття;
– суміщення теоретичної та практичної частин;
– можливість переглядати кількість навчальних годин за кермом;
– доступне онлайн навчання теоретичній частині;
– можливість зв’язатися з інструктором або автошколою через чат;
– наявність функціоналу для інструкторів, а не тільки для учнів.

Недоліки:

– неможливість користуватися застосунком людині, що не навчається в автошколі (відсутність навіть гостьового режиму);
– відсутність можливості оплати заняття прямо у застосунку;

– неможливість вибрати, чи поширювати персональні дані інструктору (деякі люди хотіли, щоб їх прізвище, або дату народження не розголошували для інструкторів);
– відсутність коментарів та оцінок, від людей, що вже пройшли навчання, або проходять зараз.
– відсутній список машин доступних для навчання;
– відсутній список інструкторів;
– немає можливості вибрати машину або інструктора для проходження практичної частини навчання.

Третій додаток, який було розглянуто в якості аналога створюваної системи має назву “Онлайн автошкола Час”.

Мобільний застосунок автошколи “Час” доступний у магазині Google Play Market та App Store безкоштовно. Згідно з даними застосунку SensorTower, названий застосунок знаходиться на 204 місці у розділі “Навчання”, серед застосунків в Україні, має менше ніж 5000 завантажень у березні 2024 року, загальна кількість завантажень перебуває в межах від 10 до 50 тисяч. Рейтинг застосунку складає 4.3 зірки на основі 100 відгуків (див.рис.1.5-1.6).

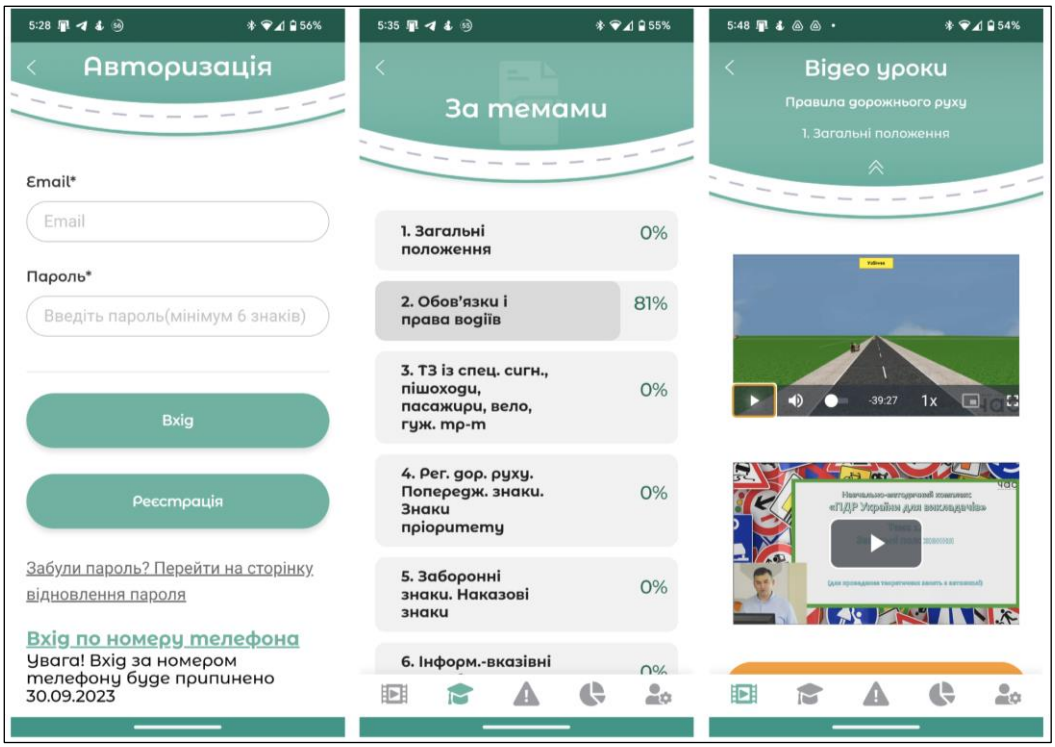


Рисунок 1.5 – Дизайн додатку онлайн автошкола Час

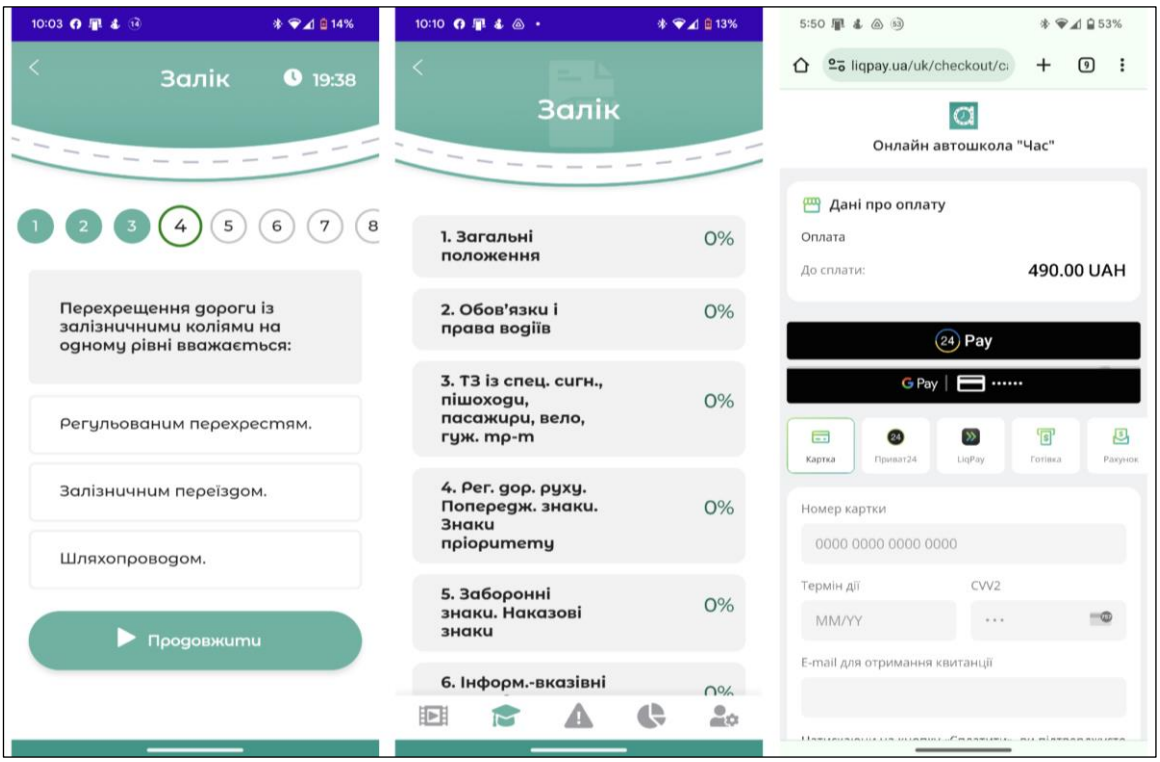


Рисунок 1.6 – Дизайн додатку онлайн автошкола Час

В результаті аналізу роботи застосунку ми змогли виділили такі його плюси та мінуси.

Переваги:

– приємний, ненав’язливий дизайн;
– можливість безкоштовно ознайомитися з майже з усім функціоналом додатку.
– багато варіантів для вибору підписки;
– присутність локалізації;
– можливість звернутися до інструктора;
– указане авторське право з відповідними документами;
– наявність всього необхідного (відеоуроки, додаткові матеріали, тести, екзамени, іспит, питання на які була дана помилкова відповідь) для підготовки до іспиту при оплаті підписки;
– наявність сторінки зі статистикою пройденого матеріалу.

Також можна виділити такі недоліки:

– відсутня можливість увійти через сторонній акаунт (хоча б через гугл).

- відображення пароля у явному вигляді на сторінці профіля, враховуючи те, що для повторного входу у застосунок не потрібна авторизація, є несек'юрним рішенням.
- зависання додатку (особисто у мене це трапилося мінімум 3 рази під час дослідження функціоналу). Користувач просто бачить білий екран. Позбутися цього можна лише повністю перезайшовши у додаток;
- відсутність локалізації навчальних матеріалів. Попри наявність загальної локалізації інтерфейсу навчальні матеріали доступні лише українською;
- при оцінці додатку зірками, користувач не бачить ніякого підтвердження, що його оцінка була прийнята, або ж відхилена;
- відсутність коментарів та оцінок інших користувачів. Ми навіть не можемо подивитися власну оцінку;
- не зберігання проміжкових результатів тестів за темами. У деяких темах може бути близько ста питань і не завжди у людини є можливість відповісти на всі за один раз. Але при виході з тестування прогрес не буде збережено. Про це користувачу видається повідомлення, проте краще б просто зберігався прогрес;
- невідповідність між іспитом у сервісному центрі, що проводиться у форматі 20 хвилин на 20 питань, та іспитом у застосунку, який надає 40 хвилин для 40 питань;
- повна відсутність будь-якої інформації щодо практичної частини навчання. Саме проходження курсу з практичної частини буде головним функціоналом нашої програмної системи.

Відповідно до результатів проведеного аналізу ринку можна зробити висновок, що кожен застосунок дотримується свого власного дизайну на кожній сторінці, через що ними приємно користуватися. Критично важливим є забезпечення доступності всіх основних елементів керування та навігації у застосунку. Оскільки в застосунках присутня оплата, необхідно захищати дані про кредитні картки користувачів. Загалом же система має бути зручною та доступною, мати інструменти для відстеження прогресу, а також надавати

аналітику щодо виконання вправ, надавати можливість для комунікації між студентом та інструктором, також слід забезпечити додаткові функції, такі як інтеграцію з гугл акаунтом, й синхронізацію записів на заняття з гугл календарем. Системи-аналоги, які були описані вище не покривають весь необхідний для користувача функціонал, це можна побачити у таблиці 1.1, де ми вказали результати оцінки додатків за 10 бальною шкалою.

Таблиця 1.1 – Порівняння створюваного додатку з конкурентами

	Назви продуктів			
Показник	Створюваний додаток	Автошкола Карат	Автошкола Контроль	Онлайн Автошкола Час
Дизайн	8	9	9	8
Функціональність	10	8	9	5
Обслуговування	9	5	5	6
Надійність	10	6	9	8
Продуктивність	10	7	8	7
Інноваційність	5	3	2	2
Клієнтське задоволення	9	8	7	8
Підтримка	8	6	6	7
Загалом	69	52	55	51

Відповідно до результатів наведених у таблиці, можемо побачити що наша система повинна бути кращою за конкурентів майже по всім параметрам.

1.3 Постановка задач

Основною метою цієї роботи є створення мобільного додатку “SPAS” для пристроїв з операційною системою Android. Враховуючи проведений аналіз конкурентів на ринку необхідно максимально задовольнити всі потреби користувачів. Система що розробляється зможе покращити досвід навчання практичним навичкам водіння в автошколах, а також автоматизує процеси керування розкладом та відстеження прогресу.

Мобільний додаток буде доступний на операційних системах Android та IOS. Окрім мобільного додатку система також складатиметься з серверної частини та веб-сайту. Однією з головних задач є розробка мінімалістичного зрозумілого інтерфейсу, з привабливою колірною схемою. Не менш важливим є якісна реалізація функціоналу, покриття всіх потреб користувачів, забезпечення зручності використання, шляхом впровадження пошуку, фільтрації та сортування на сторінках де це може бути необхідним. Під час розробки, а також після її завершення необхідно протестувати увесь створений функціонал, та переконатися у коректній роботі застосунку й відсутності непередбаченої поведінки.

2 ПЕРЕЛІК ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Постановка мети

Метою роботи є проектування та реалізація мобільної частини програмної системи, що буде задовольняти основні вимоги та потреби майбутніх користувачів, учнів автошколи, шляхом надання функцій, що допоможуть автоматизувати процес проходження практичного навчання. Створюваний додаток повинен бути надійним, доступним, захищеним та легко масштабованим.

Для імплементації мобільного застосунку було обрано .NET MAUI [3], що дозволяє забезпечити високу продуктивність та швидкість розробки завдяки єдиною базі коду для iOS та Android, а також можливість розширення на Windows. Такий підхід спрощує управління кодовою базою та забезпечує однаковий досвід для користувачів на різних платформах. Для забезпечення взаємодії нашої додатку з серверною частиною було обрано клас HttpClient [4], що надається пакетом System.Net.Http. Використання HttpClient дозволяє нам легко взаємодіяти з віддаленими серверами, відправляти та отримувати дані за допомогою різних типів HTTP-запитів, таких як GET, POST, PUT, DELETE, та інші. Цей підхід дозволяє забезпечити ефективну та надійну комунікацію між клієнтом та сервером, забезпечуючи високу продуктивність та швидкість обміну даними. Для ефективного зберігання та керування певними даними всередині застосунку було обрано Preferences [5]. Використання Preferences дозволяє нам зберігати дані у вигляді ключ-значення з мінімальними накладними витратами на утримання. Це зручний і швидкий спосіб зберігання і отримання невеликої кількості даних, таких як налаштування, вибрані опції користувача тощо. Використання Preferences допомагає забезпечити стабільну та надійну роботу застосунку, мінімізуючи залежність від зовнішніх джерел даних та забезпечуючи більшу контроль над локальною інформацією.

Архітектура мобільного застосунку повинна бути легковісною, масштабованою та протестованою. Для досягнення таких цілей було обрано MVVM (Model-View-ViewModel) архітектуру в поєднанні з .NET MAUI, яка

дозволяє чітко розділити логіку, відображення та управління даними. Приводом вибору MVVM була його придатність до створення відокремлених компонентів: моделей (Model), які представляють дані та бізнес-логіку; представлень (View), які відображають інтерфейс користувача; та моделей представлення (ViewModel), що відповідають за управління взаємодією між моделями та представленнями. Ця архітектура дозволяє підтримувати чистоту коду та розділення обов'язків між різними компонентами застосунку. Крім того, MVVM полегшує тестування, оскільки окремі компоненти можна тестувати незалежно від інших, що сприяє розвитку надійних та стабільних додатків. Використання MVVM з .NET MAUI дозволить нам створити добре структурований та легко розширюваний додаток, який буде готовий до масштабування та підтримки у майбутньому без значних змін у коді.

2.2 Загальний опис системи

Наша програмна система для організації проведення практичних занять в автошколі представляє собою комплексний мобільний додаток, призначений для операційних систем Android та IOS. Він забезпечує ефективне керування та проведення практичних занять у автошколі, а також зручний доступ до необхідної інформації для інструкторів та учнів.

Головна ідея нашої системи полягає у наданні можливості швидкого та зручного пошуку та підбору інструктора, базуючись на відгуках про нього та про машину, а також у спрощенні організації навчального процесу та полегшенні взаємодії між інструкторами та учнями через наявність вбудованого часу. Учні зможуть завжди отримати доступ до свого прогресу, або записатися на майбутнє завдання, в той час як інструктори отримують інструмент для контролю свого розкладу, а також для відстеження прогресу опанування навичок для кожного учня.

Додаток буде розроблено для платформ Android та IOS, що дозволяє користувачам отримувати доступ до всього функціоналу в будь-який час та з

будь-якого місця, використовуючи смартфони з цією операційною системою, єдиною умовою є підключення пристрою до інтернету.

2.3 Основний функціонал системи

Зазначимо, що створювана програмна система для організації проведення практичних занять в автошколі призначена для двох типів користувачів: Студент автошколи та Інструктор.

Основний функціонал програмної системи для забезпечення покриття потреб користувачів:

- реєстрація користувачів у якості студентів;
- авторизація для учнів;
- авторизація для інструкторів;
- керування особистими даними у профілі;
- перегляд інструкторів та їх рейтинг;
- перегляд відгуків про інструкторів;
- перегляд машин інструкторів та їх рейтинг;
- фільтрація, пошук та сортування інструкторів;
- можливість записатися на курс навчання до інструктора;
- можливість залишити відгук про інструктора та оцінити рейтингом машину;
- перегляд розкладу інструктора;
- можливість записатися на заняття в певний день та час та оплатити заняття одразу в застосунку;
- можливість перенести/відмінити заняття, за умови, що до заняття залишилося більше 24 годин;
- перегляд власного профілю;
- перегляд статистики по навчанню;
- перегляд прогресу у навчанні;
- можливість змінити мову інтерфейсу (українська/англійська);

- чат між інструктором та студентом;
- перегляд своїх учнів (для інструкторів);
- перегляд власного розкладу (для інструкторів);
- редагування власного розкладу (заборона записатися на певне заняття, якщо на нього ще ніхто не записаний).

2.4 Загальні обмеження

Для нашої системи існують такі загальні обмеження:

- студент не може зареєструватися, якщо йому менше 18 років;
- у інструкторів є обмежена кількість місць для запису;
- програмний додаток працює за наявності підключення до бездротової локальної мережі або мобільного зв’язку.

2.5 Припущення та залежності

Для нашої системи можна визначити такі припущення та залежності:

- користувач може записатися до інструктора, який знаходиться в тому ж місті, що й він;
- користувачі, що будуть реєструватися у системі вже пройшли теоретичне навчання та склали іспит у сервісному центрі МВС;
- студенти мають достатню кількість грошей, щоб оплачувати заняття.

3 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ

3.1 Проектування архітектури програмної системи

Для розробки мобільного застосунку ми обрали .NET MAUI (Multi-platform App UI), що є кросплатформеним фреймворком для створення нативних мобільних і десктопних застосунків, з використанням XAML та мови програмування C#. Такий вибір було обумовлено тим, що .NET MAUI надає нам можливість побудувати однаково якісний та функціональний додаток для платформ Android та iOS, зберігаючи при цьому одну кодову базу. Це дозволить нам значно зменшити час розробки та зусилля, необхідні для підтримки та оновлення додатку на різних платформах, тим самим забезпечуючи швидке впровадження та підтримку нових функцій для наших користувачів. Крім того, .NET MAUI ідеально поєднується з екосистемою .NET, що надає доступ до широкого спектру інструментів та бібліотек для розробки, тестування та розгортання додатків.

Правильне проектування архітектури при створенні програмної системи є важливою передумовою для успішної інтеграції всіх компонентів та забезпечення їхньої взаємодії відповідно до вимог та очікувань користувачів. Це означає врахування не лише поточних потреб та функціональності системи, але і масштабування, безпеки, продуктивності та можливості майбутнього розвитку. Коректно спроектована архітектура дозволить легко внести зміни та доповнення до системи у майбутньому, забезпечуючи її стабільність та ефективність протягом усього життєвого циклу.

Для нашої системи ми обрали клієнт-серверну архітектуру [6], оскільки вона надає ефективний та гнучкий спосіб організації взаємодії між різними компонентами системи. Цей підхід дозволяє розділити функціональність між клієнтом та сервером, забезпечуючи оптимальне використання ресурсів та забезпечення швидкості та надійності роботи системи. Клієнтська частина представляє собою мобільний додаток на платформі Android або IOS (див. рисунок 3.1) та відповідає за взаємодію з користувачем та відображення

інформації, тоді як серверна частина відповідає за обробку даних, збереження інформації та виконання бізнес-логіки. Це дозволяє забезпечити високу масштабованість, безпеку та продуктивність системи при збереженні простоти та зручності в її використанні для користувачів.

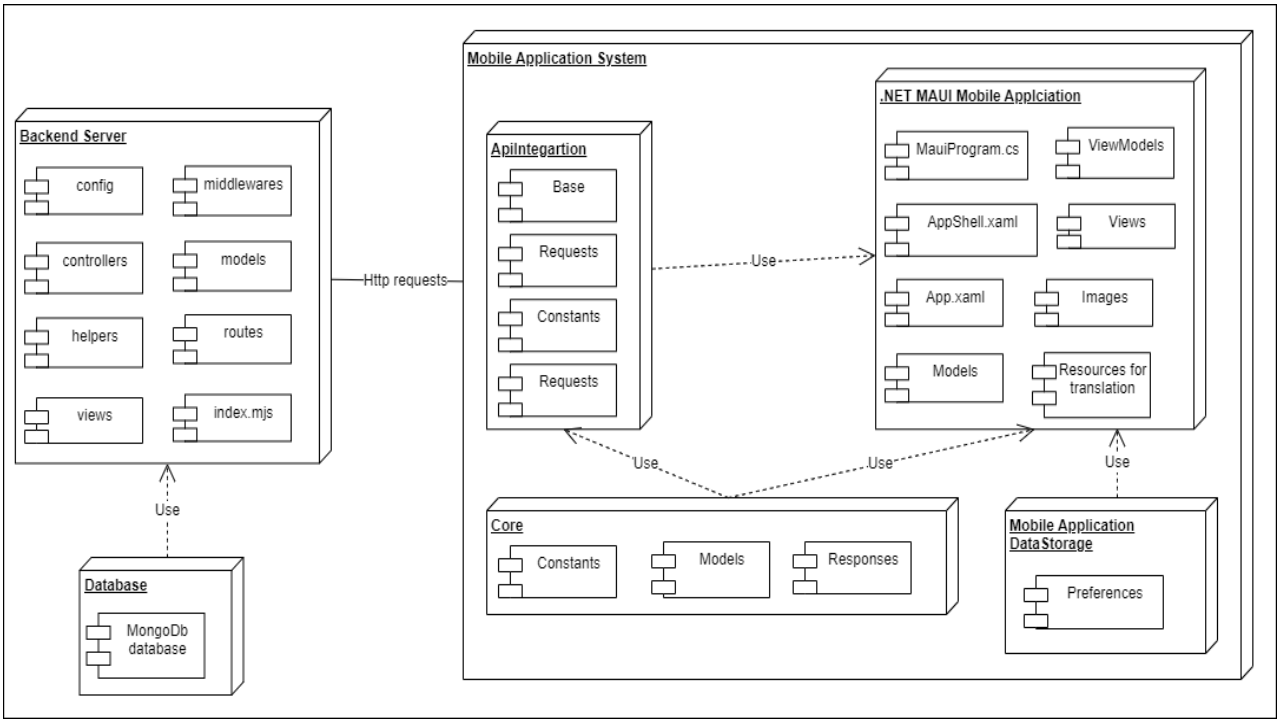


Рисунок 3.1 – Схема архітектури програмної системи

У якості клієнта виступає створюваний мобільний додаток, який буде відображати візуальну частину програмної системи для користувача. Сам мобільний додаток буде створено з використанням MVVM (model, view, view-model) архітектури. Цей підхід дозволить чітко розділити логіку бізнес-логіки (Модель), представлення даних (Вид) та логіку взаємодії між ними (ViewModel), що сприятиме покращенню підтримки та розширення функціональності додатку. Крім того, MVVM дозволяє ефективно керувати відображенням даних та реагувати на зміни в системі, що робить його ідеальним вибором для розробки мобільних застосунків. Як буде побудовано мобільний додаток представлено на діаграмі пакетів (див. рисунок 3.2).

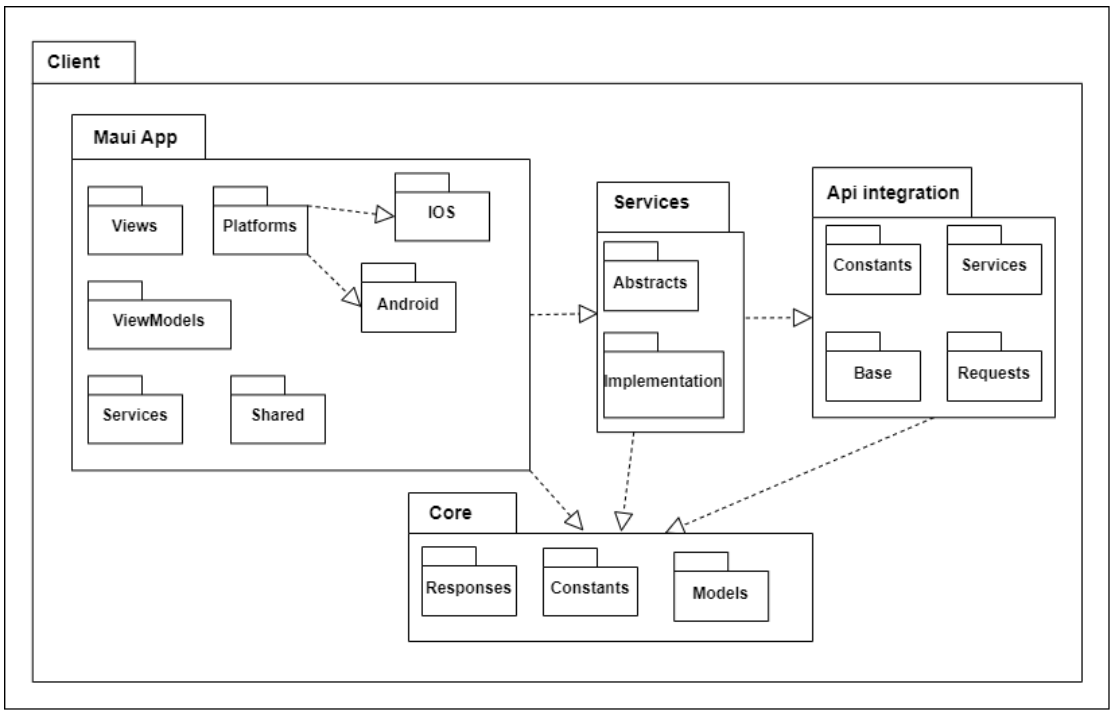


Рисунок 3.2 – Діаграма пакетів для мобільної частини програмної системи

Представлена схема показує які пакети будуть створені та як буде відбуватися взаємодія між ними.

3.2 Проектування бази даних

Після визначення всіх вимог до системи та мети її створення, а також враховуючи проведений аналіз аналогів, було обрано механізм Preferences для використання у клієнтській частині. Preferences – це механізм зберігання пар ключ-значення, який працює локально на мобільних пристроях і не вимагає підключення до зовнішнього сервера [5]. Він відмінно підходить для зберігання налаштувань користувача та невеликих обсягів даних, які потрібно зберігати локально на пристрої, і забезпечує ефективний доступ до них навіть у відсутність Інтернет-з'єднання.

Обраний механізм відповідає вимогам щодо ефективного зберігання та обробки даних на мобільних пристроях, забезпечуючи швидкий доступ до інформації та оптимальні ресурси. Використання Preferences дозволяє забезпечити надійну та ефективну роботу додатку, зменшуючи витрати на розробку та підтримку системи.

3.3 Прецеденти використання

Користуватися створюваним мобільним додатком зможуть тільки зареєстровані користувачі. Новий користувач зможе зареєструватися використавши сам мобільний додаток, або ж відвідавши веб сайт. Важливо, що для реєстрації користувачу повинно виповнитися 18 років.

Для використання додатку передбачено дві ролі, студент, що буде проходити навчання в автошколі та інструктор, що проводить навчання.

Для відображення варіантів використання системи було створено Use-case діаграму [7], що дозволяє легко візуалізувати, як різні актори взаємодіють з системою та які конкретні функціональні можливості вони використовують. Кожен використовуваний випадок (Use-Case) представляє собою конкретну функціональну операцію або послідовність операцій, які користувач може виконати в системі.

Use-Case діаграма (діаграма прецедентів) для студентів демонструє функції які користувач може отримати від системи та який шлях потрібно пройти, щоб отримати доступ до тієї чи іншої функції (див. рисунок 3.3).

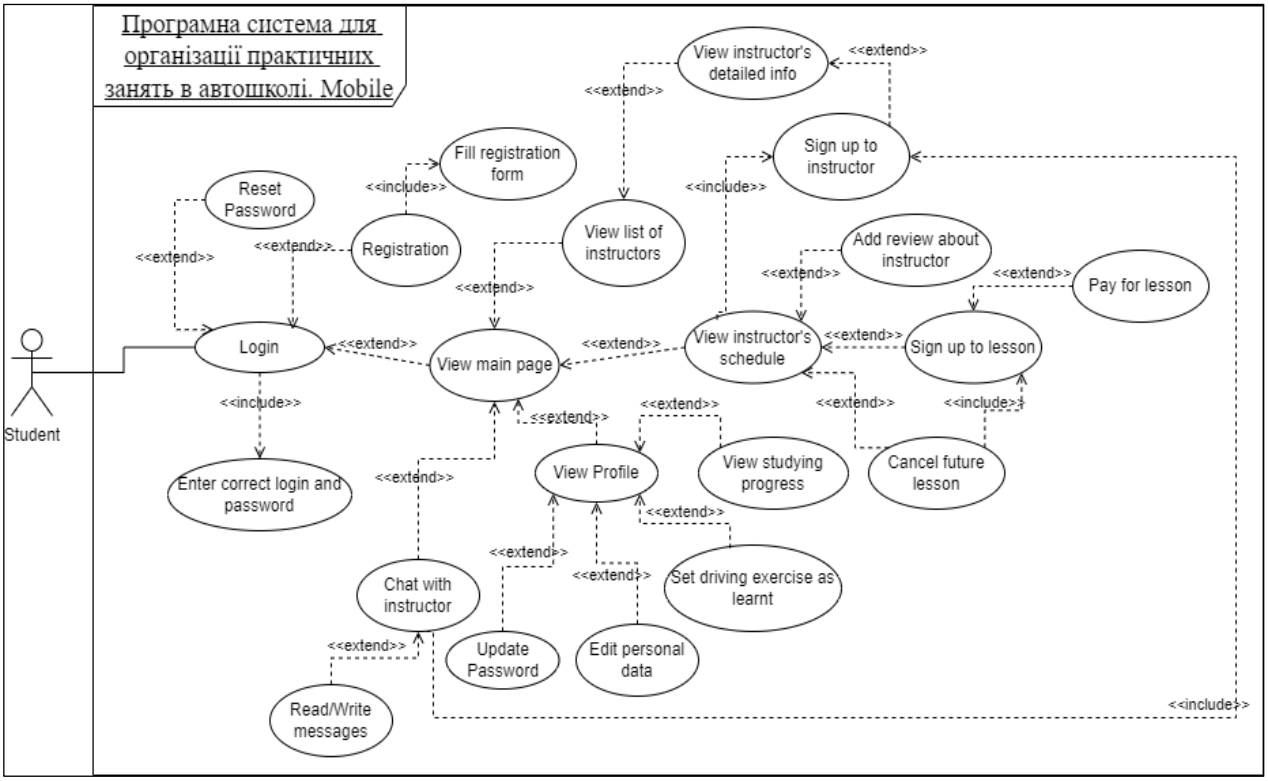


Рисунок 3.3– Use Case діаграма для користувача-студента

Для користувача доступні такі функції: реєстрація, авторизація, перегляд та редагування власного профілю, перегляд інструкторів, та детальний перегляд кожного інструктора, перегляд інформації про машину на якій може проходити навчання, відправка запиту на прив’язку до інструктора, перегляд розкладу інструктора та запис на певне заняття, для того інструктора, до якого виконано прив’язку, додавання заняття до гугл календаря, перегляд прогресу навчання та відзначення вивчених навичок, можливість залишити відгуки про інструктора у якого вже займався хоча б 1 раз, та про його машину, перегляд статистики про навчання, чат з інструктором.

Для користувачів, що мають роль інструктора програмна система надає функціонал, а також шляхи доступу до певних функцій, продемонстровані на діаграмі прецедентів (див. рисунок 3.4)

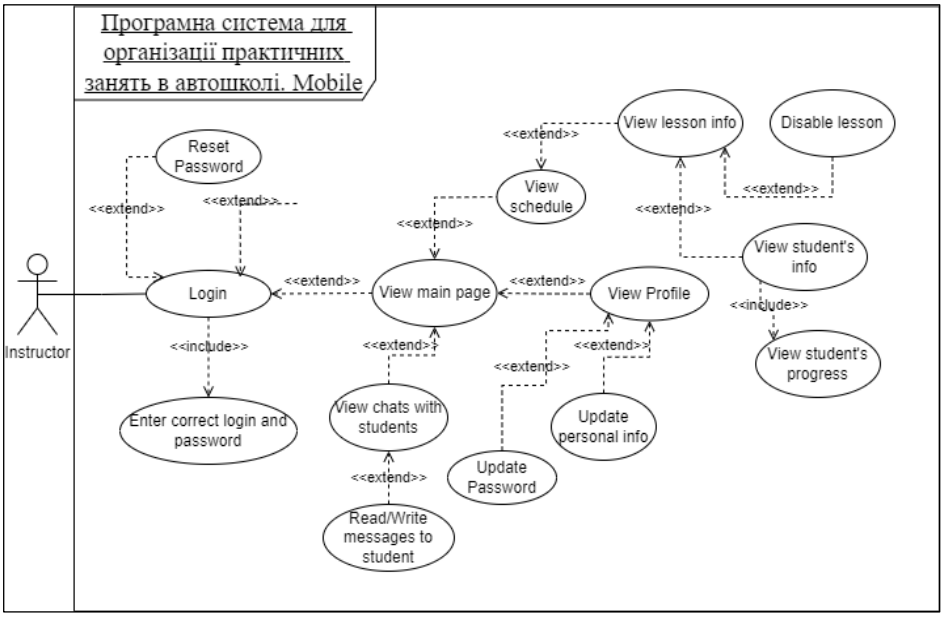


Рисунок 3.4 – Use-case

Загалом інструктору доступні такі можливості: авторизація, перегляд своїх студентів, перегляд власного розкладу, можливість зробити заняття неактивним, якщо на нього ще не записаний учень, змінити пароль, або відновити пароль, якщо старий було забуто, перегляд інформації про користувачів, які прив’язані до цього інструктора, включно з їх прогресом та кількістю проведених занять, чати зі студентами.

3.4 Розробка візуальної частини мобільного додатку

Розробка візуальної частини мобільного додатку для програмної системи автоматизації проведення практичних занять в автошколі, буде проведена з використанням .NET MAUI [3]. MAUI – це сучасний фреймворк для розробки мобільних додатків, який надає величезні можливості для створення інтерфейсів користувача. Однією з переваг даного фреймворку є його здатність забезпечувати розробку для різних платформ, таких як Android, iOS і Windows, з використанням одного і того ж коду.

У .NET MAUI візуальна частина додатку будується за допомогою віджетів. Віджети є основною одиницею побудови інтерфейсу користувача і представляють собою різні елементи, такі як кнопки, тексти, текстильні поля, таблиці тощо. Вони можуть бути статичними (без стану) або динамічними (зі збереженням стану), залежно від потреб застосунку.

Крім того, .NET MAUI надає розробникам можливість використовувати різні менеджери стану, такі як BloC, Redux, GetX тощо, для ефективного управління станом додатку та забезпечення швидкої реакції на зміни даних та взаємодію з користувачем.

Створення інтерфейсу користувача для програмної системи автоматизації проведення практичних занять в автошколі включає в себе:

- статичні елементи інтерфейсу: Такі як заголовки, кнопки, тексти, зображення, які відображають постійну інформацію та функціональність додатку;
- динамічні елементи інтерфейсу: Наприклад, списки, які відображають динамічні дані, такі як список студентів або розклад занять;
- форми і валідація: Елементи, які дозволяють користувачам вводити та надсилати дані, такі як форми реєстрації або введення результатів практичних занять;
- адаптивний дизайн: Врахування різних розмірів екранів та орієнтацій пристроїв для забезпечення оптимального відображення інтерфейсу;

– навігація: Створення навігаційних шляхів для користувачів, щоб вони могли легко переміщатися між різними частинами додатку.

Розглянемо процес запису на урок та його оплати, відповідну діаграму активності наведено на рисунку 3.5. Можна зазначити, що даний процес не містить багато розгалужень, а отже він є послідовним та зрозумілим для користувачів. Мінімальні розгалуження наявні на діаграмі дозволяють повторити невдалі спроби, що також позитивно впливає на користувацький досвід.

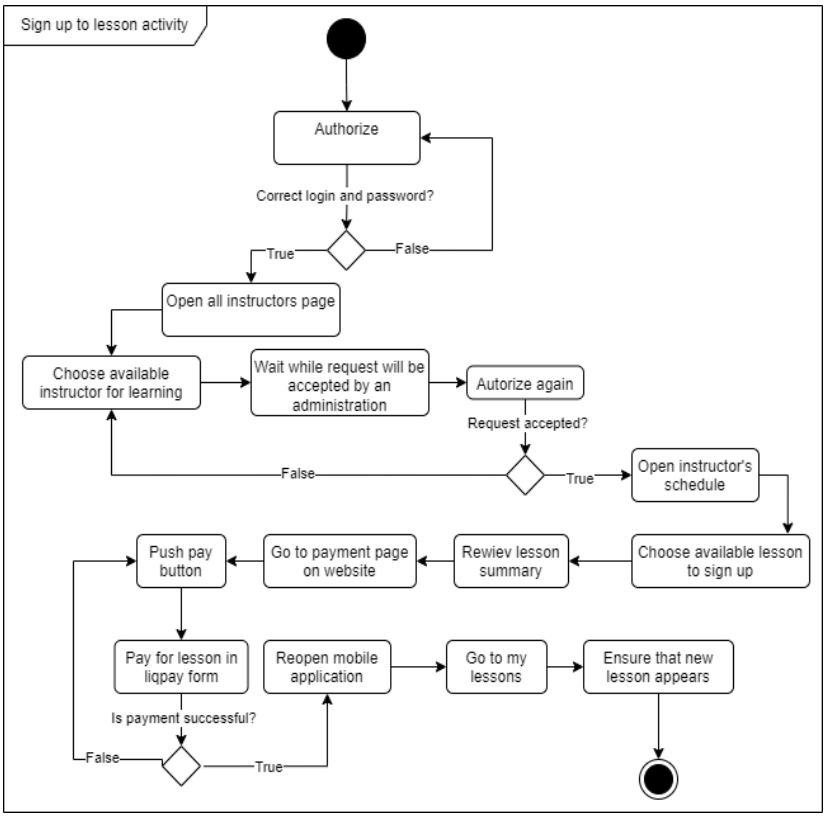


Рисунок 3.5 – Діаграма активності для процесу запису на урок та його оплати

Наведемо схему процесу реєстрації та подальшого логіну студента автошколи (див. рисунок 3.6).

На діаграмі послідовності представлено 4 об’єкти, які взаємодіють між собою: користувач, інтерфейс мобільного застосунку, серверна частина системи та база даних. Натискання на екрані обробляються інтерфейсом застосунку, який в свою чергу за необхідності відправляє запити на сервер, де дані можуть збережені до бази даних, або навпаки отримані з бази даних, оброблені та

повернені до мобільного застосунку, де результат запиту буде відображено користувачу.

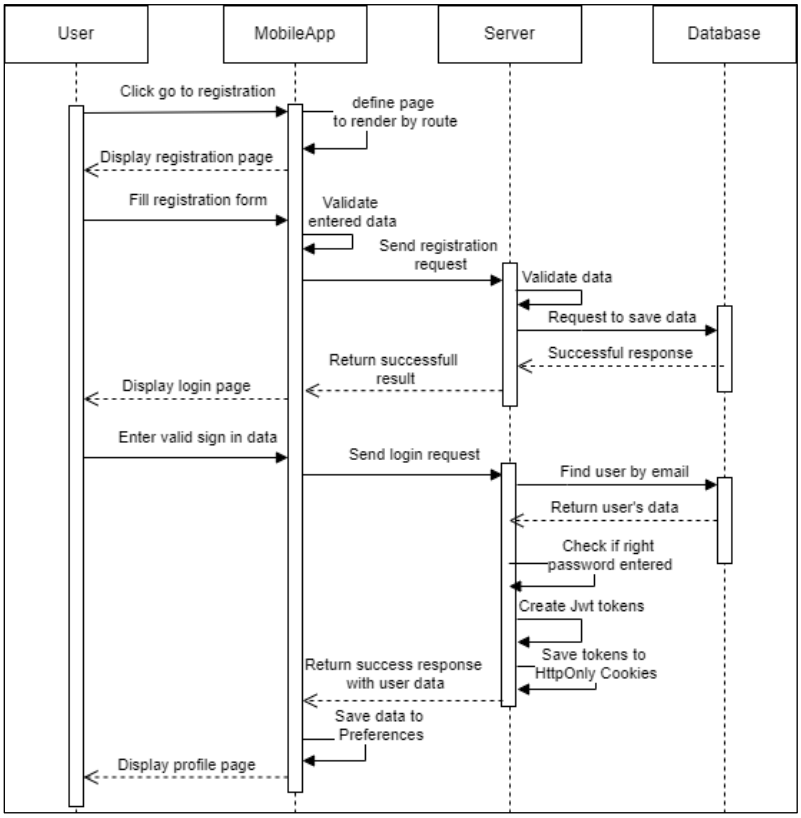


Рисунок 3.6 – Діаграма послідовності для процесу реєстрації та подальшої авторизації користувача

Використання .NET MAUI дозволить нам створити мобільний додаток з високоякісним інтерфейсом користувача, який буде відповідати потребам користувачів автошколи та забезпечить зручну та ефективну роботу з програмним забезпеченням.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Вибір засобів програмної реалізації

Перш за все ми визначили, що створюваний мобільний додаток має відповідати наступним критеріям: кросплатформеність, зручний інтерфейс, безпека, швидкодія та ефективність, масштабованість, легка підтримка та оновлення. Відповідно до цих критеріїв для проведення розробки ми обрали фреймворк .NET Maui.

.NET MAUI – це сучасний кросплатформенний фреймворк, розроблений для створення нативних мобільних і десктопних додатків. Використовуючи XAML та мову програмування C#, .NET MAUI дозволяє створювати додатки для Android та iOS з однією спільною кодовою базою, що значно зменшує витрати на розробку та обслуговування. Фреймворк забезпечує високу продуктивність, надаючи інструменти для оптимізації додатків, а також підтримує сучасні підходи до розробки, такі як MVVM (Model-View-ViewModel), що сприяє розділенню логіки додатка і його представлення, покращуючи його підтримку та масштабованість.

Код застосунку, написаний з використанням .NET MAUI, компілюється в нативний код для Android або iOS, що забезпечує високу продуктивність і ефективне використання апаратних ресурсів пристрою [3]. Це дозволяє додатку працювати швидко та плавно, надаючи користувачам нативний досвід використання з оптимальним відгуком і інтеграцією з функціями операційної системи. Також компіляція в нативний код покращує безпеку додатку, оскільки зменшує можливість реверс-інжинірингу та підвищує загальну стабільність роботи.

Для збереження даних та налаштувань користувача ми використовуємо механізм Preferences [5], який забезпечує зручне та ефективне зберігання інформації локально на пристрої, дозволяючи додатку зберігати налаштування та інші важливі дані навіть при відсутності з'єднання з сервером.

Розробка нашого мобільного додатку буде проводитися в середовищі Visual Studio 2022, яке є ідеальним вибором для розробки з використанням .NET MAUI.

Visual Studio 2022 надає потужні інструменти, серед яких можна виділити функції Hot Reload та XAML Live Preview. Hot Reload дозволяє вносити зміни в код додатка під час його виконання, без необхідності перезавантаження додатка або втрати поточного стану. Це значно прискорює процес розробки, оскільки дозволяє миттєво бачити результати змін, виправляти помилки та оптимізувати код в режимі реального часу. XAML Live Preview, у свою чергу, надає можливість візуального попереднього перегляду інтерфейсу користувача під час написання XAML-коду. Це дає змогу бачити, як виглядатимуть наші зміни в інтерфейсі безпосередньо в середовищі розробки, що підвищує точність і ефективність роботи з UI-компонентами.

Для реалізації цього мобільного додатку ми використали N-layer архітектуру та підхід MVVM. Фінальна структура проекту наведена на рисунку 4.1.

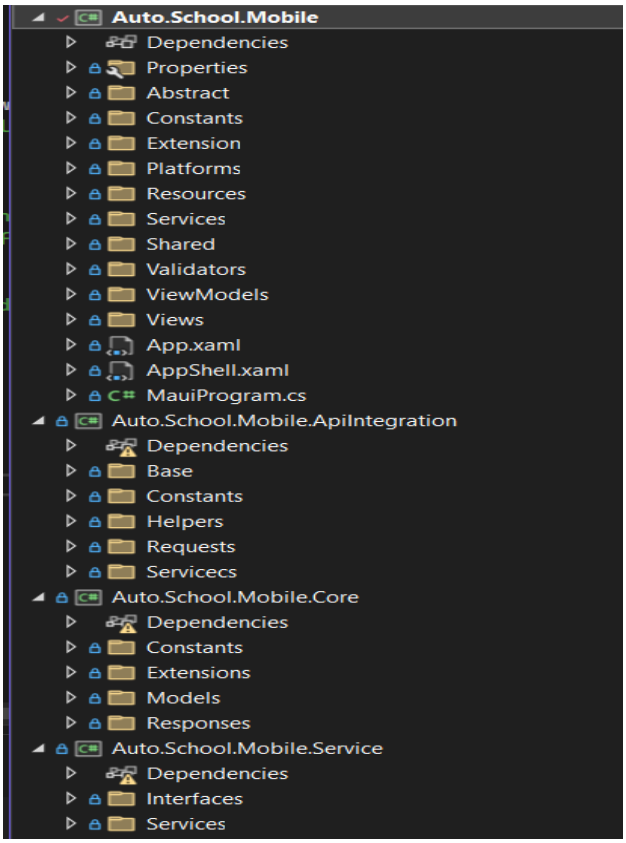


Рисунок 4.1 – Структура проекту

Програмний код застосунку складається з наступних шарів:

- шар представлення: У цьому шарі визначені сторінки, в'ю-моделі, а також допоміжні сервіси, наприклад, для динамічного перекладу повідомлень в залежності від обраної мови;
- шар сервісів: Цей шар містить інтерфейси та їх реалізацію, до яких звертається шар представлення. Шар сервісів відповідає за здійснення запитів до шару запитів до API, а також за обробку даних до та після запитів, щоб надати їх шарам представлення та запитів до API у необхідній формі;
- шар доступу до даних: Шар сервісів передає дані у потрібному форматі до цього шару, який виконує відповідні запити до серверу. Отримані результати передаються назад до шару сервісів для подальшої обробки. Це можуть бути запити на отримання даних, або на створення, редагування або видалення. До шару представлення повертаються як успішні результати запитів, так і ті, що повернули провальний результат, таким чином користувача буде проінформовано, що виконати запит не вдалося;
- спільний шар: Містить компоненти, які можуть використовуватися декількома іншими шарами, наприклад, класи моделей а також класи, що визначають відповіді від серверу, що використовуються на рівнях представлення, сервісів та запитів до серверу.

Використання моделі MVVM (Model-View-ViewModel) забезпечує чітке розділення логіки бізнесу, представлення даних та взаємодії між ними. Це покращує підтримку та розширення функціональності додатку та дозволяє ефективно керувати відображенням даних та реагувати на зміни в системі.

Для прикладу наведемо нижче код однієї з view-model, а саме для сторінки профілю студента (StudentProfileViewModel):

```
namespace Auto.School.Mobile.ViewModels
{
    public partial class StudentProfileViewModel : BaseViewModel, INotifyPropertyChanged
    {
```

```
        private readonly IStudentService _studentService;
        private readonly IInstructorService _instructorService;
        private readonly ISharedService _sharedService;
        private readonly IPopupService _popupService;
        private readonly ILessonService _lessonService;

        public StudentProfileViewModel(IStudentService
studentService, IInstructorService instructorService, ISharedService
sharedService, Abstract.IPopupService popupService, ILessonService
lessonService)
        {
            _studentService = studentService;
            _instructorService = instructorService;
            _sharedService = sharedService;
            _popupService = popupService;
            _lessonService = lessonService;
            _ = LoadStudent();
        }

        private async Task LoadStudent(){...}

        private async Task LoadInstructor(string instructorId){...}

        [ObservableProperty]
        private int numberPassedLessons;

        [ObservableProperty]
        private InstructorModel instructor;

        [ObservableProperty]
        private StudentModel student;

        [ObservableProperty]
        private bool isError = false;

        [ObservableProperty]
        private string errorMesssage = string.Empty;

        [ObservableProperty]
        private bool isInstructorRequestSent;

        [ObservableProperty]
        private string instructorMessage;

        [ObservableProperty]
        private bool isInstructorRequestAccepted;

        [ObservableProperty]
        private bool isViewInstructorsVisible;

        [ObservableProperty]
        private bool isInstructorMessageVisible;

        [ObservableProperty]
        private int numberPassedSkills;

        private bool isLoading = true;
```

```
        public bool IsLoading
        {
            get { return isLoading; }
            set
            {
                isLoading = value;
                OnPropertyChanged(nameof(IsLoading));
                OnPropertyChanged(nameof(IsNotLoading));
            }
        }

        public bool IsNotLoading
        {
            get { return !isLoading; }
            private set { }
        }

        [RelayCommand]
        public async Task UpdatePhoto(){...}

        private async Task<Stream> PickImage(){...}

        [RelayCommand]
        public async Task GoToInstructors(){...}

        [RelayCommand]
        public async Task GoToSchedule(){...}

        [RelayCommand]
        public async Task GoToLessons(){...}

        [RelayCommand]
        public async Task OpenDrivingSills(){...}

        [RelayCommand]
        public async Task OpenChat(){...}

        [RelayCommand]
        public async Task ViewDetailedInfo(){...}

        [RelayCommand]
        public async Task UpdatePassword(){...}

        [RelayCommand]
        public async Task UpdateInfo(){...}
    }
}
```

У цьому класі наявні властивості, що визначають дані, які можуть бути відображені на сторінці, також у view-model наявні команди, що представляють собою функції і призначені для виклику їх виконання по натисканню користувача. При ініціалізації даний клас здійснює запит на отримання необхідних даних з шару представлення, та у разі успішного результату виконання зберігає

отриманий екземпляр класу у своїй властивості відповідного типу, а саме класу моделі студента (StudentModel).

Між сторінкою та відповідним класом view-model виконується прив'язка даних, це дозволяє зв'язати властивості view-model та властивості сторінки таким чином, щоб зміни у властивостях view-model відображалися на сторінці. Для спрощення цього завдання ми використали пакет Community.Toolkit.Mvvm, що дозвляє позначити визначені у view-model як BindableProperty. Для сповіщення сторінки про зміну властивостей у view-model використано інтерфейс INotifyPropertyChanged.

Для прив'язки даних з view-model до сторінки, в якості значення властивості сторінки BindingContext встановлено екземпляр класу StudentProfileViewModel. Відповідний код наведемо нижче:

```
public partial class StudentProfile : ContentPage
{
  public                          StudentProfile(StudentProfileViewModel
studentProfileViewModel)
  {
     InitializeComponent();
     BindingContext = studentProfileViewModel;
  }
}
```

Код сторінки профіля користувача (StudentProfilePage), яка отримує значення з відповідних властивостей екземпляру класу StudentProfileViewModel, а також за допомогою прив'язки даних викликає визначені команди наведемо нижче:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Auto.School.Mobile.Views.StudentProfile"
             xmlns:resx="clr-
namespace:Auto.School.Mobile.Resources.Strings"
             Title="{extension:Translate
Key=StudentProfile_MyProfile}"
             xmlns:extension="clr-
namespace:Auto.School.Mobile.Extension">

    <ContentPage.Resources>
```

```
            <Style TargetType="Label">
                <Setter Property="TextColor" Value="Black" />
                <Setter Property="FontSize" Value="Medium"/>
                <Setter Property="Margin" Value="0,0,0,5" />
                <Setter                          Property="HorizontalOptions"
Value="StartAndExpand"/>
            </Style>
            <Style TargetType="Border">
                <Setter Property="Margin" Value="10,10,10,30" />
                <Setter Property="Padding" Value="5,5,5,15" />
            </Style>
        </ContentPage.Resources>
        <ContentPage.Content>
            <ScrollView>
                <StackLayout Margin="0, 0, 0, 10">
                    <ActivityIndicator   IsRunning="{Binding   IsLoading}"
IsVisible="{Binding IsLoading}" VerticalOptions="CenterAndExpand"/>
                    <Border       IsVisible="{Binding       IsNotLoading}"
BackgroundColor="Transparent">
                        <Border.StrokeShape>
                            <RoundRectangle CornerRadius="40,0,0,40" />
                        </Border.StrokeShape>
                        <StackLayout>
                            <VerticalStackLayout  Margin="10"  Padding="0"
>
                                <Frame                     CornerRadius="20"
MaximumHeightRequest="200" MaximumWidthRequest="200" Padding="0" Margin="0"
HasShadow="False"  BackgroundColor="Transparent">
                                    <Image                 Source="{Binding
Student.PhotoUrl}"   Aspect="AspectFill"   HorizontalOptions="FillAndExpand"
VerticalOptions="FillAndExpand" Margin="0"/>
                                </Frame>
                                <Button         Text="{extension:Translate
Key=StudentProfile_UpdatePhoto}"   Command="{Binding   UpdatePhotoCommand}"
HorizontalOptions="CenterAndExpand"               MaximumWidthRequest="150"
Background="#3D6DFF"         TextColor="White"         FontAttributes="Bold"
FontSize="Medium"          Margin="0,10"          LineBreakMode="WordWrap"
TextTransform="None"/>
                            </VerticalStackLayout>
                            <Border                BackgroundColor="White"
Stroke="#3D6DFF" StrokeThickness="2">
                                <Border.StrokeShape>
                                    <RoundRectangle CornerRadius="20" />
                                </Border.StrokeShape>
                                <Border.Shadow>
                                    <Shadow Brush="#3D6DFF" Opacity="0.5"
Offset="13,13" />
                                </Border.Shadow>
                                <StackLayout                    Padding="5"
HorizontalOptions="StartAndExpand">
                                    <Label      Text="{extension:Translate
Key=StudentProfile_PersonalTitle}"      HorizontalOptions="CenterAndExpand"
FontAttributes="Bold" FontSize="Large" Margin="0,5,0,10"/>
                                    <!-- User Data Section -->
                                    <StackLayout
Orientation="Horizontal">
```

```
                                        <Grid
RowDefinitions="*,*,*,*,*,*,*" ColumnDefinitions="*,*,*" ColumnSpacing="2"
RowSpacing="2" HorizontalOptions="CenterAndExpand" >
                                            <Label        Grid.Column="0"
Grid.Row="0"    Text="{extension:Translate    Key=StudentProfile_FirstName}"
HorizontalOptions="StartAndExpand" FontAttributes="Bold" />
                                            <Label        Grid.Column="1"
Grid.Row="0"  Grid.ColumnSpan="2" Text="{Binding Student.Name}" />

                                            <Label        Grid.Column="0"
Grid.Row="1"    Text="{extension:Translate    Key=StudentProfile_LastName}"
HorizontalOptions="StartAndExpand" FontAttributes="Bold"  />
                                            <Label        Grid.Column="1"
Grid.Row="1" Grid.ColumnSpan="2" Text="{Binding Student.Surname}" />

                                            <Label        Grid.Column="0"
Grid.Row="2"     Text="{extension:Translate     Key=StudentProfile_Email}"
HorizontalOptions="StartAndExpand" FontAttributes="Bold"  />
                                            <Label        Grid.Column="1"
Grid.Row="2" Grid.ColumnSpan="2" Text="{Binding Student.Email}" />

                                            <Label        Grid.Column="0"
Grid.Row="3"         Grid.ColumnSpan="2"         Text="{extension:Translate
Key=StudentProfile_BirthdayDate}"        HorizontalOptions="StartAndExpand"
FontAttributes="Bold" />
                                            <Label        Grid.Column="2"
Grid.Row="3"               Grid.ColumnSpan="1"               Text="{Binding
Student.UserData.DateOfBirth, StringFormat='{0:dd-MM-yyyy}'}" />

                                            <Label        Grid.Column="0"
Grid.Row="4"   Text="{extension:Translate  Key=StudentProfile_PhoneNumber}"
HorizontalOptions="StartAndExpand" FontAttributes="Bold" />
                                            <Label        Grid.Column="1"
Grid.Row="4"  Grid.ColumnSpan="2" Text="{Binding Student.UserData.Phone}"/>

                                            <Label        Grid.Column="0"
Grid.Row="5"         Grid.ColumnSpan="2"         Text="{extension:Translate
Key=StudentProfile_DrivingCategory}"     HorizontalOptions="StartAndExpand"
FontAttributes="Bold" />
                                            <Label        Grid.Column="2"
Grid.Row="5"   Grid.ColumnSpan="1" Text="{Binding Student.VehicleCategory}"
/>

                                            <Label        Grid.Column="0"
Grid.Row="6"         Grid.ColumnSpan="2"         Text="{extension:Translate
Key=StudentProfile_City}"                HorizontalOptions="StartAndExpand"
FontAttributes="Bold" />
                                            <Label        Grid.Column="2"
Grid.Row="6"  Grid.ColumnSpan="1" Text="{Binding Student.City.Name}"/>
                                        </Grid>
                                    </StackLayout>
                                    <HorizontalStackLayout Margin="0,10">
                                        <Button
Text="{extension:Translate                   Key=StudentProfile_UpdateInfo}"
MaximumWidthRequest="150"                  VerticalOptions="CenterAndExpand"
HorizontalOptions="CenterAndExpand"  Command="{Binding  UpdateInfoCommand}"
BackgroundColor="#3D6DFF"          TextColor="White"          Margin="5,2"
```

```
LineBreakMode="WordWrap"     TextTransform="None"     FontAttributes="Bold"
FontSize="Medium"/>
                                        <Button
Text="{extension:Translate              Key=StudentProfile_UpdatePassword}"
MaximumWidthRequest="150"                VerticalOptions="CenterAndExpand"
HorizontalOptions="CenterAndExpand"                          Command="{Binding
UpdatePasswordCommand}"     BackgroundColor="#3D6DFF"     TextColor="White"
Margin="5,2"          LineBreakMode="WordWrap"          TextTransform="None"
FontAttributes="Bold" FontSize="Medium"/>
                                    </HorizontalStackLayout>
                                </StackLayout>
                            </Border>
                            <Border                 BackgroundColor="White"
Stroke="#3D6DFF"   StrokeThickness="2"   Padding="10"   IsVisible="{Binding
IsInstructorMessageVisible}">
                                <Border.StrokeShape>
                                    <RoundRectangle CornerRadius="20" />
                                </Border.StrokeShape>
                                <Border.Shadow>
                                    <Shadow Brush="#3D6DFF" Opacity="0.5"
Offset="13,13" />
                                </Border.Shadow>
                                <StackLayout>
                                    <Label      Text="{extension:Translate
Key=StudentProfile_InstructorRequestStatusTitle}"     FontAttributes="Bold"
HorizontalOptions="CenterAndExpand"/>
                                    <Label                   Text="{Binding
InstructorMessage}" HorizontalOptions="CenterAndExpand"/>
                                    <Button            IsVisible="{Binding
IsViewInstructorsVisible}"    Command="{Binding    GoToInstructorsCommand}"
Text="{extension:Translate Key=StudentProfile_ViewInstructors}" Margin="10"
HorizontalOptions="CenterAndExpand"                MaximumWidthRequest="150"
Background="#3D6DFF"         TextColor="White"         FontAttributes="Bold"
FontSize="Medium" LineBreakMode="WordWrap" TextTransform="None"/>
                                </StackLayout>
                            </Border>
                            <Border                 BackgroundColor="White"
Stroke="#3D6DFF"   StrokeThickness="2"   Padding="10"   IsVisible="{Binding
IsInstructorRequestAccepted}">
                                <Border.StrokeShape>
                                    <RoundRectangle CornerRadius="20" />
                                </Border.StrokeShape>
                                <Border.Shadow>
                                    <Shadow Brush="#3D6DFF" Opacity="0.5"
Offset="13,13" />
                                </Border.Shadow>
                                <StackLayout>
                                    <Label      Text="{extension:Translate
Key=StudentProfile_LessonsStatisticTitle}"             FontAttributes="Bold"
HorizontalOptions="CenterAndExpand"       VerticalOptions="CenterAndExpand"
FontSize="Large"/>
                                    <Grid      Padding="5"      Margin="5"
ColumnDefinitions="*,*,*,*,*,*,*"  RowDefinitions="*,*,*"  ColumnSpacing="2"
RowSpacing="10">
                                        <Label                 Grid.Row="0"
Grid.Column="0"        Grid.ColumnSpan="6"        Text="{extension:Translate
Key=StudentProfile_NumberLessonsLabel}"
HorizontalOptions="StartAndExpand"/>
```

```
                                        <Label                  Grid.Row="0"
Grid.Column="6"             Text="{Binding             NumberPassedLessons}"
HorizontalOptions="StartAndExpand" VerticalOptions="Center"/>

                                        <Label                  Grid.Row="1"
Grid.Column="0"        Grid.ColumnSpan="6"        Text="{extension:Translate
Key=StudentProfile_ProgressLabel}" HorizontalOptions="StartAndExpand"/>
                                        <Label                  Grid.Row="1"
Grid.Column="6"        Text="{Binding        Student.DrivingSkillsProgress}"
HorizontalOptions="StartAndExpand" VerticalOptions="Center"/>

                                        <Label                  Grid.Row="2"
Grid.Column="0"        Grid.ColumnSpan="6"        Text="{extension:Translate
Key=StudentProfile_NumberLearnedExercises}"
HorizontalOptions="StartAndExpand"/>
                                        <Label                  Grid.Row="2"
Grid.Column="6"             Text="{Binding             NumberPassedSkills}"
HorizontalOptions="StartAndExpand" VerticalOptions="Center"/>
                                    </Grid>

                                    <StackLayout
Orientation="Horizontal">
                                        <Button
Text="{extension:Translate           Key=StudentProfile_GoToLessonsButton}"
Command="{Binding GoToLessonsCommand}"  HorizontalOptions="CenterAndExpand"
MaximumWidthRequest="150"       Background="#3D6DFF"       TextColor="White"
FontAttributes="Bold"       FontSize="Medium"       LineBreakMode="WordWrap"
TextTransform="None"/>
                                        <Button
Text="{extension:Translate           Key=StudentProfile_OpenDrivingSkills}"
Command="{Binding                                  OpenDrivingSillsCommand}"
HorizontalOptions="CenterAndExpand"                MaximumWidthRequest="150"
Background="#3D6DFF"       TextColor="White"       LineBreakMode="WordWrap"
FontAttributes="Bold" FontSize="Medium" TextTransform="None"/>
                                    </StackLayout>
                                </StackLayout>
                            </Border>
                            <Border                 BackgroundColor="White"
Stroke="#3D6DFF"           StrokeThickness="2"           IsVisible="{Binding
IsInstructorRequestSent}">
                                <Border.StrokeShape>
                                    <RoundRectangle CornerRadius="20" />
                                </Border.StrokeShape>
                                <Border.Shadow>
                                    <Shadow Brush="#3D6DFF" Opacity="0.5"
Offset="13,13" />
                                </Border.Shadow>
                                <StackLayout Margin="5">
                                    <Label      Text="{extension:Translate
Key=StudentProfile_InstructorInfoTitle}"             FontAttributes="Bold"
HorizontalOptions="CenterAndExpand" FontSize="Large"/>
                                    <StackLayout
Orientation="Horizontal">
                                        <Grid
HorizontalOptions="CenterAndExpand"                RowDefinitions="*,*,*,*"
ColumnDefinitions="*,*" ColumnSpacing="20" RowSpacing="5">
```

```
                                        <Frame          Grid.Column="0"
Grid.Row="0"       Grid.RowSpan="4"       Padding="0"       HasShadow="False"
BackgroundColor="Transparent">
                                            <Image    Source="{Binding
Instructor.PhotoUrl}" Aspect="AspectFill" HorizontalOptions="FillAndExpand"
VerticalOptions="FillAndExpand"/>
                                        </Frame>
                                        <Label          Grid.Column="1"
Grid.Row="0"               Text="{Binding              Instructor.FullName}"
HorizontalOptions="CenterAndExpand" MaximumWidthRequest="150" />
                                        <Button         Grid.Column="1"
Grid.Row="1"  Text="{extension:Translate  Key=StudentProfile_DetailedInfo}"
Command="{Binding      ViewDetailedInfoCommand}"      FontAttributes="Bold"
FontSize="Small"                          HorizontalOptions="CenterAndExpand"
MaximumWidthRequest="150"       Background="#3D6DFF"       TextColor="White"
LineBreakMode="WordWrap" TextTransform="None"/>
                                        <Button         Grid.Column="1"
Grid.Row="2"  Text="{extension:Translate  Key=StudentProfile_GoToSchedule}"
Command="{Binding        GoToScheduleCommand}"        FontAttributes="Bold"
FontSize="Small"                          HorizontalOptions="CenterAndExpand"
MaximumWidthRequest="150"       Background="#3D6DFF"       TextColor="White"
LineBreakMode="WordWrap" TextTransform="None"/>
                                        <Button         Grid.Column="1"
Grid.Row="3"        IsVisible="{Binding        IsInstructorRequestAccepted}"
Text="{extension:Translate                 Key=StudentProfile_SendMessage}"
FontAttributes="Bold"  FontSize="Small"  Command="{Binding  OpenChatCommand}"
HorizontalOptions="CenterAndExpand"              MaximumWidthRequest="150"
Background="#3D6DFF"       TextColor="White"       LineBreakMode="WordWrap"
TextTransform="None"/>
                                    </Grid>
                                </StackLayout>
                            </StackLayout>
                        </Border>
                    </StackLayout>
                </Border>
            </StackLayout>
        </ScrollView>
    </ContentPage.Content>
</ContentPage>
```

Ми також використовували Dependency Injection (DI), що дозволило значно покращити модульність та тестованість додатку [9]. Використання DI спрощує керування залежностями між компонентами системи, дозволяючи легко змінювати реалізації інтерфейсів без необхідності модифікації коду, який їх використовує. Завдяки DI ми змогли підвищити рівень абстракції та забезпечити більш чистий та підтримуваний код, що значно покращує процес розробки та обслуговування додатку.

Так, до прикладу було реалізовано механізм передачі об'єктів між сторінками додатку. Було визначено інтерфейс ISharedService, з двома методами для зберігання об'єкта та для його отримання:

```
namespace Auto.School.Mobile.Abstract
{
    public interface ISharedService
    {
        public void Add<T>(string key, T value) where T : class;
        public T? GetValue<T>(string key) where T : class;
    }
}
```

Реалізація інтерфейсу класом SharedService наведено нижче:

```
namespace Auto.School.Mobile.Services
{
    public class SharedService : ISharedService
    {
        private Dictionary<string, object> DTODict { get; set; } =
[];

        public void Add<T>(string key, T value) where T : class
        {
            if (DTODict.ContainsKey(key))
            {
                DTODict[key] = value;
            }
            else
            {
                DTODict.Add(key, value);
            }
        }

        public T? GetValue<T>(string key) where T : class
        {
            return DTODict.ContainsKey(key) ? DTODict[key] as T :
null;
        }
    }
}
```

Після реалізації інтерфейсу в класі Maui.Program реєструється служба з типом Singleton, це означає, що вона буде існувати весь час життя застосунку.

Даний інтерфейс використовуються у view-model для встановлення необхідних даних, що будуть потрібні на наступній сторінці, й, відповідно, для їх отримання після переходу. Приклад коду наведемо нижче:

```
[RelayCommand]
public async Task GoToSchedule()
{
    _sharedService.Add("InstructorId", Instructor.Id);
    await
Shell.Current.GoToAsync($"{nameof(InstructorScheduleStudentPage)}");
}
private async Task LoadInstructor()
{
    var                          instructorId                          =
_sharedService.GetValue<string>("InstructorId");

    if (instructorId is null)
    {
        string  instructorIdJson  =  Preferences.Get("MyInstructorId",
string.Empty);
        if (!string.IsNullOrEmpty(instructorIdJson))
        {
            instructorId                                                =
JsonConvert.DeserializeObject<string>(instructorIdJson);
        }
    }

    var response = await _instructorService.GetOne(instructorId!);

    if   (response   is   null   ||   string.Compare(response.Status,
ResponseStatuses.Fail, true) == 0)
    {
        IsError = true;
        ErrorMessage                                                    =
AppErrorMessagesConstants.FailedToLoadInstuctor;
        return;
    }

    IsError = false;
    Instructor = response.Instructor;
}
```

Завдяки тому, що реалізацію інтерфейсу ISharedService було зареєстровано через використання залежностей, у конструкторі класа StudentProfileViewModel нам достатньо передати даний інтерфейс, щоб мати можливість ініціалізувати поле з типом ISharedService, та використовувати функціонал реалізований у класі SharedService:

```
    public partial class StudentProfileViewModel : BaseViewModel,
INotifyPropertyChanged
    {
    private readonly IStudentService _studentService;
    private readonly IInstructorService _instructorService;
    private readonly ISharedService _sharedService;
    private readonly IPopupService _popupService;
    private readonly ILessonService _lessonService;

    public StudentProfileViewModel(IStudentService studentService,
IInstructorService instructorService, ISharedService sharedService,
Abstract.IPopupService popupService, ILessonService lessonService)
    {
        _studentService = studentService;
        _instructorService = instructorService;
        _sharedService = sharedService;
        _popupService = popupService;
        _lessonService = lessonService;
        _ = LoadStudent();
    }

    [RelayCommand]
    public async Task GoToSchedule()
    {
        _sharedService.Add("InstructorId", Instructor.Id);
        await
Shell.Current.GoToAsync($"{nameof(InstructorScheduleStudentPage)}");
    }
```

У шарі доступу до даних було створено універсальні інтерфейси для запитів до бекенд серверу для таких операцій: отримання, збереження, зміни та видалення, та їх відповідна реалізація. Для прикладу наведемо код інтерфейсу IGetRequest для запиту на отримання даних:

```
public interface IGetRequest
{
    public Task<TResponse> ExecuteAsync<TResponse>(string url);
}
```

Також нижче наведемо код універсального класу GetRequest, що реалізує інтерфейс:

```
namespace Auto.School.Mobile.ApiIntegration.Base.Implementation
{
    public class GetRequest(IHttpClientService httpClientService) :
IGetRequest
    {
        private readonly IHttpClientService _httpClientService =
httpClientService;
```

```
            public async Task<TResponse> ExecuteAsync<TResponse>(
                string url)
            {
                var path = RoutesConstants.BaseUrl + url;

                try
                {
                    var response = await
_httpClientService.Client.GetAsync(path);

                    if (response.IsSuccessStatusCode)
                    {
                        var responseContent = await
response.Content.ReadAsStringAsync();
                        var responseData =
JsonConvert.DeserializeObject<TResponse>(responseContent);

                        if (responseData == null)
                        {
                            throw new Exception("Cannot Deserialize
object");
                        }

                        return responseData;
                    }
                    else
                    {
                        var responseContent = await
response.Content.ReadAsStringAsync();
                        var responseData =
JsonConvert.DeserializeObject<TResponse>(responseContent);
                        throw new HttpRequestException($"Cannot send get
request to: {path},Status code: {response.StatusCode}");
                    }
                }
                catch (Exception ex)
                {
                    throw new Exception($"An error occurred while
processing the GET request: {ex.Message}");
                }
            }
        }
    }
```

Як можемо побачити в інтерфейсі та його реалізації визначено універсальний метод, тобто метод оголошений з використанням параметрів типів. Такий підхід дозволяє нам використовувати єдиний код для надсилання всіх запитів на отримання даних, вказуючи який тип ми хочемо отримати безпосередньо під час виклику метода, що дозволяє значно зменшити кількість

однакового коду. Приклад виклику універсальних методів для отримання списку міст та для отримання інформації студентом про нього наведемо нижче:

```
public class CityRequests(IGetRequest getRequest) : ICityRequest
{
    private readonly IGetRequest _getRequest = getRequest;
    public async Task<AllCitiesResponse> GetAll()
    {
        var cityResponse = await
_getRequest.ExecuteAsync<AllCitiesResponse>(RoutesConstants.GetAllCities);

        if (cityResponse.Status is null)
        {
            cityResponse.Status = ResponseStatuses.Fail;
            cityResponse.ResponseData = [];
        }

        return cityResponse;
    }
}
```

З цікавих особливостей реалізації хочемо відмітити визначення інтерфейсу IHttpClientService та створення його реалізації у класі HttpClientService. Названий інтерфейс надає методи для надсилання http запитів до бекенд сервера. Головне призначення цього сервісу у тому, щоб забезпечити єдиний Http Context для всіх запитів під час роботи додатку. Необхідність у такій реалізації виникла через те, що на бекенді для надання доступу до закритих ендпоінтів ми використовуємо Http only cookies.

Для того, щоб це досягти в класі HttpClientService визначено приватне поле з типом HttpClient, що ініціалізується лише 1 раз у конструкторі цього класу. Одночасно з цим в класі MauiProgram реєструється залежність між інтерфейсом IHttpClientService та його реалізацією з типом singleton за допомогою dependency injection, що забезпечує створення лише одного екземпляру класа HttpClientService під час роботи застосунку. Код описаних інтерфейсу та класу наведемо нижче:

```
public interface IHttpClientService
  {
      public HttpClient Client { get; }
  }
```

```
    public class HttpClientService : IHttpClientService
    {
        private static readonly CookieContainer _cookieContainer = new
CookieContainer();
        public HttpClient Client { get; }

        public HttpClientService()
        {
            HttpClientHandler handler = new HttpClientHandler();
            handler.CookieContainer = _cookieContainer;
            handler.UseCookies = true;

            Client = new HttpClient(handler);
        }
    }
```

Програмний код для реєстрації відповідного сервісу:

```
    builder.Services.AddSingleton<IHttpClientService,
HttpClientService>();
```

Підсумовуючи можна виділити такі переваги використання N-layer архітектури разом з підходом MVVM та реєстрацією сервісів за допомогою dependency injection разом з використанням дженерік класів та інтерфейсів.

- чітке розділення обов'язків: Кожен шар відповідає за свою частину функціональності, що робить код більш організованим та легким для підтримки;
- покращена масштабованість: Можливість додавати нові функції або змінювати існуючі без впливу на інші частини системи;
- полегшене тестування: Окремі шари та компоненти можна тестувати незалежно, що підвищує якість коду та зменшує кількість помилок;
- можливість повторного використання коду: Спільний шар дозволяє використовувати загальні компоненти в різних частинах додатку, що зменшує дублікат коду та сприяє його повторному використанню;
- покращена взаємодія з користувачем: MVVM забезпечує інтуїтивно зрозумілий інтерфейс користувача, що дозволяє легко адаптувати додаток під потреби користувачів;

– гнучкість та підтримуваність: Dependency injection забезпечує легке керування залежностями та можливість швидкої зміни реалізацій без необхідності змінювати код, який їх використовує;
– підвищення абстракції: Використання дженерік класів та інтерфейсів дозволяє створювати більш гнучкі та повторно використовувані компоненти, що покращує структурованість та зменшує залежності між модулями;
– зменшення зв'язаності коду: Використання інверсії контролю знижує зв'язаність між компонентами, що робить систему більш модульною та легкою для підтримки.

Ці переваги забезпечують ефективну, масштабовану та підтримувану структуру додатку, що відповідає всім вимогам сучасної розробки мобільних застосунків.

4.2 Опис мобільної частини системи

Мобільний застосунок призначений для двох типів користувачів: студентів автошколи та інструкторів, які проводять уроки. Для обох типів користувачів наявний один додаток після входу до якого буде відображено відповідний інтерфейс.

Тож перша сторінка яку бачить користувач після відкриття застосунку це сторінка входу в застосунок (див. рисунок 4.2)

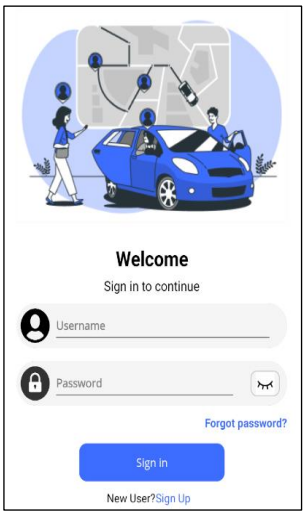


Рисунок 4.2 – Екран входу в застосунок

На цій сторінці користувач може або ввести логін та пароль, щоб увійти до застосунку, або перейти на сторінку реєстрації (див. рисунок 4.3).

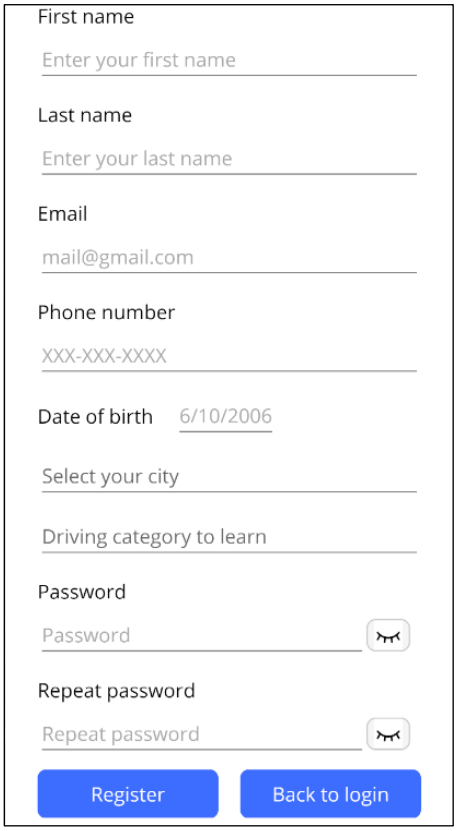


Рисунок 4.3 – Екран реєстрації студента

Зазначимо, що зареєструватися користувач може тільки як студент, інструкторів реєструють адміністратори автошколи через створений веб-застосунок. Після заповнення форми реєстрації залізними даними та натискання на кнопку зареєструватися користувача буде повернено на сторінку логічну, звідки він зможе виконати вхід в застосунок. Також зазначимо, що новому студенту на пошту приходить лист з проханням підтвердити свій емейл. Після успішного входу користувач опиниться на сторінці власного профілю.

Спочатку розглянемо всі сторінки які доступні студенту автошкол. У профілі у нього може бути декілька варіантів структури сторінки: для студента, який ще не обрав собі інструктора буде запропоновано перейти до списку інструкторів та обрати підходящого (див. Рисунок 4.4), тоді як вже записаний студент зможе побачити статистику своїх занять, де може перейти до сторінки пройдених занять або відкрити список вправ які треба освоїти, свого інструктора

та отримує можливості перейти до профілю інструктора, його розкладу або відкрити чат з інструктором (див. рисунок 4.5). Обидва типи студентів зможуть побачити свою персональну інформацію, завантажити нове фото профілю та оновити свої дані та пароль.

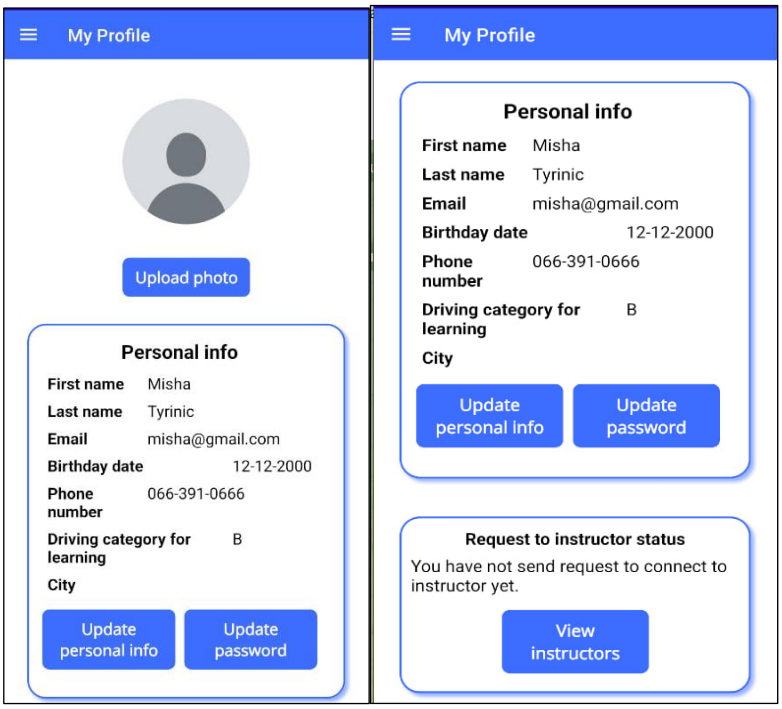


Рисунок 4.4 – Екран сторінки профілю студента без інструктора

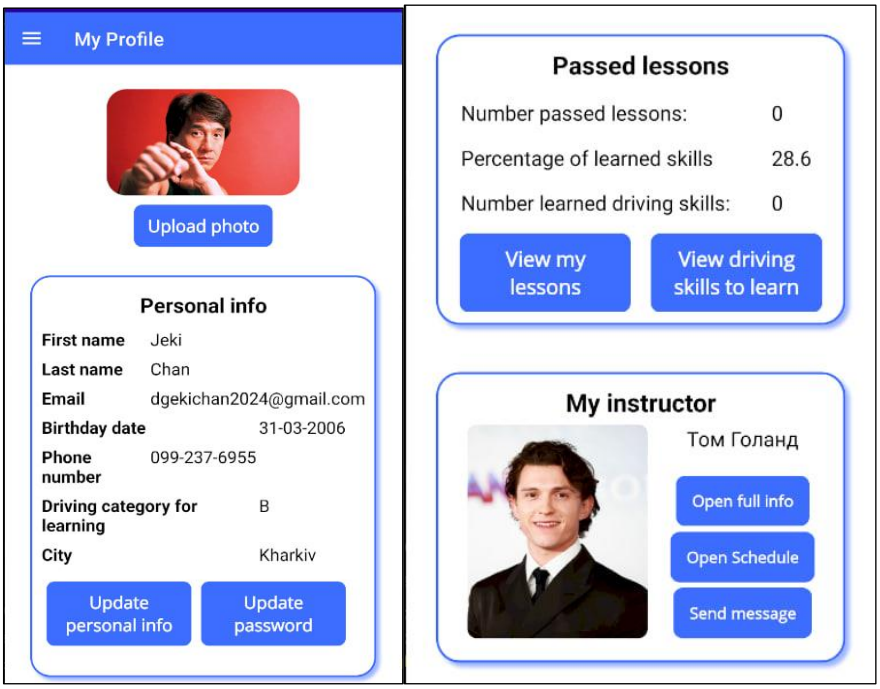


Рисунок 4.5 – Екран сторінки профілю студента закріпленого за інструктором

Натиснувши на кнопку для завантаження фото студенту буде запропоновано обрати фото з файлового менеджера телефону (див. рисунок 4.6). Після обрання він майже миттєво зможе побачити оновлений аватар на сторінці.

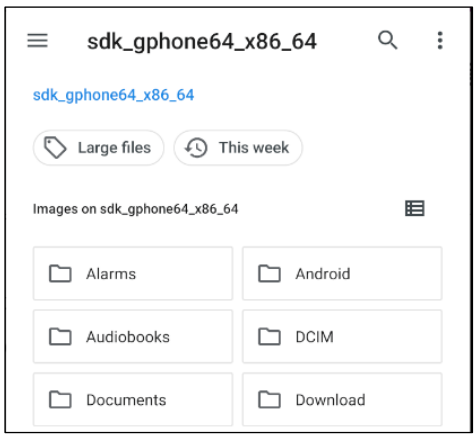


Рисунок 4.6 – Стандартна сторінка вибору фото

У разі натискання кнопки для оновлення даних або зміни паролю студент побачить спливаюче вікно з формою (див. рисунок 4.7 та рисунок 4.8), яку необхідно заповнити, та потім натиснути відповідну кнопку для оновлення.

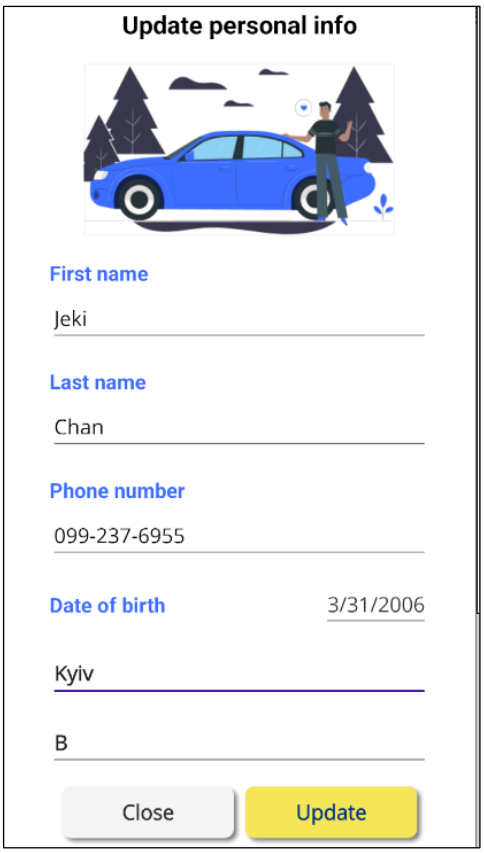


Рисунок 4.7 – Екран спливаючого вікна для оновлення даних

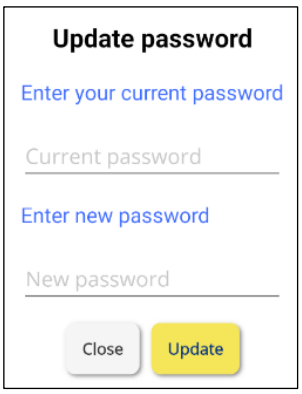


Рисунок 4.8 – Екран спливаючого вікна для оновлення паролю

Наступною можливістю є перехід до сторінки переліку моїх уроків, яка буде розглянута нижче, та перегляд вправ для вивчення (див. рисунок 4.9). Перегляд вправ відкривається у спливаючому вікні, студент має можливість відмічати вправу вивченою, щоб стежити за своїм навчальним прогресом.

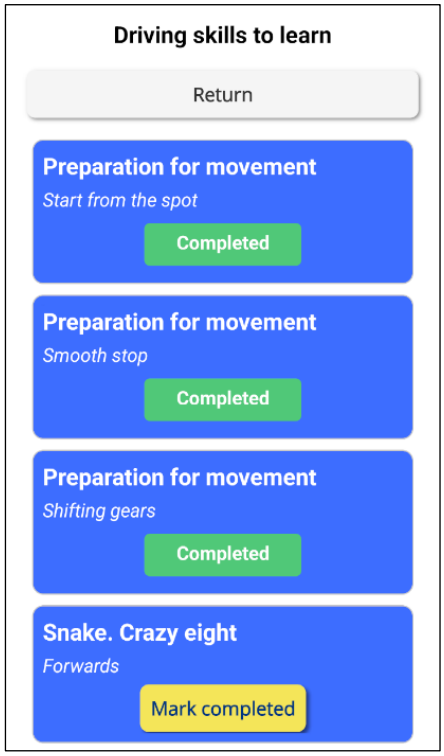


Рисунок 4.9 – Екран спливаючого вікна перегляду опанованих навчальних вправ

Також на сторінці профілю можна відкрити бічне меню застосунку (див. рисунок 4.10) з допомогою якого можна виконувати навігацію по доступним сторінкам: профіль, всі інструктори, мої уроки, мій інструктор, чат з інструктором. Розглянемо кожну сторінку окремо.

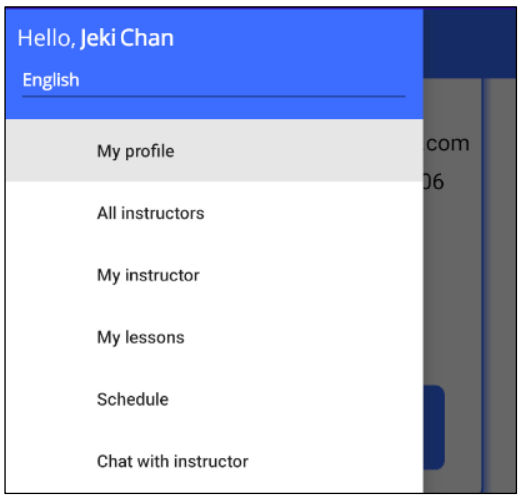


Рисунок 4.10 – Екран з відкритим бічним меню навігації

Оскільки профіль ми тільки що вже описали перейдемо до сторінки всіх інструкторів (див. рисунок 4.11).

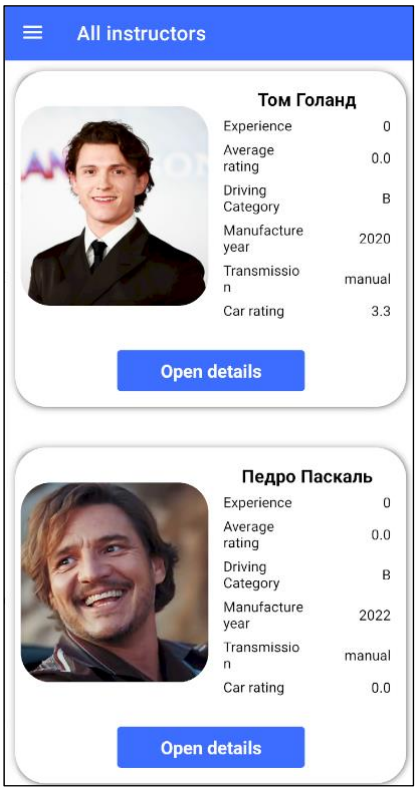


Рисунок 4.11 – Екран сторінки всіх інструкторів

На цьому екрані можна побачити картки інструкторів, які містять фото та головні відомості про самого інструктора та машину на якій буде проходити навчання. Для перегляду детальної інформації про інструктора можна натиснути

відповідну кнопку, після чого користувача буде перенаправлення на відповідну сторінку (див. рисунок 4.12).

При детальному перегляді користувачу доступно більше інформації як про самого інструктора, так і про його машину. Якщо студент ще не обрав собі інструктора, йому буде доступна кнопка для відправлення запиту до адміністрації для закріплення за інструктором. Якщо студент записаний саме до цього інструктора, він зможе перейти до залишення відгуку про інструктора (див. рисунок 4.13), а також оцінити машину (див. рисунок 4.13). Також в будь-якому разі студенту доступні до перегляду відгуки про інструктора (див. рисунок 4.14).

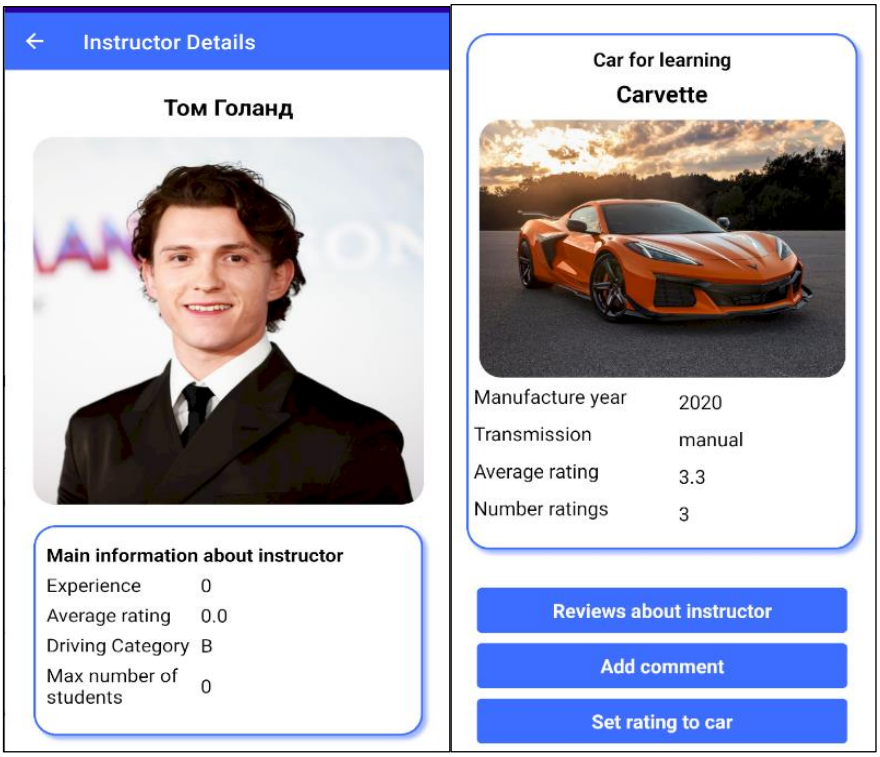


Рисунок 4.12 – Екран сторінки детальної інформації про інструктора

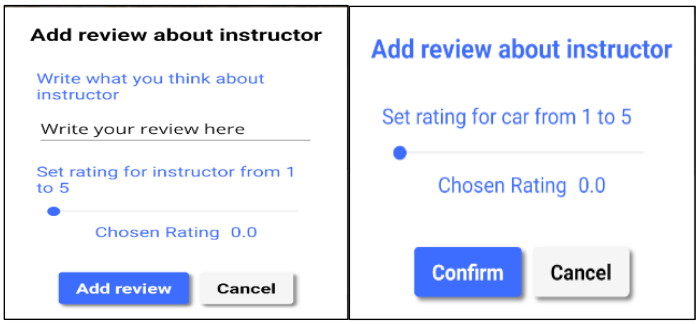


Рисунок 4.13 – Екрани спливаючого вікна для залишення відгуку про інструктора та оцінки машини

Рисунок 4.14 – Екран спливаючого вікна з відгуками про інструктора

Наступною сторінкою куди студент може перейти з бокового меню навігації це сторінка з його заняттями (див. рисунок 4.15)

Рисунок 4.15 – Екран сторінки перегляду моїх занять

Тут користувачі можуть побачити відвідані уроки, а також уроки на які вони записані та які ще мають відбутися. Майбутні уроки є можливість відмінити натиснувши відповідну кнопку, але не пізніше ніж за день до початку заняття.

Далі з бокового меню учень може потрапити до сторінки розкладу його інструктора (див. рисунок 4.16). Тут він бачить розклад уроків інструктора, та може перейти виконати запис на певний урок. Для запису необхідно перевірити дані обраного уроку у вспливаючому вікні (див. рисунок 4.17), та натиснути кнопку для переходу у веб-застосунок для оплати заняття з допомогою LiqPay. Запис буде здійснено лише після оплати студентом заняття.

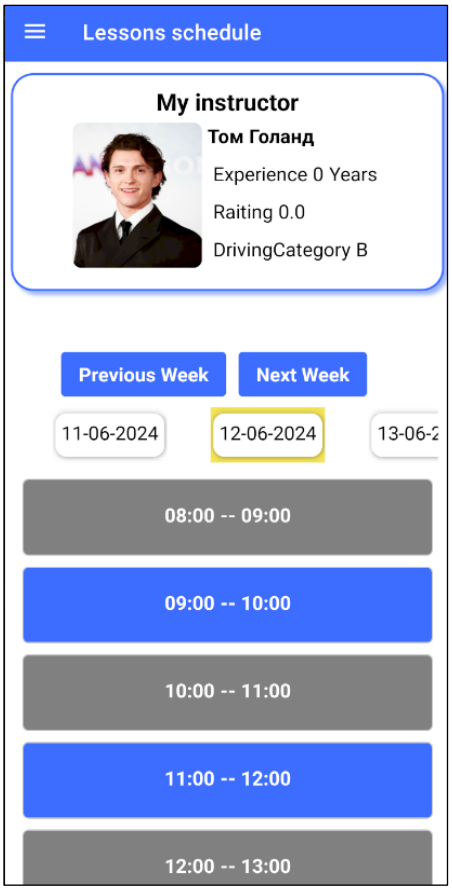


Рисунок 4.16 – Екран сторінки розкладу інструктора

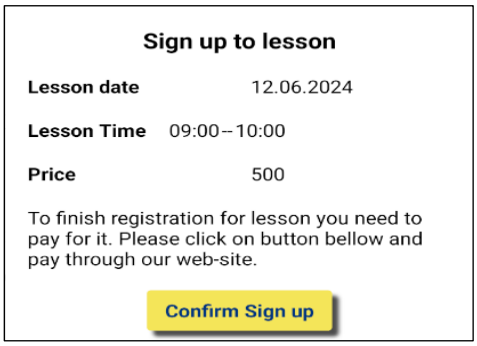


Рисунок 4.17 – Екран зі спливаючим вікном для перевірки даних уроку та переходу до оплати

Останньою доступною можливістю для студента є відкриття чату з інструктором (див. рисунок 4.18). В чаті можна переглядати надіслані та отримані раніше повідомлення, а також надсилати нові повідомлення.

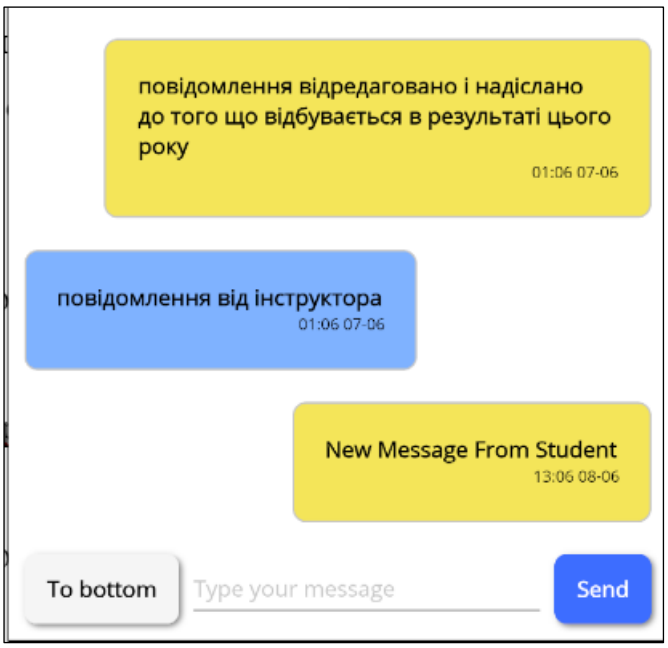


Рисунок 4.18 – Екран сторінки чату між студентом та інструктором

Перейдемо до розгляду функціоналу, що доступний інструкторам автошколи. Після логіну, так само як і студенти інструктор потрапляє до свого профілю, що схожий на профіль студента, але все ж має деякі відмінності (див. рисунок 4.19).

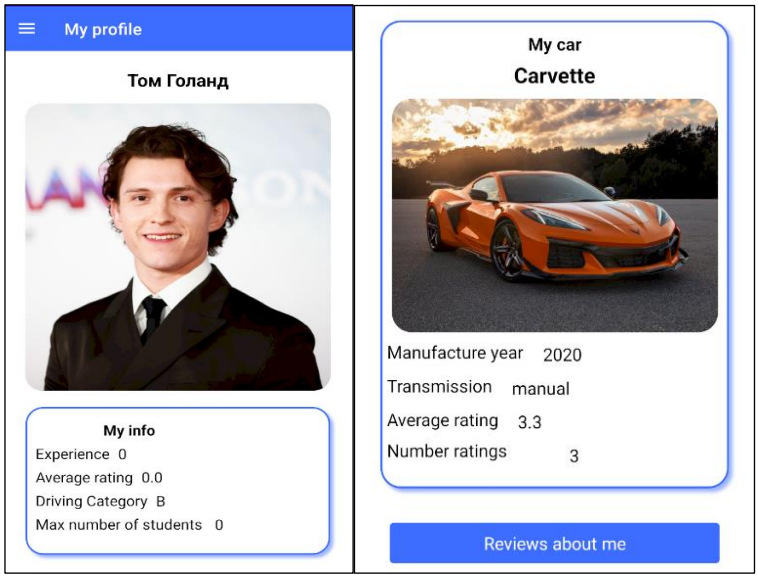


Рисунок 4.19 – Екран сторінки профіля інструктора

Загалом на сторінці свого профілю інструктор може побачити інформацію про себе та свою машину на якій проводиться навчання, а також відкрити спливаюче вікно з відгуками про нього, ідентичне до того, що бачать студенти (див. рисунок 4.13).

Окрім профілю інструктор може переглянути сторінки всіх інструкторів, свій розклад, а також чати з учнями. Розглянемо кожну сторінку детальніше.

На сторінці перегляду всіх інструкторів (див. рисунок 4.10), користувач може переглянути всіх інших працівників, що проводять навчання, окрім нього самого. Про кожного інструктора можна переглянути детальну інформацію (див. рисунок 4.11), та його машину, єдиною відмінністю від подібного функціоналу для студентів є те, що інструктору будуть недоступні опції для відправки запиту на закріплення за інструктором, а також можливість залишати коментарі або оцінювати машину.

Наступною доступною є сторінка для перегляду свого розкладу (див. рисунок 4.20).

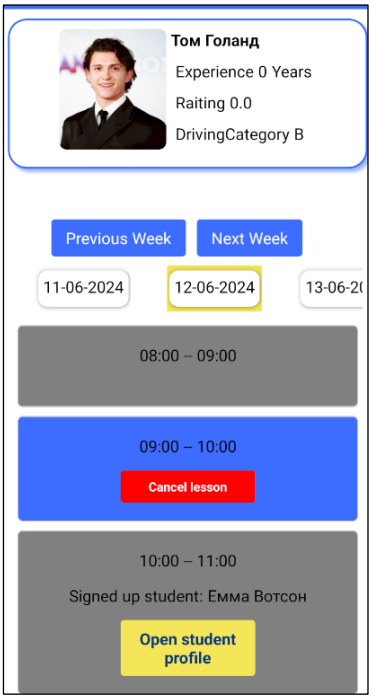


Рисунок 4.20 – Екран сторінки перегляду розкладу з боку інструктора

На даній сторінці інструктор може відмінити майбутній урок, на який ще не записався студент, а також переглянути який студент на коли записаний. У разі

якщо на урок записаний студент, існує можливість перейти до профілю студента, де вказана основна інформація про нього, а також зазначено прогрес по навчанню (див. рисунок 4.21). Вже зі сторінки профіля студента інструктор може відкрити спливаюче вікно з даними про опановані та не опановані студентом навички водіння (див. рисунок 4.22).

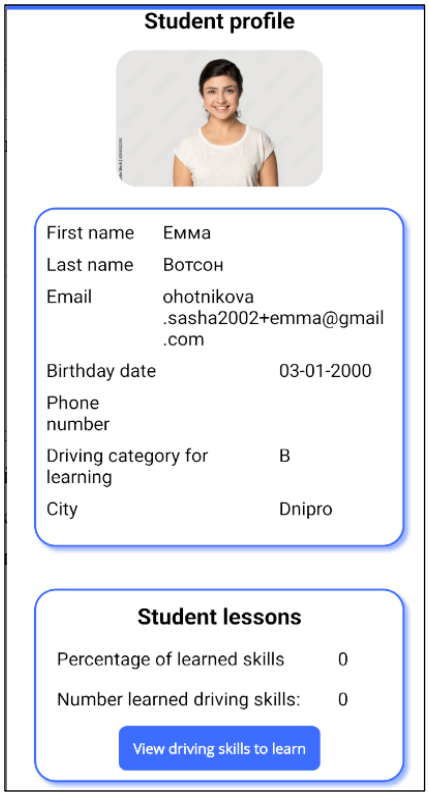


Рисунок 4.21 – Екран сторінки профіля студента

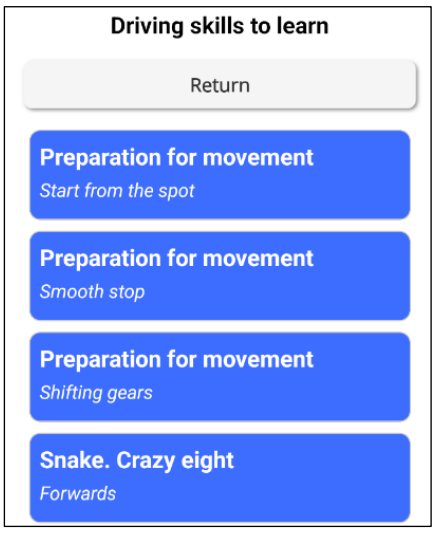


Рисунок 4.22 – Екран зі спливаючим вікном з опановуваними навичками студента

Останньою сторінкою доступною для інструкторів є сторінка чатів зі студентами (див. рисунок 4.23).

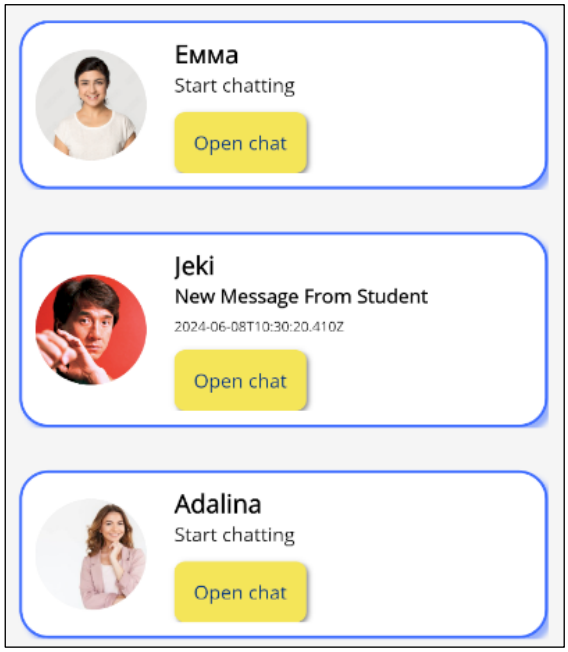


Рисунок 4.23 – Екран сторінки списку чатів з студентами

Тут у вигляді списку користувач може переглянути чати з усіма студентами. А також перейти у певний чат для надсилання або читання повідомлень інтерфейс самого чату зі сторони інструктора ідентичний до відповідного інтерфейсу зі сторони користувача (див. рисунок 4.18).

Отже було розглянуто основні сторінки та функціональні можливості мобільного застосунку для організації практичних занять в автошколі, що спрямовані на задоволення потреб як студентів так і інструкторів автошколи.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Обґрунтування вибору виду тестування

Для забезпечення якості та надійності мобільного додатку, призначеного для організації проведення практичних занять в автошколі, було обрано відповідні методи тестування. Враховуючи особливості даного застосунку, ми вирішили використати кілька видів тестування, кожен з яких має свої переваги та обґрунтування:

- юніт-тестування (Unit Testing): Юніт-тестування дозволяє перевірити окремі модулі коду, такі як методи чи функції, на їх правильність. Це забезпечує: виявлення та виправлення помилок на ранніх етапах розробки; підвищення якості коду за рахунок перевірки окремих компонентів. спрощення рефакторингу, оскільки тести гарантують, що зміни не порушать існуючу функціональність;
- регресійне тестування (Regression Testing): Регресійне тестування необхідне для перевірки того, що нові зміни в коді не порушили існуючу функціональність: забезпечення стабільності додатку після внесення нових функцій або виправлення помилок; виявлення побічних ефектів змін; підтримка безперервної інтеграції та доставки (CI/CD) шляхом автоматизації тестів;
- тестування користувацького інтерфейсу (UI Testing): Widget-тестування UI дозволяє перевірити взаємодію користувача з додатком, його зручність та відповідність дизайну. Воно гарантує коректне відображення елементів інтерфейсу на різних пристроях та в різних умовах, перевіряє логіку навігації та доступність функцій для користувачів. Widget-тестування забезпечує високу якість користувацького досвіду, дозволяючи ефективно виявляти та виправляти потенційні проблеми в інтерфейсі програми.

Вибір цих методів тестування обґрунтований необхідністю забезпечення високої якості, надійності та безпеки мобільного додатку для автошкіл.

Комплексне застосування різних видів тестування дозволить виявити та виправити помилки на різних етапах розробки, забезпечуючи стабільну та ефективну роботу додатку у виробничому середовищі.

5.2 Опис тестування

Перш за все для перевірки правильності роботи окремих модулів (функцій) було створено юніт тести. Для їх написання ми використали популярний засіб MSTest [10], що представляє собою фреймворк для написання тестів. MSTest забезпечує простий і ефективний спосіб створення і виконання тестів, дозволяючи нам легко перевіряти функціональність окремих методів та класів. Використання цього інструменту допомогло виявити та виправити помилки на ранніх етапах розробки, забезпечуючи високу якість коду та стабільність роботи додатку.

Наведемо нижче програмний код класу з юніт тестами, що призначені для перевірки функції для відправки Get запитів до серверу.

```
[TestClass]
public class GetRequestIntegrationTest
{
    private IHttpClientService _httpClientService;
    private IGetRequest _getRequest;

    [TestInitialize]
    public void SetUp()
    {
        Environment.SetEnvironmentVariable("BASE_URL",
"http://localhost:3000/api/");
        _httpClientService = new HttpClientService();
        _getRequest = new GetRequest(_httpClientService);
    }

    [TestMethod]
    public async Task ExecuteAsync_ShouldReturnSuccessResponse()
    {
        // Act
        var result = await
_getRequest.ExecuteAsync<GetAllInstructorsResponse>(RoutesConstants.GetAllI
nstructors);

        var expectedStatus = ResponseStatuses.Sucess.ToLower();

        // Assert
        Assert.IsNotNull(result);

        Assert.AreEqual(expectedStatus, result.Status?.ToLower());
```

```
        }

        [TestMethod]
        public async Task ExecuteAsync_ShouldHandleInvalidUrl()
        {
            var invalidUrl = "Invalid/Url";
            try
            {
                // Act
                await _getRequest.ExecuteAsync<object>(invalidUrl);
                Assert.Fail("Expected exception was not thrown.");
            }
            catch (HttpRequestException ex)
            {
                // Assert
                Assert.IsTrue(ex.Message.Contains("404")              ||
ex.Message.Contains("NotFound"));
            }
        }

        [TestMethod]
        public async Task ExecuteAsync_ShouldHandleInvalidReturnType()
        {
            try
            {
                // Act
                await
_getRequest.ExecuteAsync<int>(RoutesConstants.GetAllInstructors);
                Assert.Fail("Expected exception was not thrown.");
            }
            catch(JsonReaderException ex)
            {
                // Assert
                Assert.IsTrue(true);
            }
            catch (Exception ex)
            {
                Assert.Fail($"Unexpected    exception    type    thrown:
{ex.GetType()}");
            }
        }
    }
```

У наведеному фрагменті коду визначено три тести, що перевіряють роботу функції для відправки запитів на бекенд сервер у різних сценаріях. У першому сценарії ми відправляємо запит з усіма правильними параметрами та очікуємо успішну відповідь, другий сценарій перевіряє спробу відправлення запиту, коли було вказано неправильне посилання, у такому випадку очікуємо, що буде викинуто виключення, третій тест перевіряє успішну відправку запитів, але коли вказано невірний тип даних для десереалізації відповіді від серверу, для

успішного проходження тесту ми очікуємо що буде викинуто помилку певного типу, а саме JsonReaderException.

Наступним видом тестування було проведено widget-тестування, що дозволяє перевіряти окремі компоненти інтерфейсу користувача в ізоляції [8]. Цей метод тестування допомагає виявити помилки на ранніх етапах розробки, забезпечуючи правильну роботу та взаємодію окремих елементів інтерфейсу, таких як кнопки, поля вводу та інші віджети. Widget-тестування також сприяє підвищенню якості коду, зменшуючи кількість дефектів та полегшуючи подальшу підтримку і розвиток додатку. Код декількох widget-тестів наведемо нижче:

```
public class LoginTest
{
    private AndroidDriver<AndroidElement> _driver;

    [SetUp]
    public void Setup()
    {
        var driverOptions = new AppiumOptions();

driverOptions.AddAdditionalCapability(MobileCapabilityType.PlatformName,
"Android");

driverOptions.AddAdditionalCapability("appium:automationName",
"UiAutomator2");

driverOptions.AddAdditionalCapability(MobileCapabilityType.App,
"com.Auto.School.Mobile");

driverOptions.AddAdditionalCapability(MobileCapabilityType.DeviceName,
"Pixel 7 Pro - API 34");

        _driver = new AndroidDriver<AndroidElement>(new
Uri("http://192.168.0.102:4723/"), driverOptions);
        _driver.ActivateApp("com.Auto.School.Mobile");
    }

    [TearDown]
    public void TearDown()
    {
        _driver.Quit();
        _driver.Dispose();
    }

    [Test]
    public void InputCorrectLoginAndPasswordAndClickLoginButton()
    {
        var usernameEntry =
_driver.FindElementByAccessibilityId("LoginPage_UsernameEntry");
        usernameEntry.Clear();
```

```
            usernameEntry.SendKeys("dgekichan2024@gmail.com");

            var                         passwordEntry                       =
_driver.FindElementByAccessibilityId("LoginPage_PasswordEntry");
            passwordEntry.Clear();
            passwordEntry.SendKeys("Password01;");

            var                          loginButton                        =
_driver.FindElementByAccessibilityId("LoginPage_LoginButton");
            loginButton.Click();


            var                          errorLabel                         =
_driver.FindElementByAccessibilityId("LoginPage_ErrorAlert");
            Assert.IsFalse(errorLabel.Displayed);
        }

        [Test]
        public void InputWrongLoginAndPasswordAndClickLoginButton()
        {
            var                         usernameEntry                       =
_driver.FindElementByAccessibilityId("LoginPage_UsernameEntry");
            usernameEntry.Clear();
            usernameEntry.SendKeys("dgekichan@gmail.com");

            var                         passwordEntry                       =
_driver.FindElementByAccessibilityId("LoginPage_PasswordEntry");
            passwordEntry.Clear();
            passwordEntry.SendKeys("Password");

            var                          loginButton                        =
_driver.FindElementByAccessibilityId("LoginPage_LoginButton");
            loginButton.Click();

            var                          errorAlert                         =
_driver.FindElementByAccessibilityId("LoginPage_ErrorAlert");
            Assert.IsTrue(errorAlert.Displayed);
        }
    }
```

У наведеному фрагменті коду можна побачити два widget-тести, що перевіряють роботу кнопки для логіну на сторінці логіну. Перший тест перевіряє виконання входу в застосунок, коли пошта і пароль введено вірно, тоді як другий перевіряє показ повідомлення з помилкою після невдалого логіну з невірною поштою або паролем.

Останній вид тестування, який ми використали під час написання мобільного додатку було регресійне тестування. Це тестування дозволяє переконатися, що нещодавно внесені зміни або додані нові функціональності не порушили вже існуючу функціональність додатку. Регресійне тестування включає

перевірку всіх основних і критичних функцій додатку, а також тестування виправлених дефектів для забезпечення їх остаточного усунення. Це допомагає підтримувати стабільність і якість програмного забезпечення, мінімізуючи ризик виникнення нових помилок після внесення змін у код.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було спроектовано та розроблено мобільну частину програмної системи для організації проведення практичних занять в автошколі. У ході розробки ми провели аналіз предметної галузі, сформували вимоги до програмної системи, описали архітектуру та як саме було спроектовано систему, після чого було визначено програмні рішення, що були використані, а також було виконано тестування створюваної програмної системи. В результаті проведеного аналізу ринку та предметної галузі ми визначили основних конкурентів, головний функціонал, що має бути присутній в системі, а також встановлено цільову аудиторію додатку.

Основним функціоналом мобільного додатку є керування розкладом, тобто запис на заняття, з можливістю оплати прямо в середині застосунку, та їх відміна, а також відстеження власного прогресу з можливість відмічати опановані вправи у навчальній програмі, що необхідні для складання практичного іспиту з водіння іспиту.

Для розробки мобільного застосунку було обрано фреймворк .NET MAUI. Робота застосунку можлива на обох мобільних платформах – Android та IOS. Для зберігання певних даних про користувача локально на пристрої, було використано механізм, що надається .NET MAUI, під назвою Preferences, він дозволяє зберігати ключ-значення та отримати доступ до даних навіть коли пристрій не матиме доступ до інтернету.

Загалом створений мобільний застосунок, що є частиною програмної системи для організації проведення практичних занять в автошколі задовольняє потреби як інструкторів, так і учнів автошколи, представляючи собою зручний та ефективний інструмент для планування, ведення та відстеження прогресу практичних занять. Таким чином, розроблений мобільний застосунок має потенціал стати необхідним інструментом для автошколи, спростивши та оптимізувавши процес проведення занять для всіх учасників навчального процесу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Про затвердження Положення про порядок видачі посвідчень водія та допуску громадян до керування транспортними засобами. Офіційний вебпортал парламенту України. URL: https://zakon.rada.gov.ua/laws/show/340-93-п#Text (дата звернення: 11.05.2024).

2. В Україні збільшилась кількість виданих посвідчень водія. З якого віку можна отримати права: Авто новини від AUTO-Consulting - прав. Все про автобізнес: ринок автомобілів, автобусів, вантажівок. Статистика продажів. Продаж авто. AUTO-Consulting. URL: https://autoconsulting.ua/article.php?sid=56178 (дата звернення: 11.05.2024).

3. What is .NET MAUI? - .NET MAUI. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui?view=net-maui-8.0 (дата звернення: 11.05.2024).

4. HttpClient Class (System.Net.Http). Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/dotnet/api/system.net.http.httpclient?view=net-8.0 (дата звернення: 11.05.2024).

5. Preferences - .NET MAUI. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/ru-ru/dotnet/maui/platform-integration/storage/preferences?view=net-maui-8.0&tabs=windows (дата звернення: 01.06.2024).

6. Client-Server Model - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/client-server-model/ (дата звернення: 11.05.2024).

7. Rational Software Architect 9.6.1. IBM in Deutschland, Österreich und der Schweiz. URL: https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-use-case (дата звернення: 11.05.2024).

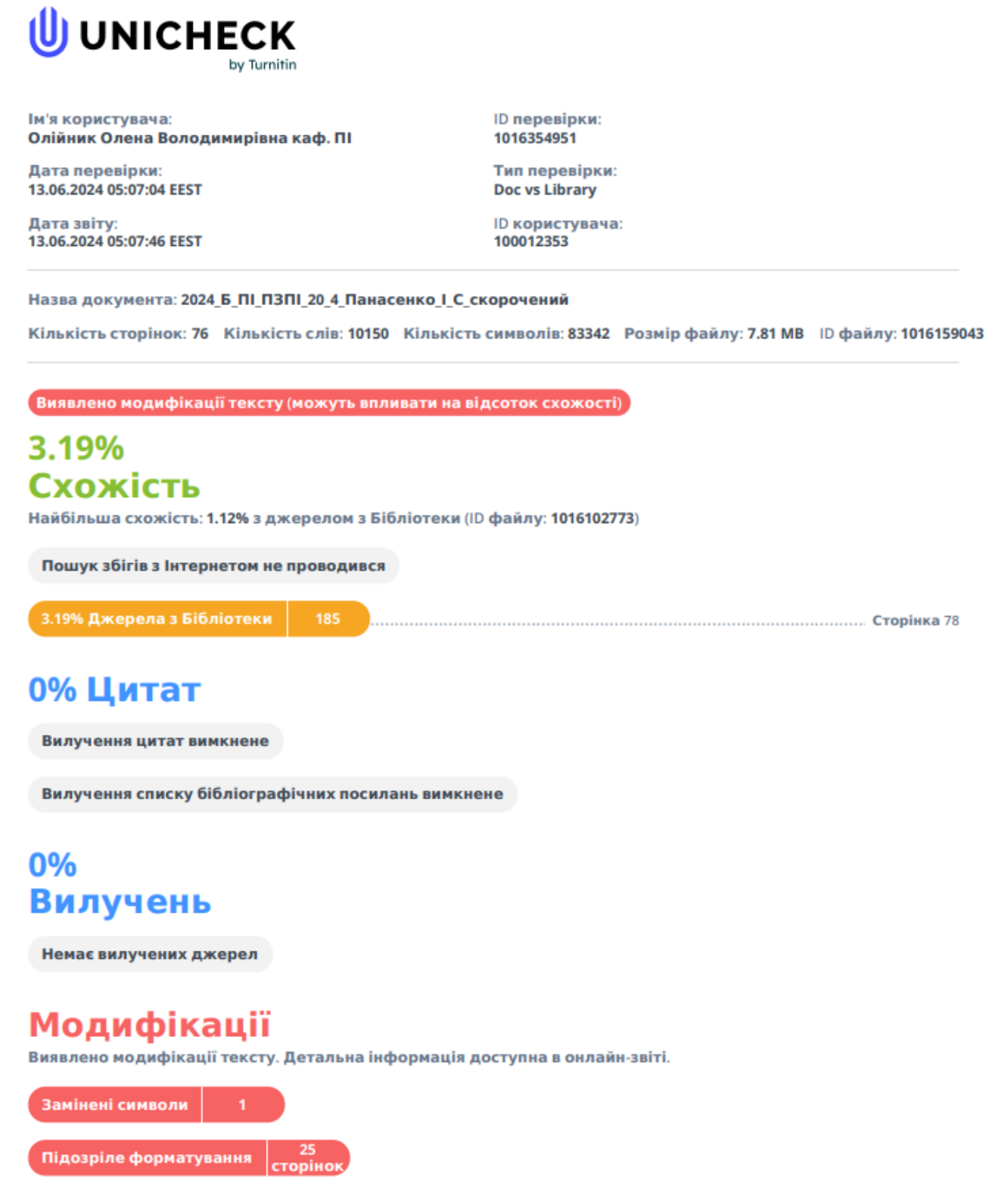
8. NET MAUI - UI testing with Appium and NUnit - Code Samples. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/samples/dotnet/maui-samples/uitest-appium-nunit/ (дата звернення: 02.06.2024).

9. Dependency injection - .NET. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/dotnet/core/extensions/dependency-injection (дата звернення: 01.06.2024).

10. Unit testing C# with MSTest and .NET - .NET. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/dotnet/core/testing/unit-testing-with-mstest (дата звернення: 02.06.2024).

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



UNICHECK
by Turnitin

Ім'я користувача:
Олійник Олена Володимирівна каф. ПІ

ID перевірки:
1016354951

Дата перевірки:
13.06.2024 05:07:04 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
13.06.2024 05:07:46 EEST

ID користувача:
100012353

Назва документа: 2024_Б_ПІ_ПЗПІ_20_4_Панасенко_І_С_скорочений

Кількість сторінок: 76 Кількість слів: 10150 Кількість символів: 83342 Розмір файлу: 7.81 MB ID файлу: 1016159043

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

3.19%
Схожість

Найбільша схожість: 1.12% з джерелом з Бібліотеки (ID файлу: 1016102773)

Пошук збігів з Інтернетом не проводився

3.19% Джерела з Бібліотеки 185 ..... Сторінка 78

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

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

Замінені символи 1

Підозріле форматування 25 сторінок

ДОДАТОК Б

Слайди презентації

Харківський національний університет радіоелектроніки
Кваліфікаційна робота бакалавра

1

Програмна система для організації практичних занять в автошколі. Mobile

Виконав ст. гр. ПЗПІ 20-4 Панасенко Ігор Сергійович
Керівник ст. викл. Олійник Олена Володимирівна

2

Мета роботи:

ПЕРЕФРАЗУВАТИ

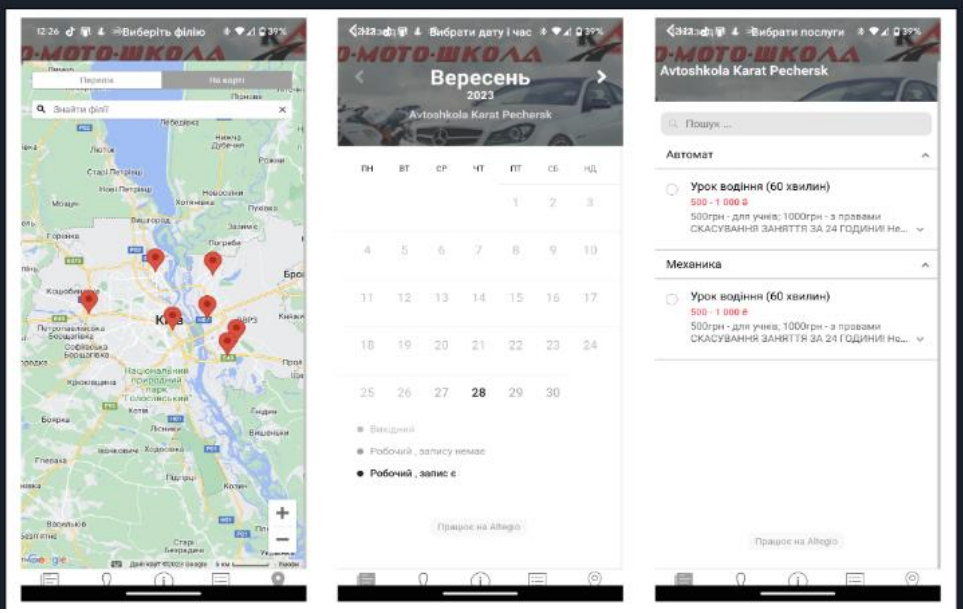
Створення зручного та ефективного інструменту на мобільній платформі для планування, ведення та аналізу практичних занять в автошколі, який задовольняє потреби як інструкторів, так і студенітв.

3

Аналіз ринку та актуальність

4

Автошкола “Карат”



5

Автошкола “Карат”

Переваги застосунку:

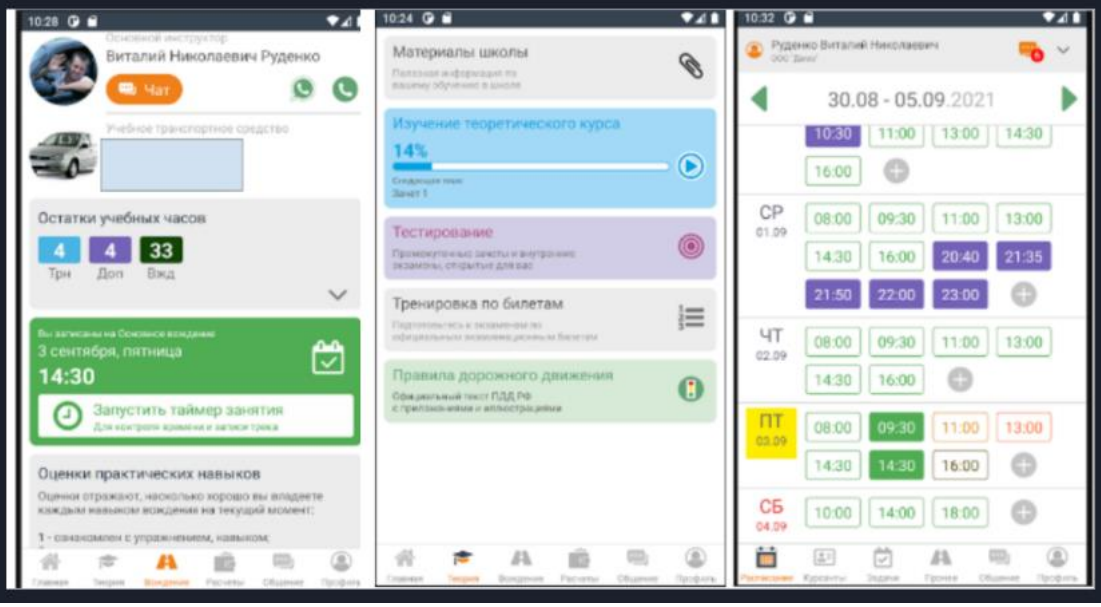
- запис на заняття;
- повідомлення про наближення часу заняття;
- попередження про скасування заняття за 24 години до його початку, у разі несплати;
- список майбутніх занять та історія проведених занять.

Недоліки застосунку:

- неправильне відображення нижнього меню, що значно ускладнює навігацію;
- немає відгуків про конкретних інструкторів та машини;
- застосунок розроблений тільки для студентів і ніяк не допомагає інструкторам з водіння.

6

Автошкола-контроль



7

Автошкола-контроль

Переваги додатку:

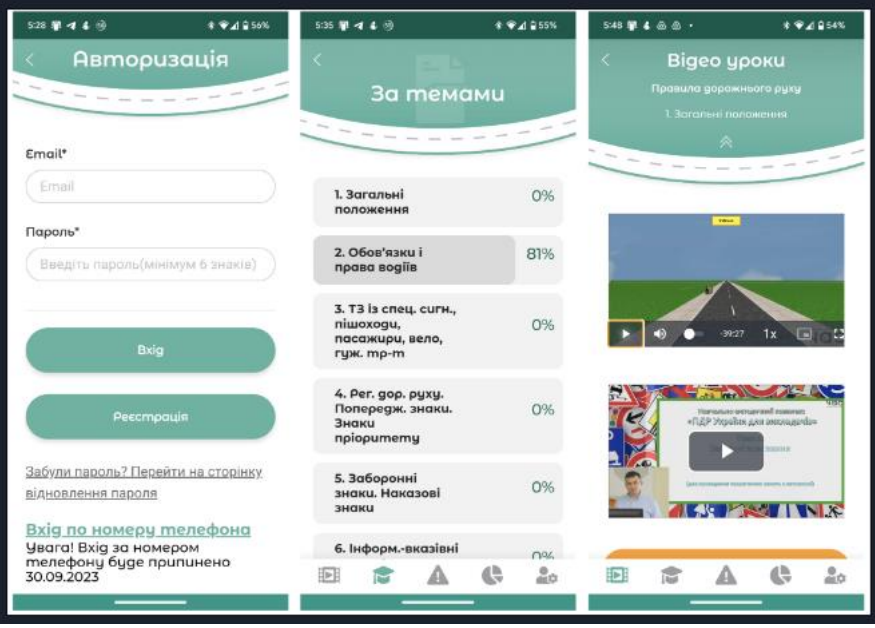
- можливість переглянути розклад занять у виді календаря;
- можливість записатися на заняття;
- можливість переглядати кількість навчальних годин за кермом;
- можливість зв'язатися з інструктором або автошколою через чат;
- наявність функціоналу для інструкторів.

Недоліки додатку:

- відсутня оплата за заняття через застосунок;
- відсутність коментарів та оцінок, від інших студентів;.
- відсутній список інструкторів;

8

Автошкола "Час"



9

Онлайн автошкола “Час”

Переваги додатку:

- приємний, ненав’язливий дизайн;
- можливість безкоштовно ознайомитися з майже з усім функціоналом додатку.
- присутність локалізації;
- наявність сторінки зі статистикою пройденого матеріалу.

Недоліки додатку:

- відображення пароля у явному вигляді на сторінці профіля;
- зависання додатку;
- відсутність коментарів та оцінок інших користувачів.

10

Висновки аналізу ринку

Відповідно до результатів проведеного аналізу ринку можна зробити висновок, що кожен застосунок дотримується свого власного дизайну на кожній сторінці. Важливим є забезпечення доступності всіх основних елементів керування та навігації у застосунку. Система повинна мати інструменти для відстеження прогресу, а також надавати аналітику щодо виконання вправ, мати можливість для комунікації між студентом та інструктором.

Загалом проаналізовані системи-аналоги, які були описані не покривають весь необхідний для користувача функціонал.

11

Постановка задачі

Основною метою цієї роботи є створення мобільного додатку, що зможе працювати на обох операційних системах Android та IOS. Система що розробляється повинна покращити досвід навчання практичним навичкам водіння в автошколах.

Під час розробки ми поставили такі задачі:

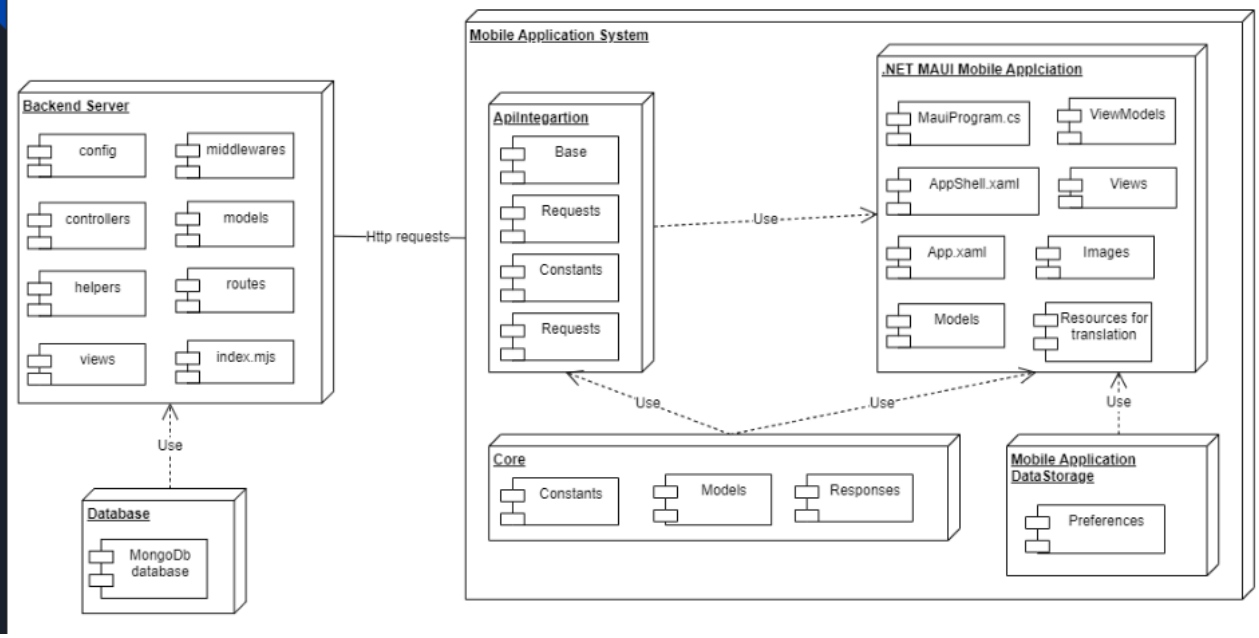
- розробка мінімалістичного зрозумілого інтерфейсу, з привабливою колірною схемою.
- якісна реалізація функціоналу,
- покриття всіх потреб користувачів,
- забезпечення зручності використання, шляхом впровадження пошуку, фільтрації та сортування на сторінках де це може бути необхідним.

12

Архітектура та проектування

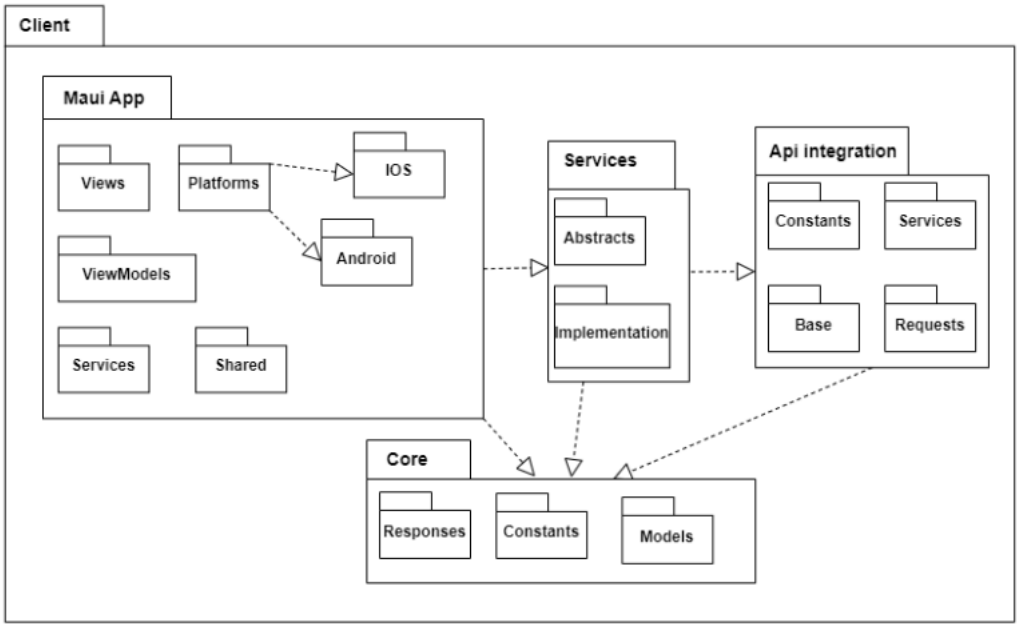
13

Діаграма розгортання



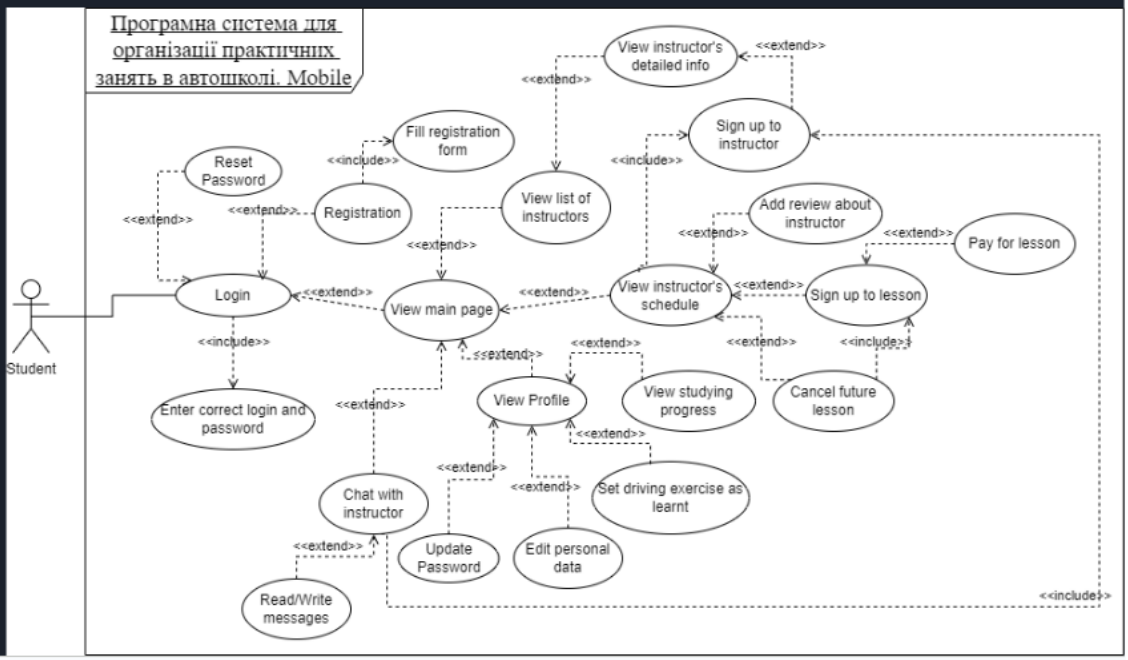
14

Діаграма пакетів



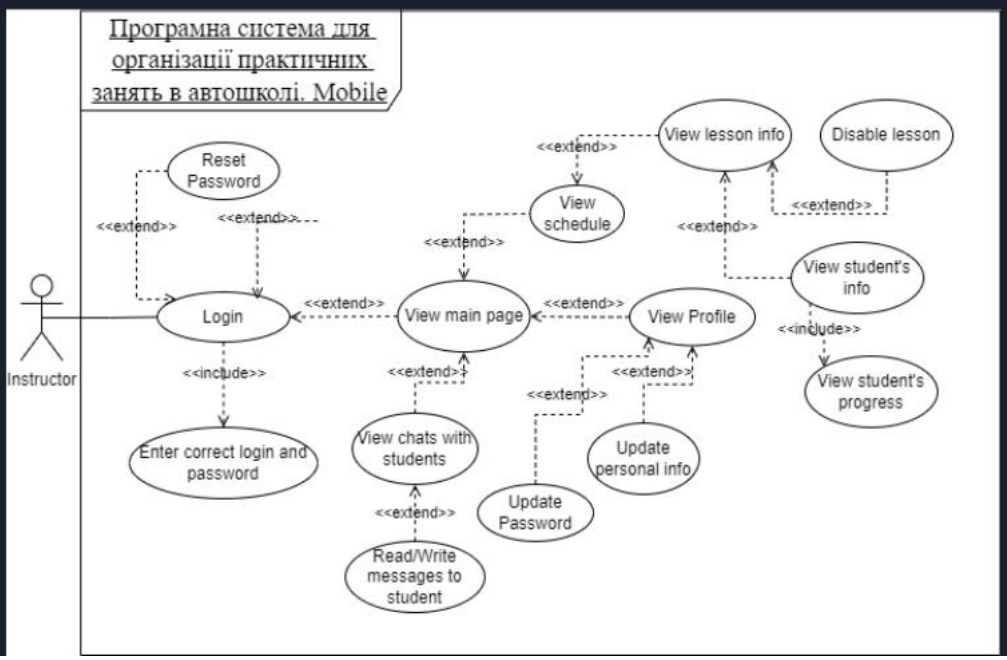
Use-case діаграма. Користувач – студент

15



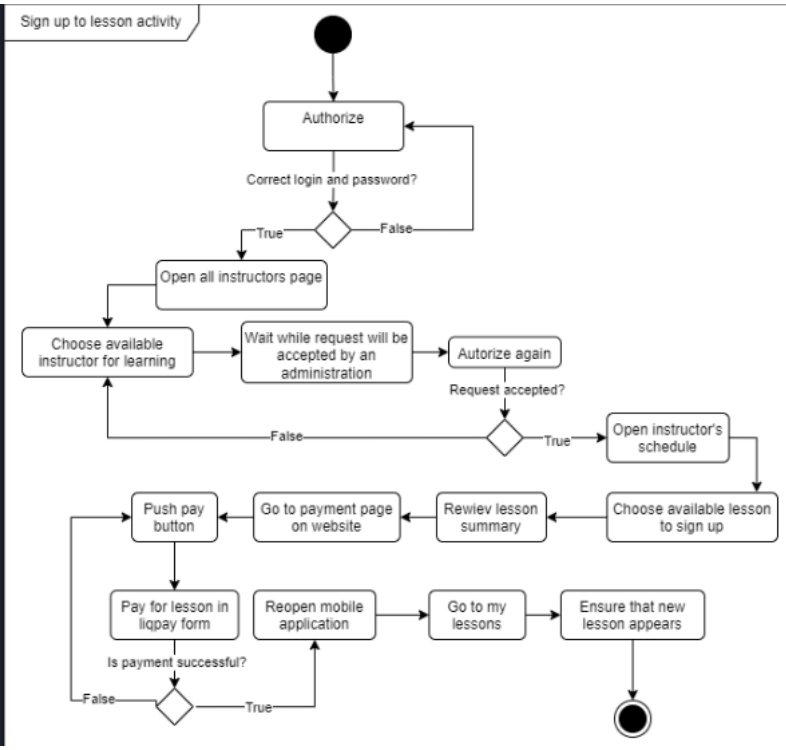
Use-case діаграма. Користувач – інструктор

16



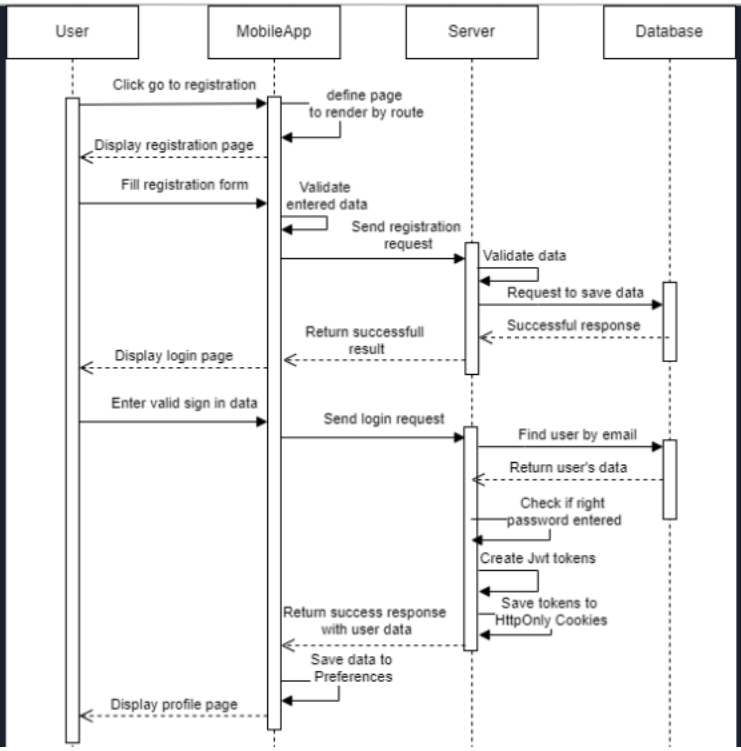
17

Діаграма активності для процесу запису на урок та його оплати



18

Діаграма послідовності для процесу реєстрації та подальшої авторизації користувача



19

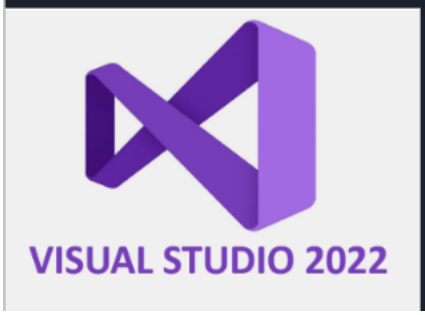
Опис прийнятих програмних рішень

Вибір засобів програмної реалізації

Для розробки мобільного застосунку було обрано наступні технології:

- Мова програмування C#.
- Фреймворк .NET MAUI
- Зберігання даних у Preferences.

Середовище розробки: Visual Studio 2022.



21

Архітектура

- N-layer (багатошарова) архітектура розділяє додаток на кілька логічних шарів, таких як презентація, бізнес-логіка, дані та інфраструктура. Це дозволяє забезпечити масштабованість, модульність та легкість обслуговування додатка. Кожен шар відповідає за свою область і взаємодіє з іншими через чітко визначені інтерфейси.
- MVVM (Model-View-ViewModel) — це шаблон проектування, який сприяє розділенню логіки інтерфейсу користувача (View) від бізнес-логіки та даних (Model) через використання ViewModel. Це полегшує тестування, забезпечує двосторонню прив'язку даних та підтримує чистий і зрозумілий код.

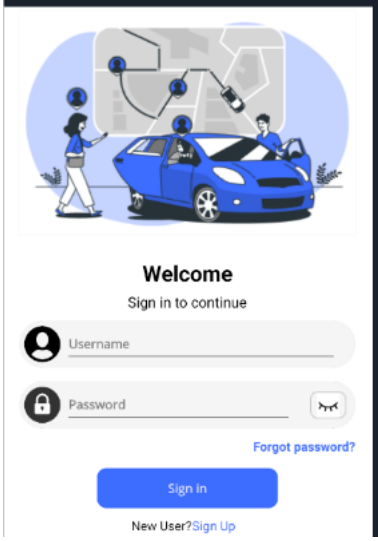
22

Результати розробки мобільного застосунку

Інтерфейс мобільного додатку для студентів з прикладами коду

23

Частина програмного коду функції для логіну



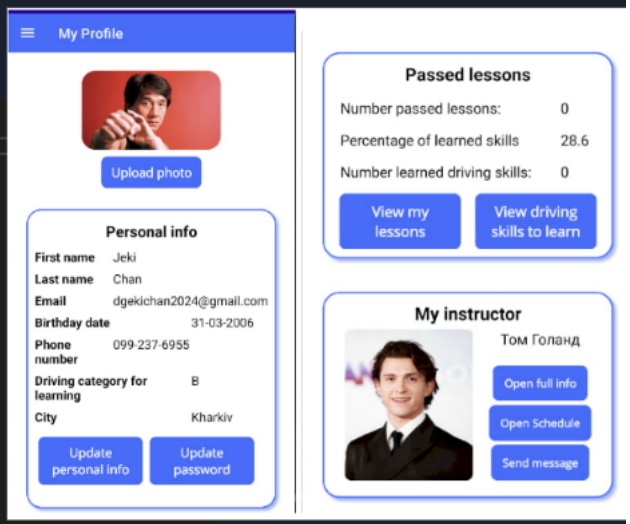

```
[RelayCommand]
2 references
public async Task Login()
{
    if (!ValidateInput())
    {
        return;
    }

    var loginModel = new LoginModel { Email = UserName, Password = Password };
    var response = await _authenticationService.LoginAsync(loginModel);

    if (response == null)
    {
        IsError = true;
        ErrorMessage = AppErrorMessagesConstants.SomethingWentWrongErrorMessage;
        return;
    }

    if (string.Compare(response.Status, "Fail", true) == 0)
    {
        IsError = true;
        ErrorMessage = response.Message ?? AppErrorMessagesConstants.SomethingWentWrongErrorMessage;
        return;
    }

    if (string.Compare(response.Status, ResponseStatuses.Sucess, true) == 0)
    {
        IsError = false;
        if (Preferences.ContainsKey("UserInfo"))
        {
            Preferences.Remove("UserInfo");
        }

        Preferences.Set("UserInfo", JsonConvert.SerializeObject(response.LoginResponseData));
```

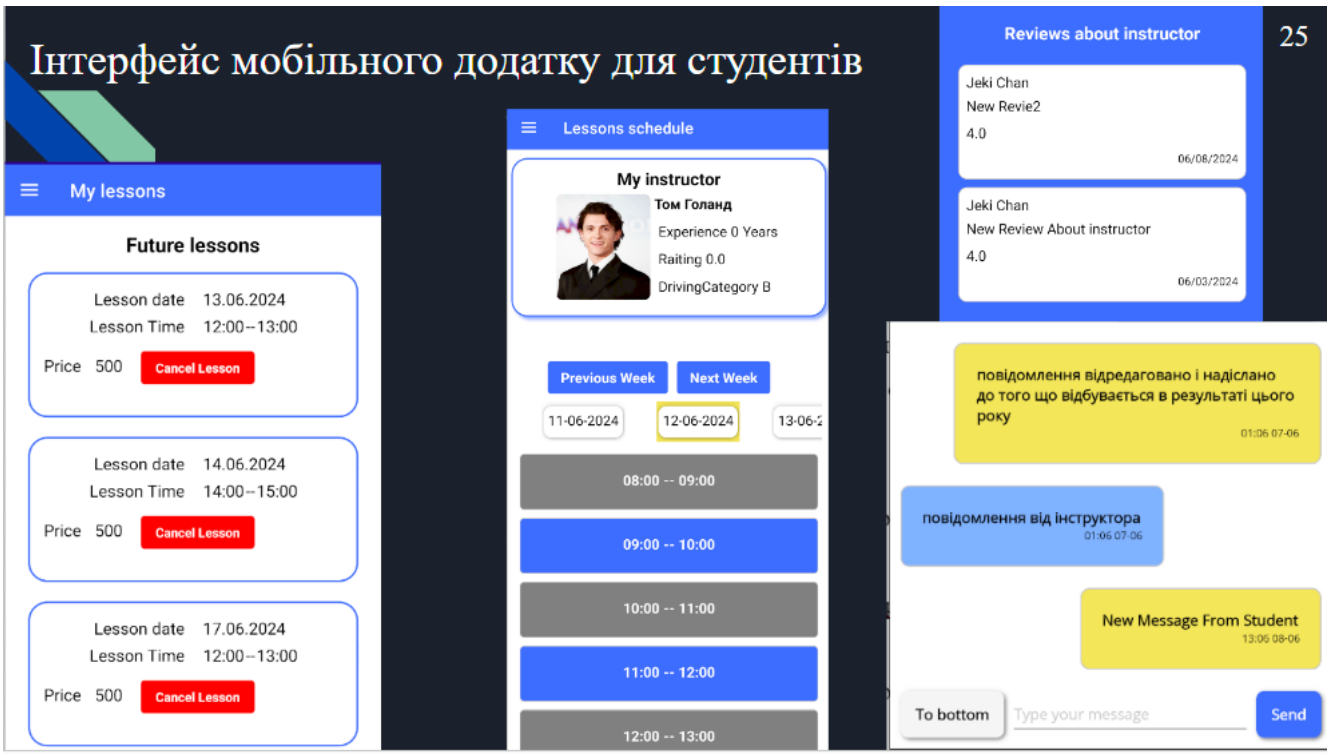
Інтерфейс мобільного додатку для студентів з прикладами коду

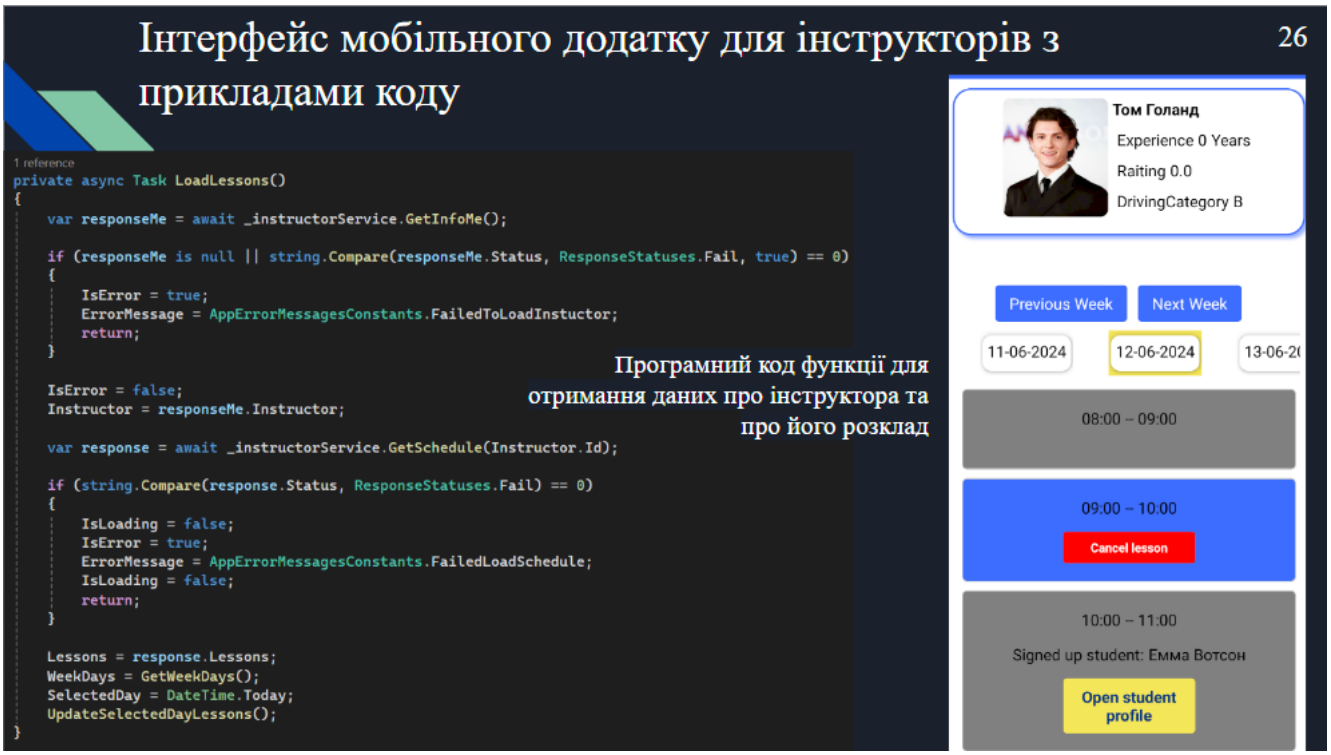
24

Частина програмного коду дженерік функції для виконання гет запитів

```
12 references
public async Task<TResponse> ExecuteAsync<TResponse>(
    string url)
{
    var path = RoutesConstants.BaseUrl + url;

    try
    {
        var response = await _httpClientService.Client.GetAsync(path);

        if (response.IsSuccessStatusCode)
        {
            var responseContent = await response.Content.ReadAsStringAsync();
            var responseData = JsonConvert.DeserializeObject<TResponse>(responseContent);

            if (responseData == null)
            {
                throw new Exception("Cannot Deserialize object");
            }

            return responseData;
        }
        else
        {
            var responseContent = await response.Content.ReadAsStringAsync();
            var responseData = JsonConvert.DeserializeObject<TResponse>(responseContent);
            throw new HttpRequestException($"Cannot send get request to: {path},Status code: {response.StatusCode}");
        }
    }
    catch (Exception ex)
```



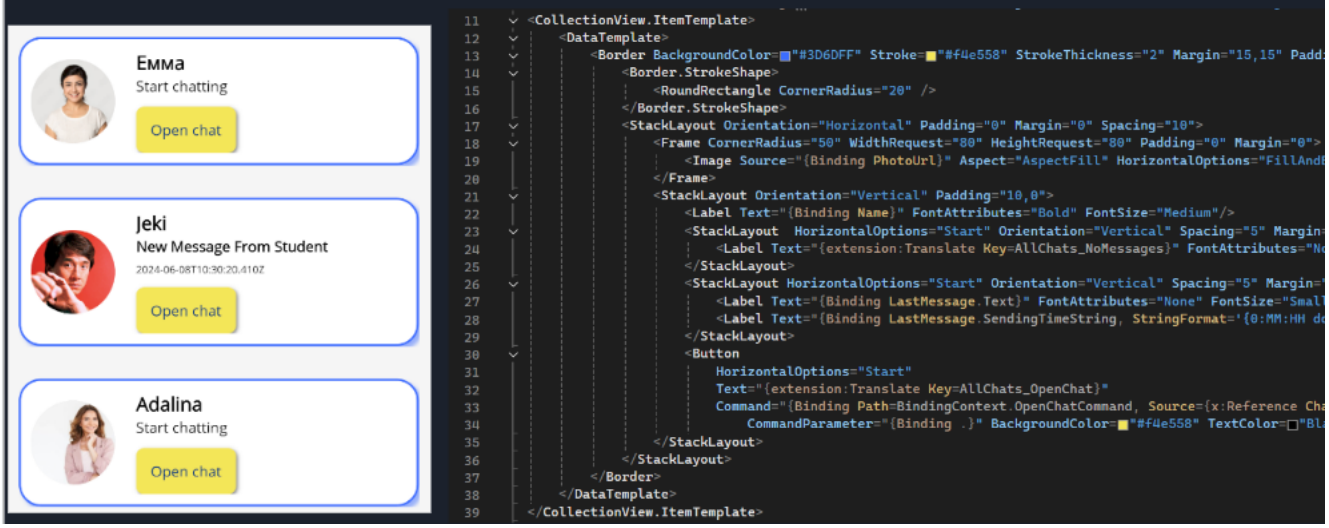

Інтерфейс мобільного додатку для студентів
25
My lessons
Future lessons
Lesson date 13.06.2024
Lesson Time 12:00--13:00
Price 500
Cancel Lesson
Lesson date 14.06.2024
Lesson Time 14:00--15:00
Price 500
Cancel Lesson
Lesson date 17.06.2024
Lesson Time 12:00--13:00
Price 500
Cancel Lesson
Lessons schedule
My instructor
Том Голанд
Experience 0 Years
Raiting 0.0
DrivingCategory B
Previous Week
Next Week
11-06-2024
12-06-2024
08:00 -- 09:00
09:00 -- 10:00
10:00 -- 11:00
11:00 -- 12:00
12:00 -- 13:00
Reviews about instructor
Jeki Chan
New Revie2
4.0
06/08/2024
Jeki Chan
New Review About instructor
4.0
06/03/2024
повідомлення відредаговано і надіслано до того що відбувається в результаті цього року
01:06 07-06
повідомлення від інструктора
01:06 07-06
New Message From Student
13:06 08-06
To bottom
Type your message
Send


Інтерфейс мобільного додатку для інструкторів з прикладами коду
26
1 reference
private async Task LoadLessons()
{
var responseMe = await _instructorService.GetInfoMe();
if (responseMe is null || string.Compare(responseMe.Status, ResponseStatuses.Fail, true) == 0)
{
IsError = true;
ErrorMessage = AppErrorMessagesConstants.FailedToLoadInstuctor;
return;
}
IsError = false;
Instructor = responseMe.Instructor;
var response = await _instructorService.GetSchedule(Instructor.Id);
if (string.Compare(response.Status, ResponseStatuses.Fail) == 0)
{
IsLoading = false;
IsError = true;
ErrorMessage = AppErrorMessagesConstants.FailedLoadSchedule;
IsLoading = false;
return;
}
Lessons = response.Lessons;
WeekDays = GetWeekDays();
SelectedDay = DateTime.Today;
UpdateSelectedDayLessons();
}
Програмний код функції для отримання даних про інструктора та про його розклад
Том Голанд
Experience 0 Years
Raiting 0.0
DrivingCategory B
Previous Week
Next Week
11-06-2024
12-06-2024
08:00 -- 09:00
09:00 -- 10:00
Cancel lesson
10:00 -- 11:00
Signed up student: Емма Вотсон
Open student profile

Інтерфейс мобільного додатку для інструкторів з прикладами коду

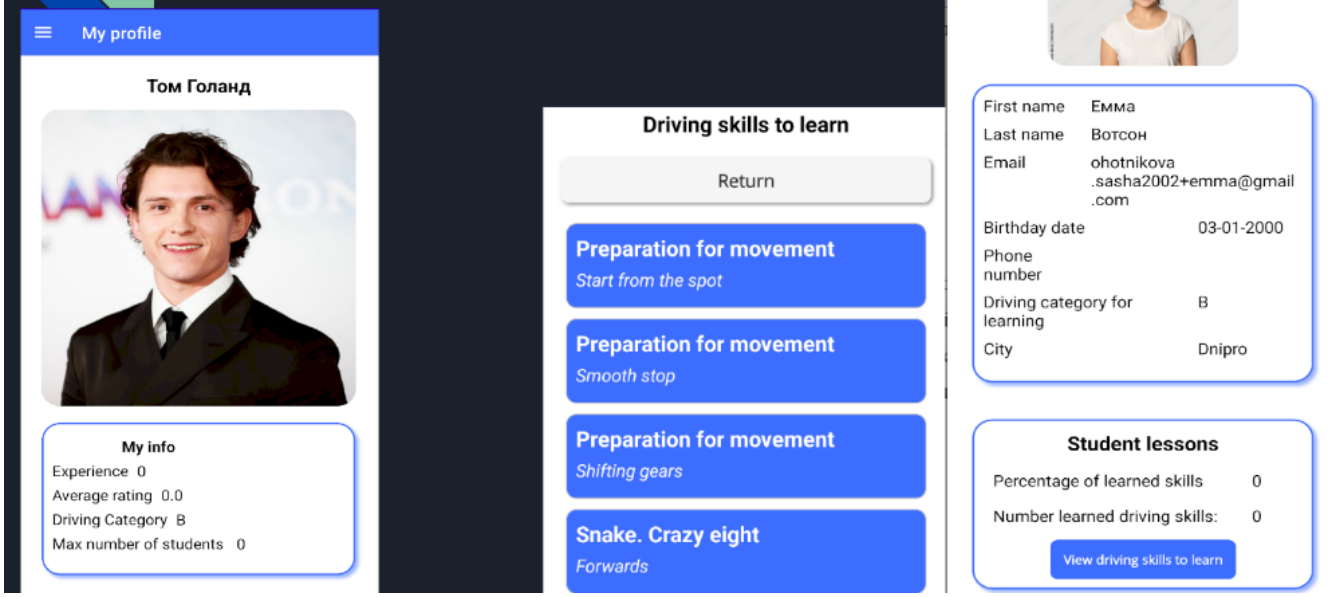
27

Частина XML коду сторінки відображення всіх чатів для інструктора



Інтерфейс мобільного додатку для інструкторів

28



29

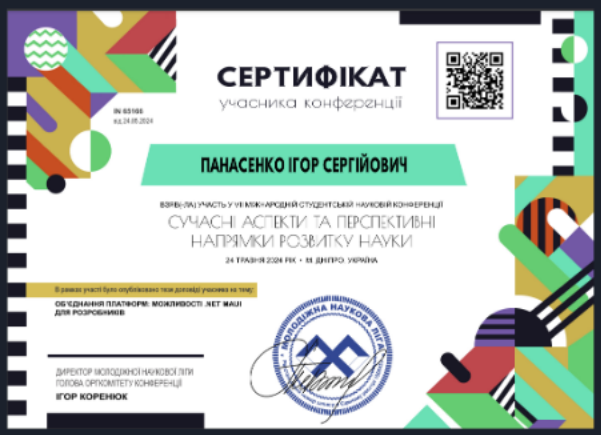
Тестування програмного забезпечення

В процесі розробки мобільного застосунку були використані наступні види тестування:

- Юніт-тестування (Unit Testing): Юніт-тестування дозволяє перевірити окремі модулі коду, такі як методи чи функції, на їх правильність.
- Регресійне тестування (Regression Testing): Регресійне тестування необхідне для перевірки того, що нові зміни в коді не порушили існуючу функціональність:
- Тестування користувацького інтерфейсу (UI Testing): Widget-тестування UI дозволяє перевірити взаємодію користувача з додатком, його зручність та відповідність дизайну.

30

Апробація результатів



Було опубліковано тези доповіді на тему: “Об’єднання платформ. Можливості .NET MAUI для розробників” на VII міжнародній студентській науковій конференції “Сучасні аспекти та перспективні напрямки розвитку науки”, 24 травня 2024 рік, м. Дніпро Україна.

31

Висновки

Проведено аналіз предметної галузі, сформовано вимоги до системи, описано архітектуру, проектування та тестування програмного забезпечення.

Розроблено мобільний додаток для організації практичних занять в автошколі.

Основний функціонал: керування розкладом, запис на заняття, оплата, скасування, відстеження прогресу.

Використано фреймворк .NET MAUI для розробки під Android та iOS.

Застосовано механізм Preferences для локального зберігання даних.

Додаток задовольняє потреби інструкторів та студентів, оптимізуючи процес проведення занять.

32

Дякую за увагу!

ДОДАТОК В

Тези доповіді на VII міжнародній студентській конференції “Сучасні аспекти та перспективні напрямки розвитку науки”

Рисунок В.1 – Сертифікат учасника конференції

ОБ’ЄДНАННЯ ПЛАТФОРМ: МОЖЛИВОСТІ .NET MAUI ДЛЯ РОЗРОБНИКІВ

Панасенко Ігор Сергійович
здобувач вищої освіти факультету комп’ютерних наук
«Харківський національний університет радіоелектроніки», Україна
Науковий керівник: Олійник Олена Володимирівна
Старший викладач кафедри програмної інженерії
«Харківський національний університет радіоелектроніки», Україна

.NET MAUI (Multi-platform App UI) є сучасним крос-платформеним фреймворком для розробки мобільних і десктопних додатків з єдиною кодовою базою. Це наступний етап у розвитку Xamarin.Forms, який поєднує простоту розробки мобільних додатків з масштабованістю та потужністю .NET. .NET MAUI дозволяє розробникам створювати додатки для iOS, Android, macOS та Windows, використовуючи єдиний кодовий базис та спільний UI-компоненти, написані мовами C# та XAML [1].

З моменту свого випуску у травні 2022 року, .NET MAUI привернула значну увагу розробників, особливо тих, хто вже знайомий з екосистемою .NET. Завдяки потужним інструментам і можливостям, .NET MAUI швидко стає популярним вибором для створення крос-платформених додатків. За даними опитувань, багато розробників, які раніше використовували Xamarin.Forms, активно переходять на .NET MAUI. Відомі компанії, такі як NBC Sports Next та Escola Ágil, вже впровадили .NET MAUI у свої проекти, що демонструє довіру до цієї технології серед великих гравців ринку [2]. Однак, незважаючи на зростаючу популярність, деякі розробники висловлюють незадоволення через існуючі проблеми з продуктивністю та стабільністю фреймворку [3].

Однією з ключових переваг MAUI є його здатність компілювати код у нативний код платформи. Це досягається за допомогою Ahead-of-Time (AOT) компіляції для Android та Just-In-Time (JIT) компіляції для інших платформ, таких як IOS і Windows. Використання AOT компіляції дозволяє перевести код на машинний рівень до виконання, що зменшує час запуску додатку та покращує його продуктивність. Крім того, це забезпечує більш ефективне використання ресурсів пристрою та покращує загальну швидкодію додатків.

Переваги компіляції в нативний код [1]:

- Покращена продуктивність: Компіляція в нативний код значно підвищує продуктивність додатків, зменшуючи час запуску та забезпечуючи більш ефективне використання ресурсів пристрою.

- Зменшення навантаження на процесор: Нативний код виконується швидше і з меншою кількістю ресурсів, що знижує навантаження на процесор і покращує енергоефективність додатку.
- Плавніший інтерфейс користувача: Завдяки зменшенню затримок і оптимізації виконання, користувачі отримують більш плавний та швидкий досвід роботи з додатком.

Окрім цього можна виділити такі переваги використання фреймворку .NET MAUI:

- Краща крос-платформна розробка: .NET MAUI дозволяє використовувати один код для різних платформ, таких як IOS, Android, macOS та Windows, що значно спрощує процес розробки та підтримки додатків [1].
- Зручна робота з інтерфейсом користувача: Використання звичних інструментів та мов програмування, таких як C# та XAML, дозволяє легко створювати стильні та функціональні інтерфейси користувача [1].
- Використання популярного шаблону проектування MVVM (Model-View-ViewModel), що спрощує розробку, тестування та підтримку додатків, забезпечуючи чистоту коду та зручну структуру проекту [4].
- Розділення обов'язків: MVVM дозволяє розділити додаток на три основні компоненти: модель (Model), вигляд (View) та модель-представлення (ViewModel), що полегшує розробку великих та складних додатків.
- Підтримка двунаправленого зв'язку даних: MVVM забезпечує механізм двунаправленого зв'язку даних між моделлю та виглядом, що дозволяє автоматично оновлювати інтерфейс користувача при зміні даних в програмі.
- Інтерфейсні тести: Завдяки розділенню логіки додатку та його інтерфейсу, використання MVVM спрощує написання інтерфейсних тестів, що дозволяє швидко та ефективно виявляти та виправляти помилки.

- Підтримка асинхронної роботи: .NET MAUI та архітектура MVVM ідеально підходять для розробки додатків, що використовують асинхронні операції, такі як завантаження даних з мережі чи робота з базою даних [5].
- Широкі можливості розширення: .NET MAUI та MVVM надають широкі можливості для розширення функціональності додатку за допомогою сторонніх бібліотек та інструментів.

Однак незважаючи на досить велику кількість позитивних аспектів, існують також негативні сторони у використанні даної платформи.

- Складність вивчення: Для новачків може знадобитися досить тривалий час, щоб оволодіти .NET MAUI та розуміти концепції MVVM, що може вимагати додаткового часу та зусиль.
- Можливість виникнення проблем з продуктивністю: Недоліки або неоптимальне використання пам'яті можуть призвести до проблем з продуктивністю додатку, особливо на мобільних пристроях з обмеженими ресурсами [5].
- Обмеження крос-платформеності: Хоча .NET MAUI спрощує розробку для різних платформ, можуть виникнути ситуації, коли необхідно писати платформо-специфічний код для досягнення певної функціональності.
- Сумісність зі старим кодом: Перехід на .NET MAUI та MVVM може вимагати переписування частин існуючого коду, особливо якщо він базується на застарілих технологіях або підходах [4].
- Можливість зростання розміру додатку: З використанням крос-платформених інструментів та бібліотек може збільшитися розмір додатку, що може вплинути на час завантаження та використання ресурсів пристрою.

У підсумку можемо підкреслити, що .NET MAUI є потужним інструментом для розробки крос-платформених додатків, що дозволяє значно скоротити час та зусилля, необхідні для створення додатків для різних платформ. Завдяки підтримці MVVM архітектури, розробка стає більш структурованою, а код

чистішим та легшим для тестування. Однак, як і будь-яка технологія, .NET MAUI має свої мінуси, такі як складність вивчення для новачків та можливі проблеми з продуктивністю. Попри це, .NET MAUI продовжує набирати популярність і обіцяє стати важливим інструментом у арсеналі розробників крос-платформених додатків у 2024 році та надалі. Оскільки відбувається активний розвиток цього фреймвору, для успішного та ефективного використання .NET MAUI розробникам необхідно вкласти час у його детальне вивчення, а після підтримувати актуальні знання даної технології.

Загалом, .NET MAUI є чудовим вибором для розробників, які вже працюють з екосистемою .NET і хочуть створювати ефективні та красиві крос-платформові додатки.

Список використаних джерел:

1. What is .NET MAUI? - .NET MAUI. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui?view=net-maui-8.0 (дата звернення: 17.05.2024).

2. .NET MAUI customers showcase | .NET. *Microsoft*. URL: https://dotnet.microsoft.com/en-us/platform/customers/maui (дата звернення: 17.05.2024).

3. Speed R. Microsoft .NET MAUI mired in developer complaints. *The Register: Enterprise Technology News and Analysis*. URL: https://www.theregister.com/2023/11/29/microsoft_net_maui_devs_unhappy/ (дата звернення: 17.05.2024).

4. Luchaninov Y. 2024 Cross-Platform Mobile App Development Frameworks Compared. *MobiDev*. URL: https://mobidev.biz/blog/cross-platform-mobile-development-frameworks-comparison (дата звернення: 17.05.2024).

5. Is .NET MAUI the Right Investment for Your Brand?. *Bottle Rocket*. URL: https://www.bottlerocketstudios.com/news-views/is-net-maui-the-right-investment-for-your-brand/ (дата звернення: 17.05.2024).