

**Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення**

**КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»**

**РОЗРОБКА КРОСПЛАТФОРМНОГО МОБІЛЬНОГО ЗАСТОСУНКУ
ДЛЯ ВИВЧЕННЯ ПРАВИЛ ДОРОЖНЬОГО РУХУ**

**DEVELOPMENT OF A CROSS-PLATFORM MOBILE APPLICATION FOR
STUDYING TRAFFIC RULES**

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗ-44
Васюренко Віталій Ігорович

Керівник:
Бойко Лев Степанович

Кваліфікаційну роботу
допущено до захисту
«__» _____ 20__ р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна

Луцьк – 2026 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних та інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Галузь знань: 12 «Інформаційні технології»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітня програма: «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

«__» _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Васюренку Віталію Ігоровичу

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «Розробка кросплатформного мобільного застосунку для вивчення правил дорожнього руху»

Керівник роботи: доцент Бойко Л. С.

затверджені наказом закладу вищої освіти від «31» грудня 2025 р. № 541/01-02

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра «29» травня 2026 р.

3. Вихідні дані до роботи: предметна область вивчення правил дорожнього руху, технології кросплатформної мобільної розробки (Flutter, Dart), засоби локального зберігання даних (Hive, SharedPreferences), принципи проєктування мобільних інформаційних систем, сучасні підходи до організації навчального процесу та реалізації користувацького інтерфейсу мобільних застосунків.

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): аналіз сучасного стану проблем використання мобільних технологій для вивчення правил дорожнього руху, аналіз існуючих програмних рішень та наукових досліджень у предметній області, постановка завдання, формування вимог до мобільного застосунку, проєктування програмного забезпечення, обґрунтування вибору технологій та засобів розробки, практична реалізація кросплатформного мобільного застосунку, розробка структури локального зберігання даних, створення користувацької документації, формування інсталяційного пакета, тестування та налагодження програмного продукту.

5. Перелік графічного матеріалу: 23 рисунки, 18 лістингів, 1 таблиця.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	<i>Бойко Л. С.</i>		
Специфікація вимог до розробленої системи	<i>Бойко Л. С.</i>		
Розробка об'єкта проектування	<i>Бойко Л. С.</i>		
Нормоконтроль	<i>Суринович О. М.</i>		
Гарант ОП	<i>Ліщина Н. М.</i>		
Показник запозичень тексту	_____ %		
Академічна доброчесність	<i>Бойко Л. С.</i>		

7. Дата видачі завдання «3» лютого 2026 р.

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 14.02.2026 р.	
2	Аналіз проблеми розробки та впровадження об'єкту проектування	до 03.03.2026 р.	
3	Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 14.03.2026 р.	
4	Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних	до 14.04.2026 р.	
5	Практична реалізація об'єкта проектування та розробка бази даних	до 20.04.2026 р.	
6	Тестування та налагодження об'єкта проектування	до 04.05.2026 р.	
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедрі	до 29.05.2026 р.	

Здобувач вищої освіти _____

Віталій ВАСЮРЕНКО

Керівник кваліфікаційної роботи _____

Лев БОЙКО

АНОТАЦІЯ

Васюренко В. І. Розробка кросплатформного мобільного застосунку для вивчення правил дорожнього руху. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2026.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків та списку використаних джерел.

У першому розділі проведено дослідження предметної області, пов'язаної з використанням мобільних технологій для вивчення правил дорожнього руху, проаналізовано існуючі програмні рішення та сучасні наукові дослідження, визначено їх переваги і недоліки, а також обґрунтовано доцільність розробки нового програмного продукту. У другому розділі сформовано вимоги до мобільного застосунку, виконано проектування системи із застосуванням UML-нотації та обґрунтовано вибір інструментів, засобів, методів і технологій розробки. У третьому розділі описано процес реалізації кросплатформного мобільного застосунку, розроблено структуру локального зберігання даних, підготовлено користувацьку документацію, сформовано інсталяційний пакет, а також наведено результати тестування та налагодження програмного забезпечення. У висновках підсумовано отримані результати, підтверджено досягнення мети роботи та повне виконання поставлених завдань.

Ключові слова: мобільний застосунок, правила дорожнього руху, Flutter, Dart, кросплатформна розробка, тестування знань, Hive, SharedPreferences, Android.

ABSTRACT

Vasiurenko V. I. Development of a Cross-Platform Mobile Application for Studying Traffic Rules. Manuscript. Bachelor's qualification work of the Educational Program “Software Engineering” in the specialty “Software Engineering”. Lutsk National Technical University. Lutsk, 2026.

The bachelor's qualification work consists of an introduction, three chapters, conclusions, and a list of references.

The first chapter presents a study of the subject area related to the use of mobile technologies for learning traffic rules, analyzes existing software solutions and modern scientific research, identifies their advantages and disadvantages, and substantiates the feasibility of developing a new software product. The second chapter defines the requirements for the mobile application, describes the system design using UML notation, and justifies the choice of development tools, methods, techniques, and technologies. The third chapter describes the implementation process of the cross-platform mobile application, develops the structure of local data storage, prepares user documentation, creates an installation package, and presents the results of software testing and debugging. The conclusions summarize the obtained results, confirm the achievement of the work objective, and the complete fulfillment of the assigned tasks.

Keywords: mobile application, traffic rules, Flutter, Dart, cross-platform development, knowledge testing, Hive, SharedPreferences, Android.

ЗМІСТ

ВСТУП	7
РОЗДІЛ 1 АНАЛІЗ МОБІЛЬНИХ ЗАСТОСУНКІВ ДЛЯ ВИВЧЕННЯ ПРАВИЛ ДОРОЖНЬОГО РУХУ ТА ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ	9
1.1 Аналіз сучасного стану проблеми	9
1.2 Постановка завдання на кваліфікаційну роботу бакалавра	14
РОЗДІЛ 2 СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ	16
2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення	16
2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання	21
РОЗДІЛ 3 РОЗРОБКА КРОСПЛАТФОРМНОГО МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ВИВЧЕННЯ ПРАВИЛ ДОРОЖНЬОГО РУХУ	26
3.1 Практична реалізація об'єкта проектування	26
3.2 Тестування та налагодження інформаційно-комп'ютерної системи	42
3.3 Розробка локальної бази даних застосунку	44
3.4 Розробка документації для програмного продукту	49
3.5 Розробка інсталяційного пакета	53
ВИСНОВКИ	57
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	59

ВСТУП

Актуальність теми. У сучасних умовах розвитку інформаційних технологій і широкого поширення мобільних пристроїв цифрові засоби навчання стають важливою складовою освітнього процесу. Особливо актуальним є використання мобільних застосунків для вивчення правил дорожнього руху, оскільки безпека дорожнього руху значною мірою залежить від підготовки майбутніх водіїв та якості засвоєння теоретичних знань. Мобільні застосунки забезпечують швидкий доступ до навчальних матеріалів, можливість контролю знань і підвищують зручність підготовки до теоретичного іспиту.

Сучасні тенденції розвитку мобільних рішень орієнтовані на використання кросплатформних підходів, підвищення зручності користувацького інтерфейсу, інтеграцію елементів гейміфікації та персоналізацію навчального процесу. Водночас багато застосунків для вивчення правил дорожнього руху зосереджені переважно на тестуванні та підготовці до іспиту, приділяючи недостатньо уваги мотивації користувачів, поясненню помилок і підтримці під час навчання.

Метою кваліфікаційної роботи бакалавра є розробка кросплатформного мобільного застосунку для вивчення правил дорожнього руху з використанням сучасних технологій мобільної розробки та інтерактивних методів навчання.

Об'єктом дослідження є процес навчання користувачів правилам дорожнього руху із застосуванням мобільних інформаційних технологій.

Предметом дослідження є методи, засоби та технології розробки кросплатформних мобільних застосунків для вивчення правил дорожнього руху, а також організація навчального процесу за допомогою цифрових інструментів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- провести аналіз предметної області, сучасних програмних рішень та наукових досліджень у сфері мобільних технологій навчання;
- виконати проєктування програмного забезпечення із застосуванням UML-діаграм та засобів моделювання програмних систем;

- реалізувати кросплатформний мобільний застосунок із використанням фреймворку Flutter та мови програмування Dart;
- розробити модулі теоретичного навчання та довідкових матеріалів із використанням локального зберігання даних;
- реалізувати тренувальний та екзаменаційний режими тестування із застосуванням алгоритмів випадкового формування запитань та перевірки відповідей користувача;
- реалізувати систему збереження статистики, нарахування досвіду, досягнень, персонального помічника та налаштувань користувача з використанням технологій Hive та SharedPreferences;
- реалізувати адаптивний користувацький інтерфейс із підтримкою світлої та темної тем оформлення засобами Flutter Material Design;
- провести тестування та налагодження програмного продукту, сформувати інсталяційні пакети APK і AAB та забезпечити можливість публікації застосунку в магазині Google Play.

Практичне значення роботи полягає у можливості використання розробленого програмного продукту для самостійного вивчення правил дорожнього руху, підготовки до складання теоретичного іспиту та підвищення рівня обізнаності у сфері безпеки дорожнього руху. Галуззю застосування розробленої системи є освітні та інформаційні мобільні сервіси, а результати роботи можуть бути використані учнями автошкіл, майбутніми водіями та всіма зацікавленими особами для вивчення й перевірки знань правил дорожнього руху.

РОЗДІЛ 1

АНАЛІЗ МОБІЛЬНИХ ЗАСТОСУНКІВ ДЛЯ ВИВЧЕННЯ ПРАВИЛ ДОРОЖНЬОГО РУХУ ТА ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

Нині існує велика кількість мобільних застосунків для вивчення правил дорожнього руху та підготовки до теоретичного іспиту. Більшість таких систем орієнтовані на проходження тестів, перегляд навчальних білетів, ознайомлення з дорожніми знаками та перевірку рівня знань користувача. Стрімкий розвиток мобільних технологій та поширення смартфонів посприяли появі значної кількості цифрових рішень для навчання правилам дорожнього руху. Сучасні додатки дозволяють користувачам проходити тести у будь-який час, повторювати складні теми та отримувати доступ до навчальних матеріалів без необхідності використання друкованих посібників.

Серед найпопулярніших застосунків в Україні можна виділити GREEN-WAY [1], ПДР України та Штрафи UA [2]. Вони мають різний функціонал та орієнтовані на різні категорії користувачів, проте всі вони пов'язані із вивченням правил дорожнього руху або використанням інформації для водіїв.

Одним із найбільш відомих застосунків для вивчення ПДР є GREEN-WAY зображений на рисунку 1.1. Даний застосунок містить велику базу тестових питань, режим проходження екзамену, статистику результатів, тематичні категорії та навчальні матеріали. До переваг цієї системи можна віднести широкий функціонал, регулярне оновлення питань відповідно до актуальних правил дорожнього руху, високу популярність серед користувачів та можливість комплексної підготовки до теоретичного іспиту. Крім цього, застосунок забезпечує швидкий доступ до навчальних матеріалів та підтримує різні режими тестування. Проте цей додаток має і певні недоліки. Інтерфейс містить значну кількість елементів та функцій, які можуть ускладнювати використання для нових користувачів і дітей. Основний акцент у застосунку зроблений саме на

підготовку до складання іспиту, а не на поступове навчання користувача та пояснення правил дорожнього руху. Тут відсутній персональний супровід, інтерактивний помічник і механізми адаптації навчального процесу для різних вікових категорій користувачів.

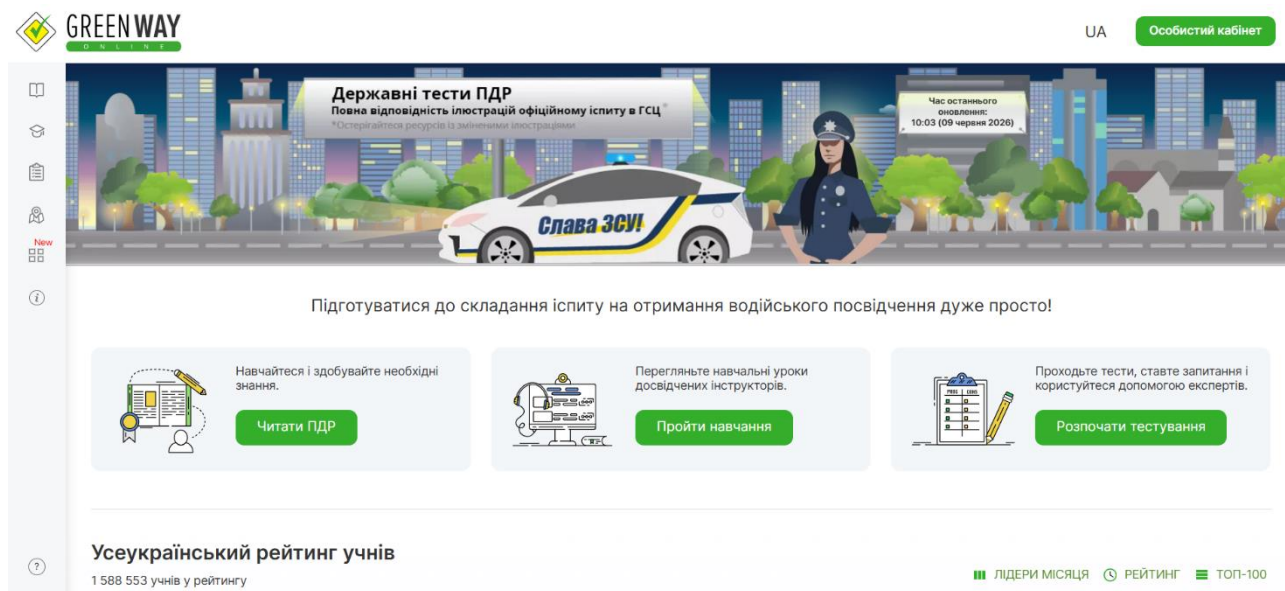


Рисунок 1.1 – Зображення інтерфейсу GREEN-WAY

Також популярними є різні мобільні застосунки під загальною назвою ПДР України. Вони зазвичай надають доступ до тестів, правил дорожнього руху, дорожніх знаків та режимів тренування. Перевагами подібних додатків є можливість офлайн-використання, швидкий доступ до навчальних матеріалів, простота проходження тестів та невисокі системні вимоги. Більшість таких програм дозволяють користувачам проходити підготовку без постійного доступу до мережі Інтернет, що є важливою перевагою для мобільних застосунків.

Однак велика частина таких рішень має застарілий дизайн та недостатній рівень інтерактивності. Процес навчання користувача часто обмежується проходженням тестів без пояснення помилок і надання зворотного зв'язку. Крім цього, у більшості систем відсутні елементи гейміфікації, системи мотивації користувачів, нагороди або інтерактивні механізми взаємодії, які могли б зробити навчальний процес цікавішим та ефективнішим.

Серед активних водіїв популярним є застосунок Штрафи UA зображений на рисунку 1.2. Його основне призначення полягає у перегляді штрафів, інформації про транспортні засоби та отриманні довідкової інформації. Застосунок також містить і окремі матеріали, пов'язані з правилами дорожнього руху та дорожніми знаками. До переваг системи можна віднести сучасний інтерфейс, інтеграцію різних сервісів та зручність її використання. Проте цей додаток орієнтований переважно на досвідчених водіїв, а не на користувачів, які лише починають вивчати правила дорожнього руху. У застосунку відсутні навчальні сценарії, інтерактивні режими підготовки, елементи гейміфікації та система адаптивного навчання. Крім цього, система не забезпечує покрокового пояснення помилок користувача та не містить інструментів для підтримання мотивації під час навчального процесу.

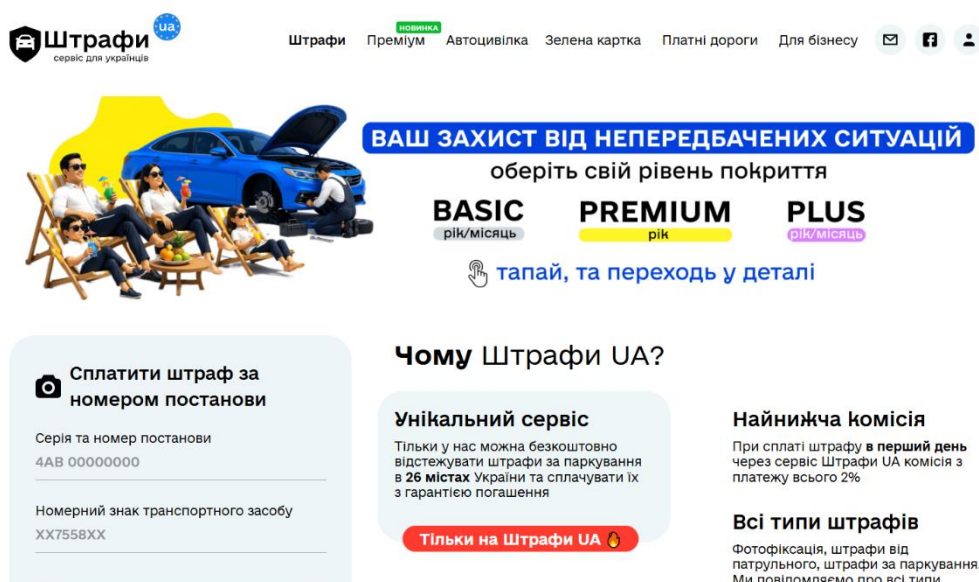


Рисунок 1.2 – Зображення інтерфейсу Штрафи UA

Проведений аналіз аналогів показав, що більшість сучасних мобільних застосунків для вивчення правил дорожнього руху орієнтовані на підготовку до складання теоретичного іспиту та проходження тестових завдань. При цьому у багатьох системах недостатньо уваги приділяється зручності користувацького інтерфейсу, інтерактивності навчального процесу та поясненню помилок користувача під час навчання. Значна частина додатків має перевантажений

інтерфейс або складну структуру навігації, що може створювати труднощі для нових користувачів та дітей.

Також у більшості розглянутих застосунків відсутні елементи гейміфікації та механізми персоналізованого супроводу та адаптації навчального процесу під користувача, які здатні підвищити мотивацію та зацікавленість під час навчання. Існуючі рішення зосереджені на механічному проходженні тестів, а не на поступовому та зрозумілому засвоєнні правил дорожнього руху. Виявлені недоліки існуючих програмних рішень зумовлюють необхідність аналізу сучасних наукових досліджень у сфері мобільної розробки та цифрових технологій навчання.

У межах аналізу наукових робіт у сфері мобільної розробки було виявлено стрімке зростання популярності кросплатформних технологій [3]. Це пов'язано зі збільшенням кількості мобільних пристроїв та потребою створення додатків для різних операційних систем. Зазначена тенденція зумовлює активне впровадження кросплатформних підходів у процес розробки мобільних рішень.

Нині кросплатформні фреймворки використовуються дедалі частіше, оскільки вони дають змогу скоротити час розробки, зменшити витрати на підтримку програмного забезпечення та використовувати єдину кодову базу для різних операційних систем. Також у дослідженні [3] розглянуто основні підходи до мобільної розробки, серед яких нативна, веб та кросплатформна розробка. Автори зазначають, що сучасні кросплатформні рішення мають переваги над традиційними підходами, оскільки забезпечують продуктивність, наближену до нативної. Окремо підкреслюється, що Flutter є перспективним фреймворком завдяки високій швидкості роботи, активній спільноті розробників та можливості створення адаптивних інтерфейсів користувача [3].

У проаналізованих дослідженнях проведено порівняння нативного та кросплатформного підходів до створення мобільних застосунків. Крім цього проаналізовано особливості використання Android Studio та Flutter, а також досліджено зручність користувацького інтерфейсу, ефективність процесу розробки й продуктивність застосунків, створених із використанням цього

технологічного стеку. Результати експериментів і порівнянь показали, що Flutter дозволяє скоротити час розробки майже на 28 % та зменшити обсяг програмного коду більш ніж на 30 %. Окремо відзначено функцію Hot Reload, яка прискорює процес тестування та внесення змін до інтерфейсу застосунку [4].

Також значна увага в дослідженні [4] приділяється зручності використання мобільних застосунків. За результатами проведеного тестування Flutter отримав вищі показники користувацької зручності порівняно з нативною розробкою. Користувачі позитивно оцінили сучасний зовнішній вигляд застосунків, швидкість роботи інтерфейсу та зручність взаємодії з елементами керування. Отримані результати підтверджують ефективність використання Flutter для створення сучасних мобільних застосунків із простим та зрозумілим інтерфейсом, що є важливим аспектом під час розробки застосунку для вивчення правил дорожнього руху [4].

Окрім технічних аспектів мобільної розробки, важливим фактором також є ефективність процесу навчання та залучення користувачів. Останніми роками особливо актуальними стали дослідження щодо використання цифрових технологій та елементів гейміфікації для навчання правилам дорожнього руху. Інтерактивні платформи для навчання дітей і підлітків основам безпечної поведінки на дорозі, які використовують VR-технології, відеоматеріали та систему миттєвого зворотного зв'язку сприяють підвищенню ефективності навчального процесу та роблять його більш цікавим і зрозумілим [5].

Автори зазначають, що використання гейміфікації позитивно впливає на мотивацію користувачів і покращує засвоєння навчального матеріалу [5]. Особливу увагу приділено віртуальним сценаріям дорожніх ситуацій, де учасники можуть аналізувати ризики, визначати небезпечні ситуації та приймати рішення в умовах, наближених до реального дорожнього середовища. У дослідженні [5] також підкреслюється важливість простого та зрозумілого інтерфейсу для залучення користувачів різних вікових категорій. Отримані результати свідчать про те, що використання інтерактивних технологій та навчання у формі гри дозволяє підвищити рівень знань правил дорожнього руху.

Таким чином, проведений аналіз наукових джерел підтвердив актуальність використання кросплатформних технологій для створення сучасних мобільних застосунків та ефективність використання Flutter для реалізації програмних продуктів із простим і зрозумілим інтерфейсом. Крім того, результати досліджень демонструють доцільність застосування елементів гейміфікації, інтерактивного навчання та системи зворотного зв'язку під час створення програмних засобів для вивчення правил дорожнього руху.

На основі результатів проведеного аналізу наукових робіт, а також з урахуванням виявлених недоліків існуючих програмних рішень, доцільним рішенням буде розробка нового кросплатформного мобільного застосунку для вивчення правил дорожнього руху. Використання сучасних технологій і інтерактивних підходів до навчання сприяє підвищенню зацікавленості користувачів, покращенню засвоєння навчального матеріалу та формуванню навичок безпечної поведінки на дорозі.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

На основі проведеного аналізу предметної області, існуючих програмних рішень та сучасних підходів до розробки мобільних застосунків було визначено основну мету кваліфікаційної роботи бакалавра, яка полягає у розробці кросплатформного мобільного застосунку для вивчення правил дорожнього руху з використанням сучасних технологій мобільної розробки та інтерактивних методів навчання.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- провести аналіз предметної області, існуючих мобільних застосунків для вивчення правил дорожнього руху та сучасних наукових досліджень у сфері мобільної розробки і цифрових технологій навчання;
- виконати проєктування програмного забезпечення із застосуванням UML-нотації та розробити моделі основних елементів мобільного застосунку;

- розробити кросплатформний мобільний застосунок засобами фреймворку Flutter та мови програмування Dart;
- реалізувати модуль навчання, який забезпечить перегляд теоретичних матеріалів, категорій правил дорожнього руху та дорожніх знаків із використанням локального зберігання даних;
- реалізувати тренувальний та екзаменаційний режими тестування з використанням алгоритмів випадкового формування тестових завдань, перевірки відповідей користувача та обчислення результатів тестування;
- реалізувати систему персонального помічника, статистики користувача, нарахування досвіду та досягнень із використанням технологій Hive і SharedPreferences для локального збереження даних;
- розробити адаптивний користувацький інтерфейс із підтримкою світлої та темної тем оформлення засобами Flutter Material Design;
- провести тестування та налагодження програмного продукту, виявити та усунути можливі недоліки, сформувати інсталяційні пакети APK і AAB для платформи Android та здійснити публікацію застосунку в магазині Google Play.

РОЗДІЛ 2

СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проєктування програмного забезпечення

Перед початком розробки мобільного застосунку для вивчення правил дорожнього руху, необхідно здійснити аналіз вимог до програмного забезпечення, визначити основні функціональні можливості системи та виконати моделювання її структури й основних елементів. Це дозволить визначити взаємодію окремих компонентів застосунку, сформувавши структуру даних, описати логіку роботи системи та забезпечити ефективну реалізацію функціональних можливостей програмного продукту зі зручним користувацьким інтерфейсом.

Мета розроблюваного програмного забезпечення полягає у створенні кросплатформного мобільного застосунку для вивчення правил дорожнього руху, який забезпечуватиме зручний доступ до навчальних матеріалів, проходження тестування та інтерактивну підтримку користувача під час навчального процесу. Особливістю цього застосунку буде орієнтація на користувачів різних вікових категорій, використання простого та зрозумілого інтерфейсу, а також наявність персонального помічника для пояснення помилок і підтримки мотивації користувача.

Основні завдання роботи полягають у реалізації тренувального та екзаменаційного режимів тестування, забезпеченні доступу до дорожніх знаків і навчальних матеріалів, збереженні статистики результатів користувача, підтримці офлайн-режиму роботи, впровадженні системи нагород і досягнень, а також забезпеченні зручної навігації та адаптивного користувацького інтерфейсу.

Для моделювання програмної системи було обрано об'єктно-орієнтований підхід та UML-нотацію, оскільки такий підхід дозволяє представити структуру системи, взаємодію її елементів та сценарії використання застосунку. Використання UML-діаграм дає можливість спростити процес проєктування

програмного забезпечення та забезпечити зрозуміле представлення архітектури застосунку.

Визначення та аналіз вимог до програмного забезпечення здійснювалися на основі дослідження предметної області, аналізу аналогів та визначення потреб потенційних користувачів. Основними способами виявлення вимог були аналіз функціональних можливостей аналогічних систем, визначення основних сценаріїв використання застосунку та формування вимог до зручності користувацького інтерфейсу. Особлива увага приділялася забезпеченню простоти використання застосунку для користувачів різних вікових категорій, можливості офлайн-роботи та інтерактивному навчанню з використанням персонального помічника.

Проектування архітектури програмного забезпечення відбувалося із використанням модульного підходу, який передбачає поділ застосунку на окремі функціональні компоненти. Основними модулями системи є модуль проходження тестів, модуль режиму іспиту, модуль статистики, модуль дорожніх знаків, модуль досягнень та модуль персонального помічника. Такий підхід дозволяє забезпечити логічну організацію програмної системи, спростити підтримку застосунку та підвищити зручність взаємодії користувача із системою.

Специфікація вимог до програмного забезпечення включає в себе як функціональні так і нефункціональні вимоги. До функціональних вимог належать проходження тестів правил дорожнього руху, перегляд теоретичних матеріалів, аналіз результатів тестування, збереження статистики користувача, робота персонального помічника, система нагород та підтримка режиму іспиту. До нефункціональних вимог належать простота та зрозумілість інтерфейсу, стабільність роботи застосунку, швидкодія, підтримка офлайн-режиму, кросплатформність та адаптація інтерфейсу для користувачів різних вікових категорій.

Функціональні характеристики розроблюваної системи спрямовані на роботу з локальною базою даних, у якій зберігаються питання, варіанти відповідей, категорії, теоретичні матеріали, статистика користувача, досягнення

та налаштування застосунку. Під час роботи системи має виконуватись оновлення статистичних даних користувача, збереження результатів тестування та обробка відповідей у режимі реального часу. Застосунок не повинен потребувати підключення до мережі Інтернет, оскільки весь матеріал буде зберігатися локально на пристрої користувача.

Оскільки у застосунку не буде реалізовано механізми авторизації та аутентифікації, система не здійснюватиме розмежування прав доступу між користувачами. Контроль введеної інформації реалізовуватиметься шляхом вибору користувачем одного з доступних варіантів відповіді під час тестування. Для забезпечення цілісності даних система оновлюватиме статистику, результати тестування та дані користувача після виконання певних дій. Результати роботи системи відображатимуться у вигляді статистики, повідомлень персонального помічника та підсумкових результатів тестування.

Для формалізованого опису функціональних можливостей розроблюваної системи та аналізу взаємодії користувача із застосунком було використано UML-діаграму варіантів використання. Такий тип діаграм дає можливість представити основні сценарії роботи системи, визначити функції програмного забезпечення та встановити взаємозв'язок між користувачем і окремими компонентами застосунку.

Мобільний застосунок працюватиме повністю офлайн та не передбачатиме створення облікового запису або авторизації користувачів, тому основним актором системи є користувач. Саме користувач взаємодіатиме з усіма функціональними модулями застосунку, проходитиме навчальні тести, переглядатиме інформаційні матеріали та отримуватиме результати тестування. Діаграма варіантів використання відображає основні можливості системи та логіку взаємодії користувача із функціональними елементами. На рисунку 2.1 наведено діаграму варіантів використання мобільного застосунку.

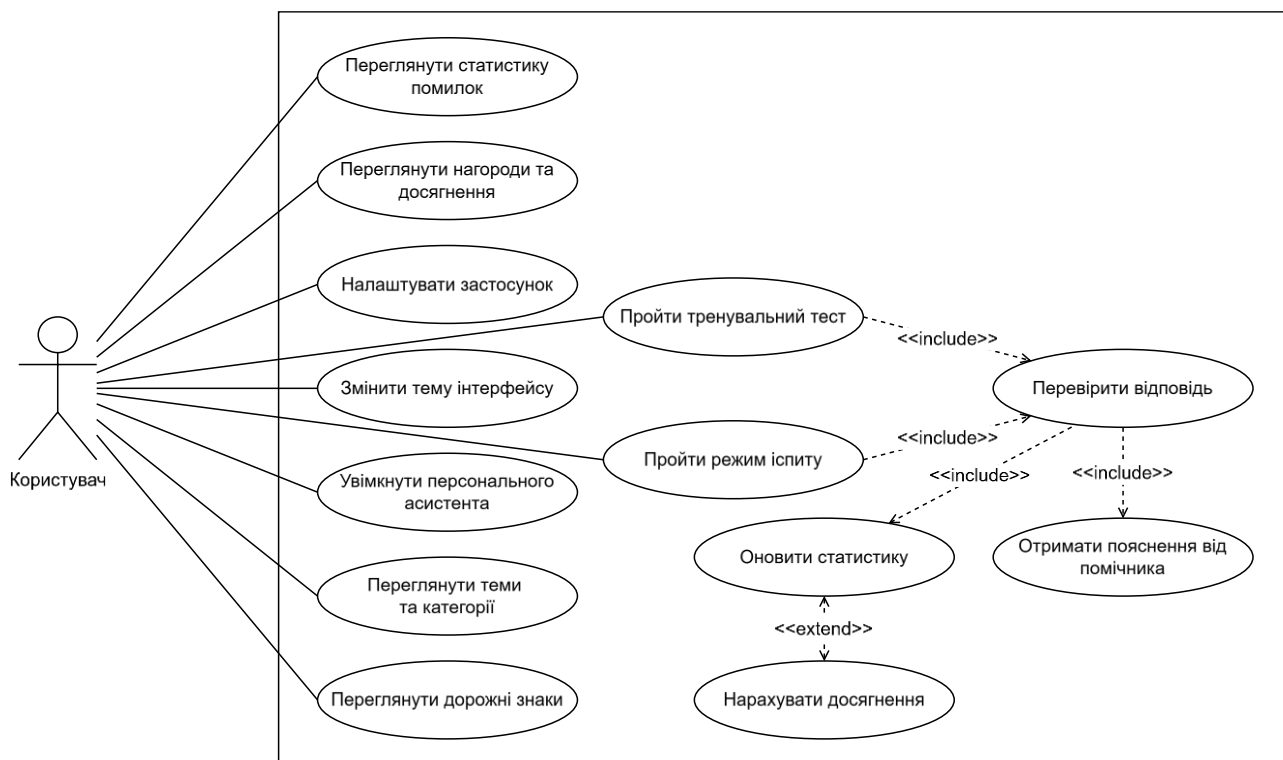


Рисунок 2.1 – Діаграма варіантів використання системи

Основними варіантами використання додатку є проходження тренувальних тестів та режиму іспиту, перегляд навчальних білетів, тем та категорій правил дорожнього руху, аналіз статистики помилок, перегляд нагород і досягнень та зміна теми інтерфейсу застосунку.

Окрему увагу на діаграмі приділено функціоналу проходження тестів, оскільки саме він є основною частиною навчального процесу. Під час проходження тесту система повинна виконувати перевірку відповідей користувача, оновлювати статистику результатів та забезпечувати взаємодію із персональним помічником. У разі правильної відповіді помічник позитивно реагує на дії користувача, а у випадку помилки система відображає правильну відповідь та пояснення. Крім цього, після завершення тестування система зможе нараховувати користувачу нагороди та досягнення залежно від його результатів.

Для детального опису логіки роботи мобільного застосунку та послідовності дій користувача розроблено UML-діаграму активності, яка дозволяє відобразити процес виконання операцій, взаємодію користувача із

системою та особливості роботи окремих модулів застосунку. Діаграма демонструє послідовність дій користувача і відповідні реакції системи.

На рисунку 2.2 наведено діаграму активності проходження тестування у мобільному застосунку. У діаграмі розмежовано дії користувача та системи, що дає можливість продемонструвати взаємодію між користувачем і програмним забезпеченням у процесі навчання.

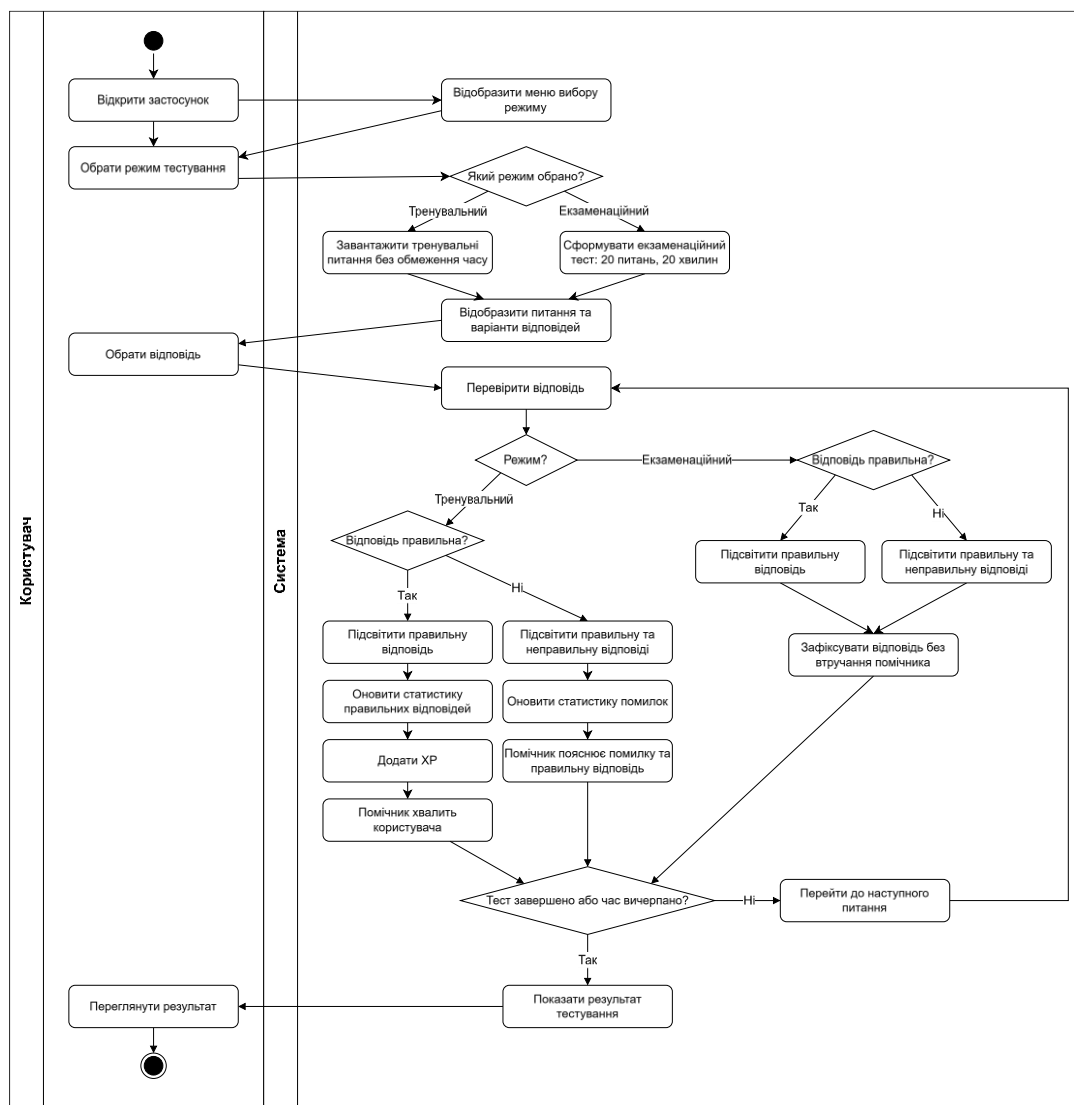


Рисунок 2.2 – Діаграма активності проходження тестування

Процес роботи із системою починається з відкриття користувачем мобільного застосунку та вибору режиму тестування. Користувач може обрати один із двох режимів: тренувальний або екзаменаційний. Після вибору режиму система завантажує відповідний сценарій тестування.

У тренувальному режимі користувачу надається можливість проходження тестів без обмеження часу та кількості питань. Після вибору відповіді система виконує перевірку результату, якщо відповідь правильна то додаток підсвічує правильний варіант відповіді, оновлює статистику правильних відповідей та нараховує користувачу ігровий досвід. Крім цього, персональний помічник відображає мотиваційне повідомлення для підтримки зацікавленості користувача. Якщо відповідь неправильна, додаток відображає правильний і неправильний варіанти відповіді, оновлює статистику помилок, а персональний помічник пояснює причину помилки та аргументує правильну відповідь.

Під час проєктування користувацького інтерфейсу важливо забезпечити комфортну взаємодію користувача із системою та зручність використання застосунку для користувачів різних вікових категорій. Основні екрани системи повинні забезпечувати швидкий доступ до режимів тестування, теоретичних матеріалів, статистики та налаштувань застосунку. Для покращення сприйняття інформації має бути впроваджено мінімалістичний дизайн інтерфейсу, адаптивне розташування елементів керування і підтримку світлої й темної тем оформлення.

Інформаційні потоки в системі організовані з метою забезпечення швидкої взаємодії користувача з функціональними модулями застосунку. Користувач через інтерфейс взаємодіє з модулями тестування, статистики, налаштувань, теоретичних матеріалів та персональним помічником, після чого результати зберігаються у локальній базі даних та відображаються в інтерфейсі застосунку, що забезпечує стабільну роботу та комфортне використання додатку без необхідності постійного доступу до мережі Інтернет.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Серед сучасних засобів розробки мобільних застосунків можна виділити нативну розробку для Android, кросплатформний фреймворк React Native та SDK Flutter. Нативна розробка дозволяє створювати високопродуктивні

застосунки, проте потребує окремої реалізації для кожної платформи, що збільшує складність підтримки програмного забезпечення та витрати часу на розробку. React Native забезпечує можливість створення кросплатформних застосунків із використанням JavaScript, однак має певні обмеження у продуктивності та реалізації складних елементів інтерфейсу [6].

Flutter, як і React Native дозволяє створювати кросплатформні мобільні застосунки з використанням єдиної кодової бази для різних платформ, що значно спрощує процес розробки та підтримки програмного продукту. До основних переваг Flutter належать висока продуктивність, сучасний механізм побудови адаптивного інтерфейсу користувача з плавними анімаціями та швидким оновленням застосунку завдяки функції Hot Reload, а також підтримка різних платформ, зокрема Android, Web та Windows [7].

Основною мовою програмування під час розробки застосунку було обрано Dart, яка є базовою мовою для Flutter та забезпечує високу продуктивність, підтримку асинхронного програмування й ефективну роботу з кросплатформними застосунками [8].

Для реалізації локального зберігання даних у застосунку було обрано Hive та SharedPreferences. Hive використовується для збереження даних профілю користувача, статистики, досягнень та інших структурованих даних у застосунку і його основними перевагами є висока швидкість роботи, простота інтеграції та можливість роботи без підключення до мережі Інтернет [9]. SharedPreferences використовується для збереження налаштувань застосунку, зокрема теми оформлення, параметрів персонального помічника та історії тестування [10].

Реалізація логіки роботи застосунку здійснюється із використанням StatefulWidget, setState та FutureBuilder, які дозволяють ефективно керувати станом інтерфейсу користувача, оновлювати дані в режимі реального часу та забезпечувати взаємодію між окремими компонентами системи. Застосування такого підходу зумовлене тим, що додаток має локальну архітектуру, не використовує складну клієнт-серверну взаємодію та не потребує застосування складних механізмів керування станом.

Під час розробки програмного забезпечення також використано підхід MVVM у поєднанні з багаторівневою архітектурою, що дозволяє розділити інтерфейс користувача, бізнес-логіку та роботу з даними на окремі логічні рівні, спростити підтримку програмного коду та підвищити зрозумілість структури застосунку [11]. Рівень представлення відповідає за відображення інтерфейсу користувача, рівень логіки реалізує обробку тестування та взаємодію з персональним помічником, а рівень даних забезпечує роботу з локальним збереженням інформації за допомогою Hive та SharedPreferences.

Для представлення архітектури додатку з основними компонентами їх взаємодії та зв'язками між окремими модулями системи побудовано UML-діаграму компонентів. На рисунку 2.3 наведено діаграму компонентів розроблюваного мобільного застосунку.

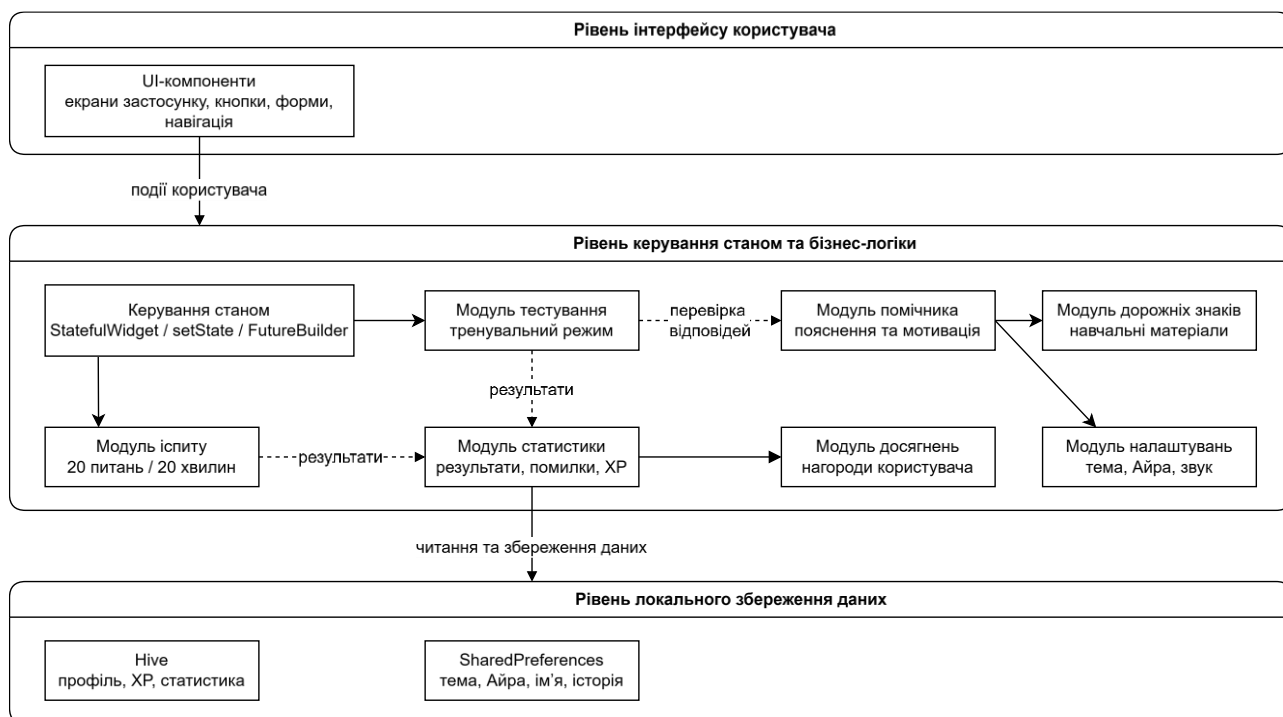


Рисунок 2.3 – Діаграма компонентів системи

Структура застосунку поділена на декілька логічних рівнів: рівень інтерфейсу користувача, рівень керування станом та бізнес-логіки, а також рівень локального збереження даних. Рівень інтерфейсу користувача відповідає за відображення екранів застосунку, елементів навігації та взаємодію користувача

із системою. На рівні бізнес-логіки розташовані основні функціональні модулі додатку, зокрема модуль тренувального тестування, екзаменаційного режиму, персонального помічника, статистики, досягнень та навчальних матеріалів, а рівень локального збереження даних відповідає за збереження профілю користувача, статистики та досвіду, параметрів застосунку та налаштувань.

Для опису логіки перевірки відповідей користувача було побудовано блок-схему алгоритму перевірки відповіді (рис. 2.4). Використання блок-схеми дозволяє відобразити послідовність виконання операцій та процес обробки відповідей користувача у системі під час проходження тестування.

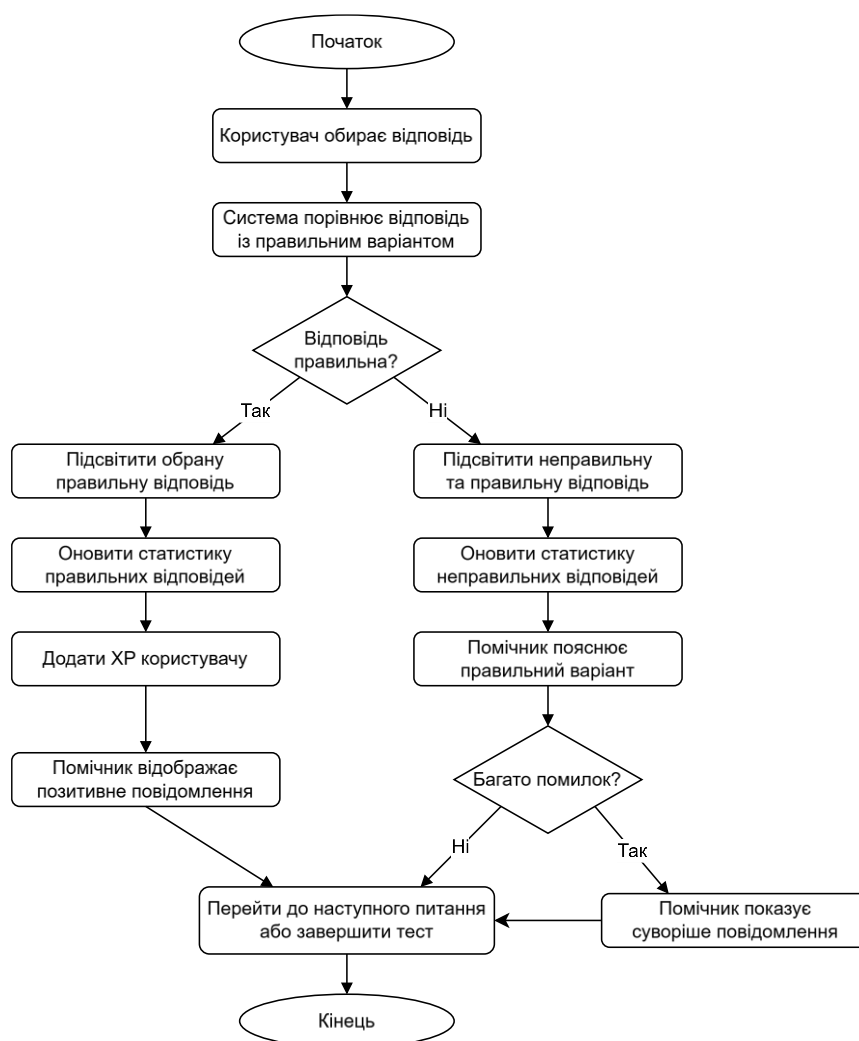


Рисунок 2.4 – Блок-схема алгоритму перевірки відповіді

Робота алгоритму починається з вибору користувачем одного із запропонованих варіантів відповіді. Після цього система виконує порівняння

обраної відповіді із правильним варіантом та визначає результат перевірки. У разі правильної відповіді система підсвічує правильний варіант, оновлює статистику правильних відповідей, нараховує користувачу – ігровий досвід та відображає позитивне повідомлення від персонального помічника.

Якщо відповідь користувача є неправильною, система підсвічує правильний та неправильний варіанти відповіді, оновлює статистику помилок та відображає пояснення від персонального помічника. Додатково система аналізує кількість неправильних відповідей користувача, і у випадку великої кількості помилок помічник може відображати суворіші повідомлення для стимулювання уважнішого проходження тестування. Після завершення перевірки система виконує перехід до наступного питання або завершує тестування з відображенням результату.

Побудована блок-схема дозволяє продемонструвати алгоритм роботи системи під час проходження тестування, логіку оновлення статистики та механізм взаємодії користувача із персональним помічником. Результати моделювання підтверджують, що розроблений алгоритм забезпечує не лише перевірку знань користувача, а й підтримує інтерактивний навчальний процес із використанням елементів мотивації та пояснення помилок.

Таким чином, обрані програмні засоби, методи та алгоритми дають можливість реалізувати кросплатформний мобільний застосунок із підтримкою офлайн-режиму роботи, інтерактивного навчання та локального збереження даних. Використання Flutter, Dart і локальних засобів збереження інформації забезпечить ефективне впровадження функціональних можливостей системи, а побудовані діаграми та блок-схеми допоможуть сформувати чітку структуру програмного забезпечення та логіку роботи основних модулів застосунку.

РОЗДІЛ 3

РОЗРОБКА КРОСПЛАТФОРМНОГО МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ВИВЧЕННЯ ПРАВИЛ ДОРОЖНЬОГО РУХУ

3.1 Практична реалізація об'єкта проєктування

Розробку мобільного застосунку для вивчення правил дорожнього руху було розпочато з реалізації головного модуля для запуску додатку. Основним файлом проєкту є `main.dart`, у якому виконується початкова ініціалізація застосунку, підключення локального сховища даних, перевірка наявності імені користувача та запуск головного екрану. Саме з цього файлу починається робота мобільного застосунку після його запуску на пристрої користувача.

У файлі `main.dart` реалізовано механізм початкового налаштування додатку, код якого наведено в лістингу 3.1. Після запуску виконується реєстрація моделі профілю користувача та відкриття локального сховища `Hive`. Наступним кроком є перевірка наявності збереженого імені користувача і якщо ім'я відсутнє, відкривається екран первинного налаштування, в іншому випадку (дані вже збережені) – користувач одразу переходить на головний екран.

Лістинг 3.1 – Ініціалізація та запуск мобільного застосунку

```
Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();

  await Hive.initFlutter();
  Hive.registerAdapter(UserProfileAdapter());
  await Hive.openBox<UserProfile>("user_profile");

  final hasName = await UserNameManager.hasName();

  runApp(PDRApp(firstStart: !hasName));
}
```

кінець лістингу 3.1

Наведений фрагмент коду демонструє використання асинхронної ініціалізації перед запуском застосунку. Виклик функції `WidgetsFlutterBinding.ensureInitialized()` забезпечує коректну роботу Flutter перед

виконанням асинхронних операцій [12]. Після завершення ініціалізації визначається сценарій запуску залежно від наявності збережених даних користувача, а отриманий результат використовується для вибору початкового екрана застосунку через параметр `firstStart`.

Завершивши початкову ініціалізацію запускається віджет `PDRApp`, який наслідується від `StatefulWidget` та відповідає за конфігурацію користувацького інтерфейсу. У методі `build` даного віджета створюється об'єкт `MaterialApp`, у якому визначаються основні параметри роботи, маршрути навігації, теми інтерфейсу та початковий екран застосунку (ліст. 3.2).

Лістинг 3.2 – Налаштування головного модуля застосунку

```
return MaterialApp(
  title: "ПДР України 2026",
  debugShowCheckedModeBanner: false,

  theme: PdrTheme.light,
  darkTheme: PdrTheme.dark,
  themeMode: _isDark ? ThemeMode.dark : ThemeMode.light,

  home: widget.firstStart
    ? const SetNameScreen(firstSetup: true)
    : const HomeScreen(),

  routes: {
    "/settings": (_) => SettingsScreen(
      toggleTheme: toggleTheme,
      isDark: _isDark,
    ),
    "/history": (_) => const HistoryScreen(),
    "/theory": (_) => const TheoryScreen(),
    "/test": (_) => const TestMenuScreen(),
    "/signs": (_) => const TrafficSignsScreen(),
    "/profile": (_) => const ProfileScreen(),
  },
);
```

кінець лістингу 3.2

У наведеному лістингу відображено основні параметри конфігурації застосунку. За допомогою властивостей `theme`, `darkTheme` та `themeMode` реалізовано підтримку світлої та темної тем інтерфейсу. Вибір активної теми

здійснюється на основі значення змінної `_isDark`, яка завантажується із локального сховища налаштувань. Властивість `home` визначає початковий екран залежно від результатів первинного налаштування користувача. Також у конфігурації задано маршрути навігації, які забезпечують перехід між основними функціональними модулями застосунку.

Після запуску застосунку користувач потрапляє на головний екран, який виконує роль центральної точки навігації та забезпечує доступ до основних функціональних можливостей системи (ліст. 3.3).

Лістинг 3.3 – Реалізація основних елементів головного екрана

```
Column(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Image.asset(
      "assets/images/pdr_logo.png",
      width: 190,
    ),

    _btn(
      icon: Icons.menu_book_rounded,
      text: "Теорія ПДР",
      route: "/theory",
    ),

    _btn(
      icon: Icons.quiz_rounded,
      text: "Тестування",
      route: "/test",
      primary: true,
    ),

    _btn(
      icon: Icons.traffic_rounded,
      text: "Дорожні знаки",
      route: "/signs",
    ),
  ],
)
```

кінець лістингу 3.3

Наведений лістинг демонструє реалізацію центральної частини головного екрана застосунку, зображеного на рисунку 3.1. Для відображення логотипа

використовується віджет Image.asset, який завантажує зображення з локальних ресурсів проєкту [13]. Основні елементи меню створюються за допомогою допоміжного методу `_btn`, що дозволяє уникнути дублювання коду та забезпечити єдиний стиль оформлення інтерфейсу. Під час виклику даного методу задаються іконка, текст кнопки та маршрут переходу, на основі яких формується відповідний пункт меню.

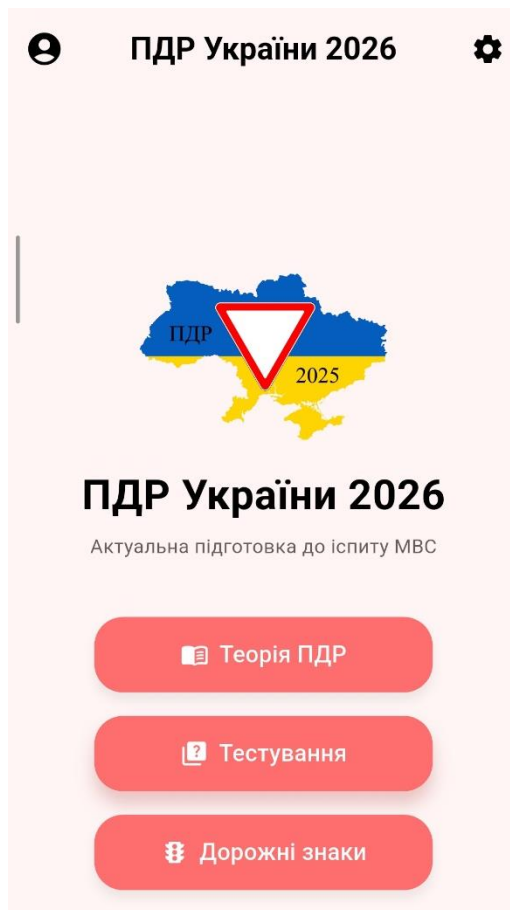


Рисунок 3.1 – Головний екран мобільного застосунку

Після реалізації головного модуля було розроблено користувацький інтерфейс застосунку, який забезпечує доступ до його основних функціональних можливостей. Під час проєктування інтерфейсу особлива увага приділялася зручності навігації, зрозумілому розташуванню елементів керування та забезпеченню єдиного стилю оформлення на всіх екранах в мобільному додатку.

Одним із основних екранів застосунку є екран теоретичних матеріалів, призначений для вивчення правил дорожнього руху, де реалізовано відображення переліку тематичних розділів за допомогою віджетів Card та ListTile (рис. 3.2).

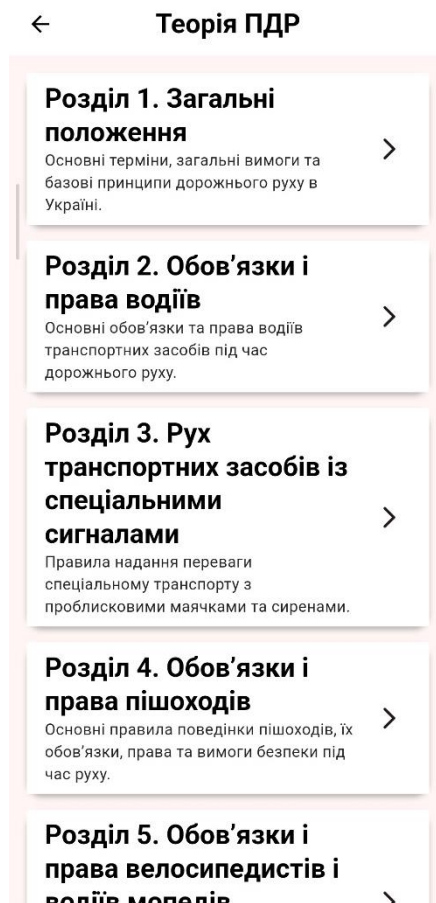


Рисунок 3.2 – Екран розділів теорії ПДР

Вибравши потрібний розділ користувач переходить до екрана SectionDetailsScreen, який призначений для перегляду навчального матеріалу. У ньому інформація подається у вигляді структурованих блоків, що містять номер пункту, назву, текстове пояснення та, за наявності, ілюстрацію. Такий спосіб подання матеріалу сприяє кращому сприйняттю інформації та підвищує зручність вивчення правил дорожнього руху. Загальний вигляд екрана перегляду навчального матеріалу наведено на рисунку 3.3.

← Розділ 1. Загальні полож...

Основні терміни, загальні вимоги та базові принципи дорожнього руху в Україні.

1.1. Поняття Правил дорожнього руху

Правила дорожнього руху встановлюють єдиний порядок організації руху на дорогах України і є обов'язковими для всіх учасників руху.

1.2. Учасники дорожнього руху

До учасників дорожнього руху належать водії, пішоходи, пасажери, велосипедисти, погоничі тварин, а також інші особи, які беруть участь у процесі руху.

1.3. Правосторонній рух

В Україні діє правосторонній рух: усі транспортні засоби повинні рухатися правою стороною проїзної частини.

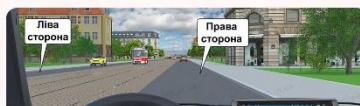


Рисунок 3.3 – Екран перегляду навчального матеріалу

Для забезпечення єдиного візуального стилю в додатку використовуються централізовані налаштування теми, у яких визначаються кольори, вигляд карток, кнопок, полів введення та інших елементів інтерфейсу, що дозволяє використовувати однакове оформлення на всіх екранах в застосунку. Окрім цього, реалізовано підтримку світлої та темної тем, що надає користувачу можливість налаштовувати зовнішній вигляд додатку відповідно до власних уподобань.

Після реалізації екранів застосунку було розроблено модуль тестування, яка є одним із ключових функціональних модулів у мобільному додатку для вивчення правил дорожнього руху. Даний модуль забезпечує проходження тестових питань, перевірку відповідей користувача, оновлення статистики, нарахування досвіду та формування результату після завершення тестування. У застосунку реалізовано два режими роботи: тренувальний та екзаменаційний, вибір яких здійснюється через окремий екран, зображений на рисунку 3.4.

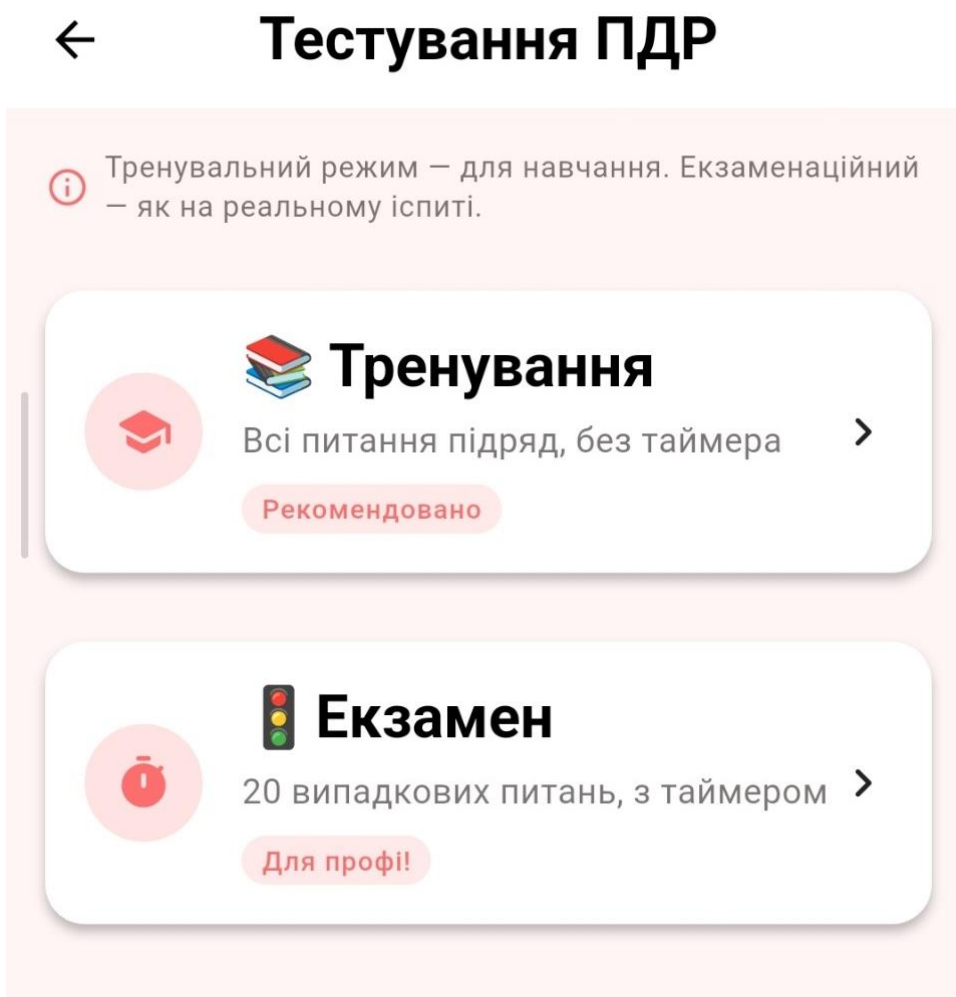


Рисунок 3.4 – Екран вибору режиму тестування

Тренувальний режим призначений для поступового вивчення правил дорожнього руху без обмежень у часі. У цьому режимі користувач може проходити всі доступні питання, отримувати пояснення після помилок та взаємодіяти з персональним помічником. Екзаменаційний режим реалізовано за принципом офіційного теоретичного іспиту: користувачу пропонується 20 випадкових питань, а час проходження обмежено 20 хвилинами.

Запуск обраного режиму тестування здійснюється шляхом переходу до екрана `TestRunnerScreen`, якому передаються параметри, що визначають особливості проходження тесту. Залежно від вибраного режиму налаштовуються умови тестування, набір запитань та додаткові параметри виконання (ліст. 3.4).

Лістинг 3.4 – Запуск режимів тестування

```
Navigator.push(  
  context,  
  MaterialPageRoute(  
    builder: (_) => TestRunnerScreen(  
      title: 'Тренувальний режим',  
      questions: randomized,  
      withTimer: false,  
      trainingMode: true,  
    ),  
  ),  
);  
  
Navigator.push(  
  context,  
  MaterialPageRoute(  
    builder: (_) => TestRunnerScreen(  
      title: 'Екзаменаційний режим',  
      questions: randomized.take(20).toList(),  
      withTimer: true,  
      trainingMode: false,  
      timeLimitSeconds: 1200,  
    ),  
  ),  
);
```

кінець лістингу 3.4

У наведеному фрагменті коду реалізовано передачу параметрів до екрана проходження тестування. Параметр `withTimer` визначає, чи використовується обмеження часу, а `trainingMode` вказує на тип режиму проходження. Значення `timeLimitSeconds` задається в секундах і відповідає 20 хвилинам, що дозволяє наблизити екзаменаційний режим до умов реального теоретичного іспиту.

Перед початком тестування усі питання та варіанти відповідей формуються у випадковому порядку, що дозволяє уникнути механічного запам'ятовування порядку відповідей і тим самим сприяє кращому засвоєнню навчального матеріалу. Для цього реалізовано допоміжні методи (ліст. 3.5), які забезпечують рандомізацію питань і варіантів відповідей зі збереженням правильного індексу після зміни їх порядку.

Лістинг 3.5 – Методи рандомізації питань і варіантів відповідей

```

static List<TestQuestionModel> randomizeQuestions (
    List<TestQuestionModel> list,
) {
    final items = List<TestQuestionModel>.from(list);
    items.shuffle(Random());
    return items;
}

static TestQuestionModel randomizeAnswers(TestQuestionModel q) {
    final answers = List<String>.from(q.answers);
    final correctAnswer = answers[q.correctIndex];

    answers.shuffle(Random());
    final newIndex = answers.indexOf(correctAnswer);

    return TestQuestionModel(
        id: q.id,
        sectionId: q.sectionId,
        question: q.question,
        answers: answers,
        correctIndex: newIndex,
        imagePath: q.imagePath,
        explanation: q.explanation,
    );
}

```

кінець лістингу 3.5

Метод `randomizeQuestions` забезпечує випадкове впорядкування списку питань, тоді як метод `randomizeAnswers` змінює порядок варіантів відповідей із подальшим визначенням нового індексу правильної відповіді, що дозволяє коректно виконувати перевірку результатів тестування після рандомізації даних.

Після формування випадкового набору запитань додаток виконує перевірку відповідей користувача та фіксує результати проходження тестування (ліст. 3.6).

Лістинг 3.6 – Перевірка відповіді та оновлення статистики користувача

```

Future<void> _registerAnswer (
    bool isCorrect,
    TestQuestionModel q,
) async {
    final profile = await UserProfileManager.loadProfile();

    if (isCorrect) {
        correctAnswers++;
    }
}

```

```

    profile.correctAnswers++;
    profile.addXP(5);
    wrongStreak = 0;
  } else {
    wrongStreak++;
    profile.wrongAnswers++;
  }

  await UserProfileManager.saveProfile(profile);

  if (widget.trainingMode && airaEnabled) {
    setState(() => showAira = true);
    airaController.forward(from: 0);
  }
}

```

кінець лістингу 3.6

Даний лістинг демонструє реалізований механізм нарахування досвіду, оновлення статистики та збереження отриманих результатів. У разі правильної відповіді збільшується кількість правильних відповідей і нараховується досвід, а при помилці фіксується збільшення кількості неправильних відповідей, після чого вся оновлена інформація зберігається у локальному сховищі пристрою.

Під час проходження тренувального тестування на екрані відображаються текст запитання, варіанти відповідей та індикатор прогресу, що показує поточний етап виконання тесту. Загальний вигляд екрана наведено на рисунку 3.5.

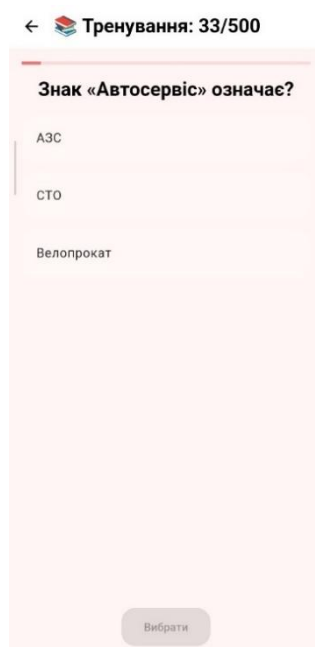


Рисунок 3.5 – Екран проходження тренувального тестування

Після вибору одного із варіантів відповіді система відображає результат перевірки за допомогою кольорового виділення, що дозволяє користувачу одразу оцінити правильність свого вибору. Приклад відображення правильної та неправильної відповіді наведено на рисунку 3.6.

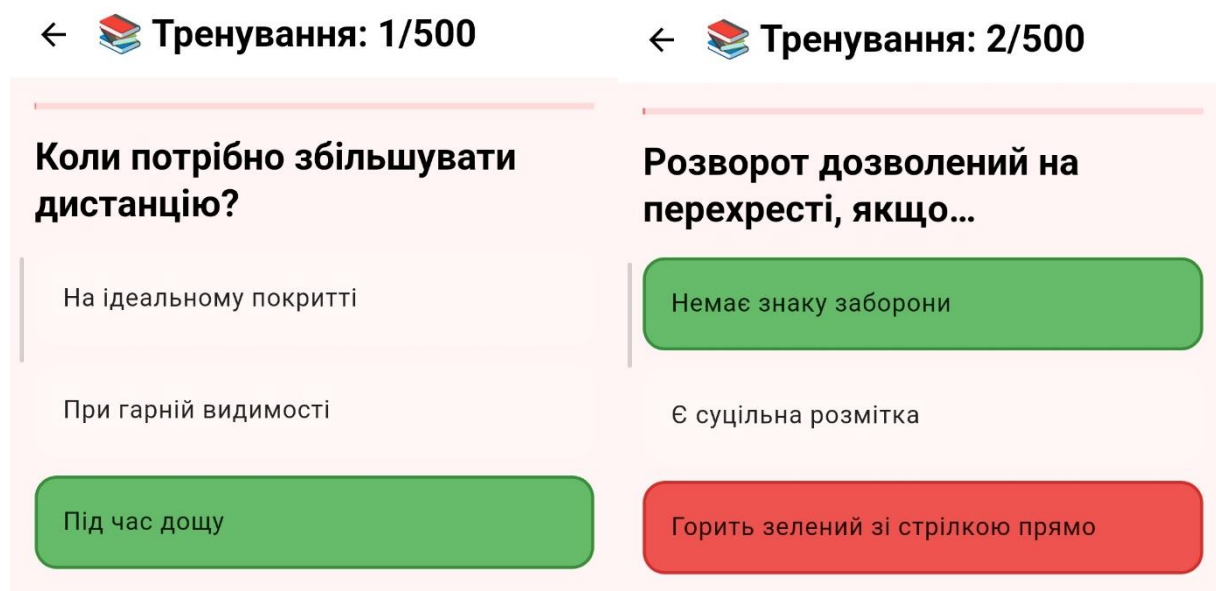


Рисунок 3.6 – Відображення правильної та неправильної відповіді користувача

Для контролю тривалості проходження екзаменаційного режиму в застосунку реалізовано таймер, фрагмент коду якого наведено в лістингу 3.7. Якщо тестування запускається з обмеженням часу, створюється періодичний таймер, який щосекунди зменшує значення часу, що залишився до завершення тестування. Після закінчення відведеного часу викликається метод `_finishTest()`, який завершує тестування та формує підсумковий результат.

Лістинг 3.7 – Реалізація таймера для екзаменаційного режиму

```
timeLeft = widget.timeLimitSeconds ?? 0;

if (widget.withTimer && widget.timeLimitSeconds != null) {
  timer = Timer.periodic(
    const Duration(seconds: 1),
    (_) {
      if (timeLeft <= 0) {
        _finishTest();
      } else {
```

```

        setState(() => timeLeft--);
    }
},
);
}

```

кінець лістингу 3.7

Загальний вигляд екрану екзаменаційного режиму з відображенням таймера наведено на рисунку 3.7.

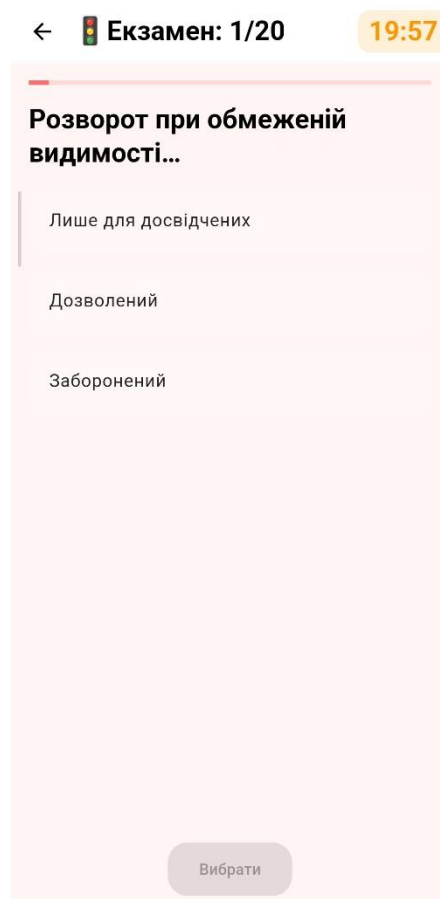


Рисунок 3.7 – Екран екзаменаційного режиму із таймером

Після завершення тестування система зберігає результат проходження, оновлює кількість пройдених тестів та виконує перевірку досягнень користувача. Підсумковий результат відображається на окремому екрані у вигляді відсоткового показника успішності та візуального індикатора результату, зображеного на рисунку 3.8.

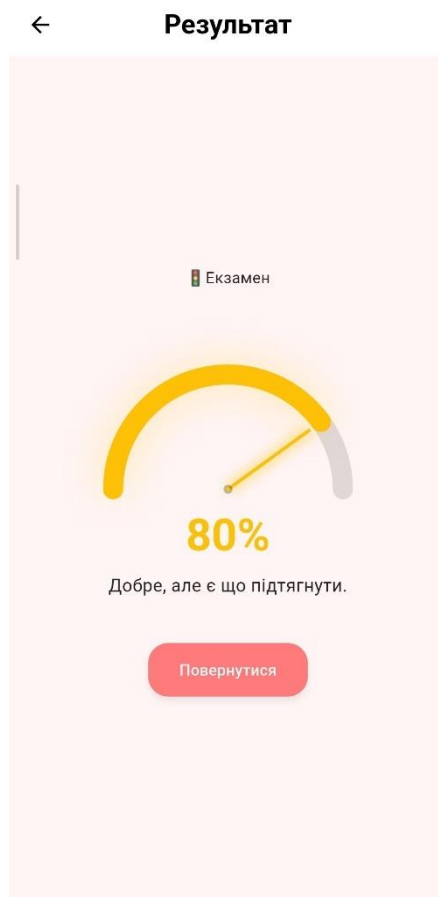


Рисунок 3.8 – Екран підсумкового результату тестування

Наступним етапом практичної реалізації застосунку стала розробка персонального помічника Айри, яка використовується в тренувальному режимі та призначена для підвищення інтерактивності навчального процесу й підтримки мотивації користувача під час проходження тестування. На відміну від екзаменаційного режиму, умови якого максимально наближені до реального іспиту, у тренувальному режимі користувач отримує додаткові пояснення, підказки та реакції помічника після вибору відповіді.

Логіка роботи Айри залежить від результату відповіді користувача. Якщо відповідь правильна, помічник відображає мотиваційне повідомлення та використовує відповідне зображення. Якщо відповідь неправильна – користувачу відображається пояснення помилки. У разі кількох неправильних відповідей поспіль Айра переходить у суворіший режим реакції, що додатково привертає увагу користувача до необхідності уважнішого проходження тесту. Реалізацію механізму вибору реакції персонального помічника наведено в лістингу 3.8.

Лістинг 3.8 – Вибір реакції персонального помічника

```

if (isCorrect) {
    airaImage = "assets/images/aira_happy.gif";
    airaText = _random([
        "Молодець ❤️",
        "Так тримати ☆",
        "Ти на правильному шляху 🍀",
    ]);
} else if (wrongStreak >= 5) {
    airaImage = "assets/images/aira_angry.gif";
    airaText = "${q.explanation}\n\nАйра вже сердиться...";
} else {
    airaImage = "assets/images/aira_sad.gif";
    airaText = q.explanation ?? "Помилка.";
}

```

кінець лістингу 3.8

Наведений фрагмент демонструє реалізацію механізму вибору реакції персонального помічника залежно від правильності відповіді користувача та серії помилок. Для відображення різних станів використовуються окремі графічні зображення та текстові повідомлення, приклади яких наведено на рисунку 3.9. Автоматична зміна реакцій після перевірки відповіді робить взаємодію із застосунком більш динамічною, забезпечує зворотний зв'язок і сприяє підвищенню зацікавленості користувача під час навчання.



Рисунок 3.9 – Приклади реакцій персонального помічника Айри

Для відображення персонального помічника на екрані тестування використовується окремий блок інтерфейсу із зображенням Айри та текстовим

повідомленням, який з'являється за допомогою анімації, забезпечуючи плавну взаємодію з користувачем. У застосунку також реалізовано можливість керування відображенням помічника, реалізацію якої наведено в лістингу 3.9. Користувач може увімкнути або вимкнути Айру в налаштуваннях додатку, а обраний параметр автоматично відновлюється після повторного запуску застосунку.

Лістинг 3.9 – Налаштування відображення помічника

```
class SettingsManager {
    static const _airaKey = 'show_aira';

    static Future<bool> isAiraEnabled() async {
        final prefs = await SharedPreferences.getInstance();

        return prefs.getBool(_airaKey) ?? true;
    }

    static Future<void> setAiraEnabled(bool value) async {
        final prefs = await SharedPreferences.getInstance();

        await prefs.setBool(_airaKey, value);
    }
}
```

кінець лістингу 3.9

Наведений фрагмент коду демонструє реалізацію збереження параметру відображення Айри за допомогою `SharedPreferences`. Метод `isAiraEnabled()` зчитує збережене значення параметра, а `setAiraEnabled()` оновлює його. Якщо налаштування відсутнє, помічник за замовчуванням вважається увімкненим.

Окрім налаштувань персонального помічника, у застосунку локально зберігаються дані профілю користувача та результати проходження тестування. Оскільки застосунок працює в офлайн-режимі, інформація про профіль користувача, статистику проходження тестів і налаштування зберігається безпосередньо на пристрої користувача за допомогою `Hive`, що забезпечує швидкий доступ до потрібних даних під час роботи застосунку (ліст. 3.10).

Лістинг 3.10 – Завантаження та збереження профілю користувача

```

static Future<UserProfile> loadProfile() async {
    final box = await Hive.openBox<UserProfile>(boxName);

    if (box.isEmpty) {
        final profile = UserProfile(username: "Користувач");
        await box.put(0, profile);
        return profile;
    }

    return box.get(0)!;
}

static Future<void> saveProfile(UserProfile profile) async {
    final box = await Hive.openBox<UserProfile>(boxName);
    await box.put(0, profile);
}

```

кінець лістингу 3.10

У наведеному фрагменті коду метод `loadProfile()` перевіряє наявність збережених даних і за потреби створює початковий профіль із базовим іменем користувача, а `saveProfile()` використовується для збереження оновлених даних після зміни статистики, досвіду або інших параметрів користувача. Дані профілю відображаються на окремому екрані, наведеному на рисунку 3.10.



Рисунок 3.10 – Екран профілю користувача

Таким чином, у результаті практичної реалізації було створено мобільний застосунок для вивчення правил дорожнього руху. Розроблені програмні модулі забезпечують вивчення навчального матеріалу, проходження тестування, збереження результатів та персоналізацію процесу навчання.

3.2 Тестування та налагодження інформаційно-комп'ютерної системи

Тестування мобільного застосунку здійснювалося протягом усього процесу розробки та було спрямоване на перевірку коректності й стабільності роботи програмного забезпечення. Основною метою тестування було виявлення можливих помилок, перевірка правильності функціонування окремих модулів застосунку, забезпечення коректної взаємодії між ними та оцінка зручності використання користувацького інтерфейсу.

Для перевірки працездатності застосунку використовувалося ручне функціональне тестування, яке виконувалося як за допомогою емулятора Android Studio, так і на реальних Android-пристроях різних конфігурацій, що дозволило оцінити поведінку застосунку в різних умовах використання та виявити недоліки, які не завжди проявляються під час роботи в середовищі емуляції.

У процесі тестування перевірялася робота головного меню застосунку, модулів перегляду теоретичних матеріалів та дорожніх знаків, тренувального й екзаменаційного режимів тестування, системи досягнень, персонального помічника, а також збереження статистики та налаштувань користувача.

Під час перевірки екзаменаційного режиму було виявлено помилку, пов'язану з відображенням елементів інтерфейсу. Заголовок екрана містив повну назву режиму «Екзаменаційний режим», яка займала значну частину верхньої панелі застосунку. На пристроях із меншою шириною екрана це призводило до зміщення таймера за межі екрана, через що користувач не міг бачити залишок часу для проходження тестування. Для усунення даного недоліку було змінено текст заголовка на скорочений варіант «Екзамен», що дозволило звільнити необхідний простір для коректного відображення таймера незалежно від розміру

екрана пристрою. Після внесення змін інтерфейс екзаменаційного режиму почав коректно відображатися на всіх протестованих пристроях. Результат виправлення, а також вигляд екрана до внесення змін наведено на рисунку 3.11.

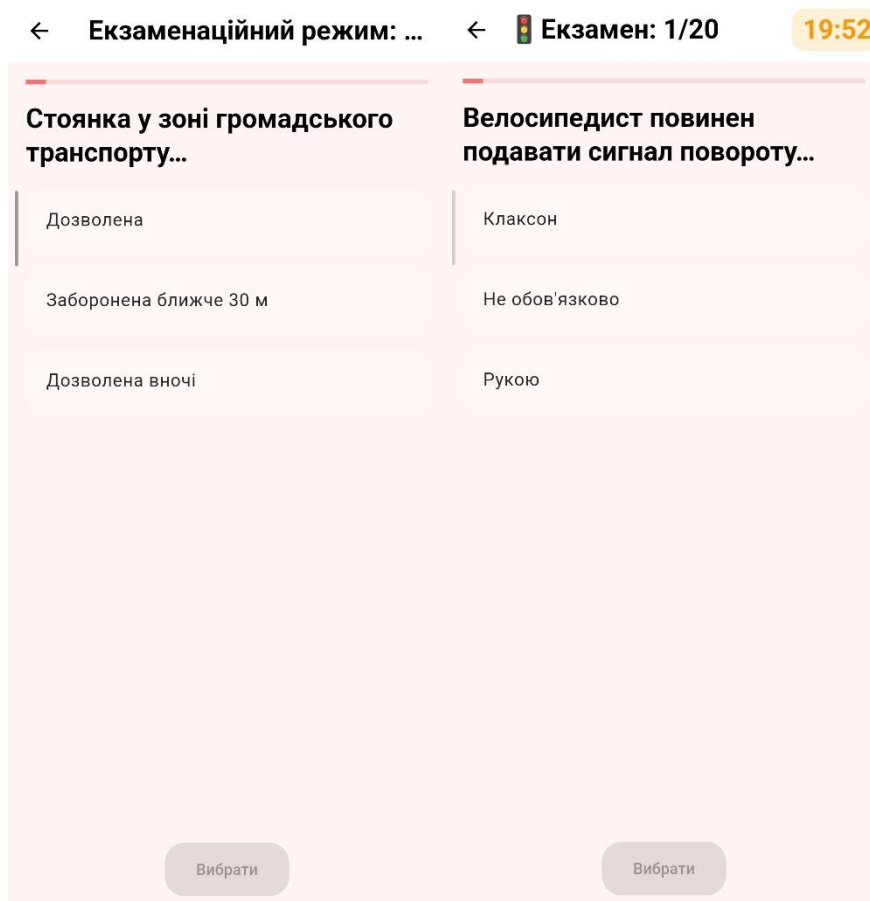


Рисунок 3.11 – Екран екзаменаційного режиму до та після внесення змін

Під час перевірки екрана відображення результатів тестування було виявлено недолік, пов'язаний із некоректною роботою механізму формування результату та відображення візуального індикатора. Через помилки в логіці формування результату та налаштуванні параметрів відображення візуального індикатора виникали проблеми з коректним відображенням кольорів і текстових елементів. У деяких випадках текст накладався один на одного, а також виникало повідомлення про переповнення області відображення інтерфейсу, що негативно впливало на зручність використання застосунку. Для виправлення цього недоліку було змінено алгоритм формування результату тестування та внесено зміни до параметрів відображення елементів інтерфейсу. Результат виправлення, а також

вигляд до внесення змін наведено на рисунку 3.12. Після налагодження всі текстові елементи почали відображатися коректно, а колір індикатора результату став відповідати фактичному відсотку правильних відповідей користувача.

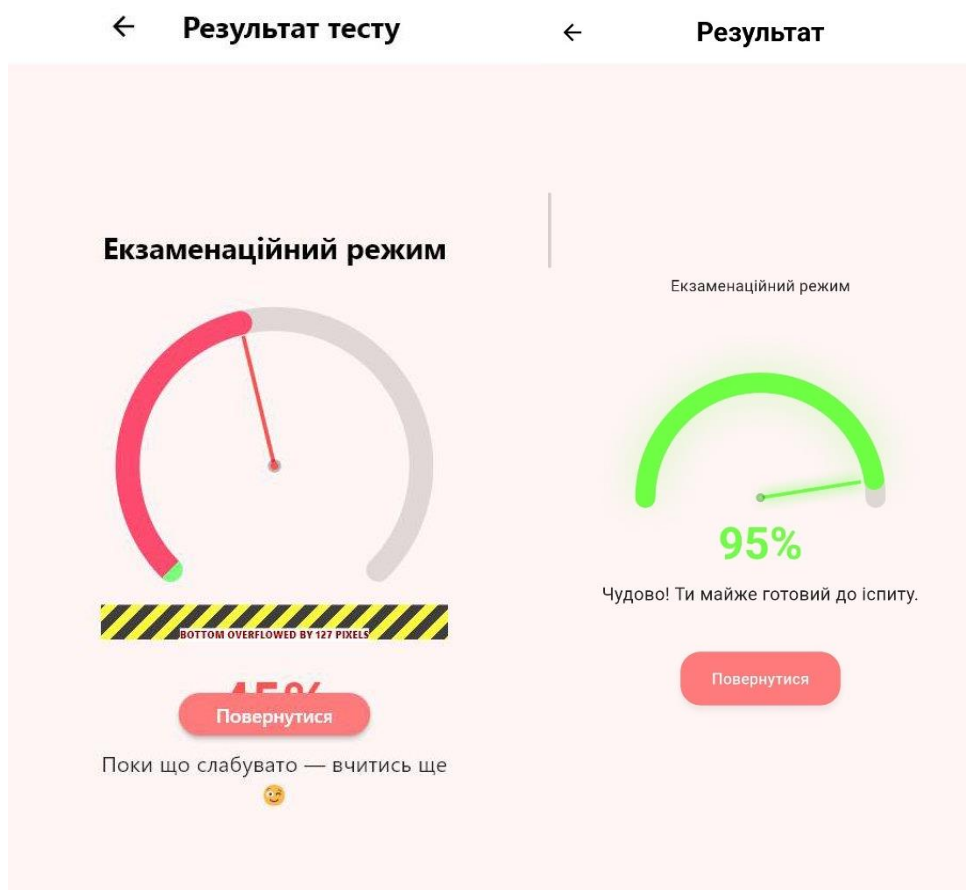


Рисунок 3.12 – Екран результатів тестування до та після внесення змін

Таким чином, у результаті проведеного тестування підтверджено коректність роботи основних функціональних модулів мобільного застосунку та стабільність його функціонування на різних пристроях, а отримані результати свідчать про готовність програмного продукту до практичного використання як засобу підготовки до теоретичного іспиту з правил дорожнього руху.

3.3 Розробка локальної бази даних застосунку

Оскільки розроблений мобільний застосунок працює в офлайн-режимі та не використовує серверну частину, для збереження й обробки даних у ньому

реалізовано локальну структуру зберігання інформації. Дані про профіль користувача, статистику проходження тестів, налаштування, теоретичні матеріали та дорожні знаки зберігаються на пристрої користувача, що забезпечує доступ до основних функцій застосунку без підключення до мережі Інтернет.

Локальне зберігання даних у застосунку реалізовано за допомогою бібліотек: Hive та SharedPreferences. Hive використовується для роботи з профілем користувача, досвідом, рівнем, статистикою відповідей і досягненнями, а SharedPreferences – для зберігання простих параметрів, таких як ім'я користувача, вибрана тема інтерфейсу та налаштування відображення персонального помічника Айри. З метою оптимізації застосунку використано JSON-файли для зберігання навчального контенту, зокрема тестових питань, теоретичних матеріалів та дорожніх знаків. Загальну структуру локального зберігання даних наведено на рисунку 3.13.

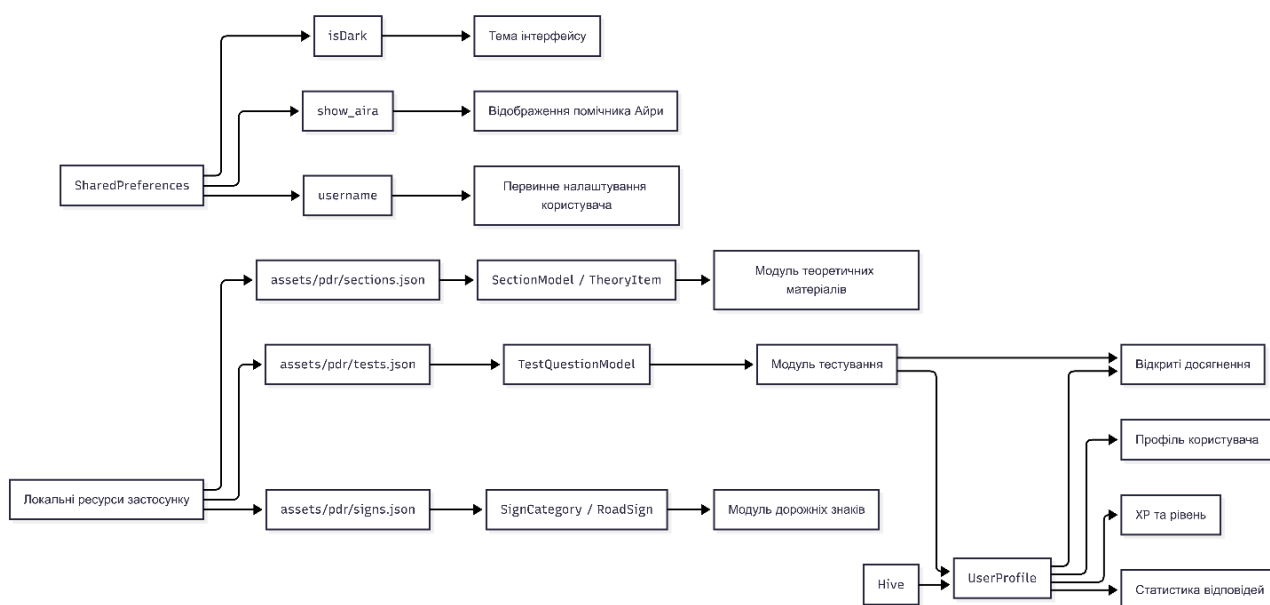


Рисунок 3.13 – Структура локального збереження даних застосунку

Однією з основних моделей застосунку є модель профілю користувача, реалізована з використанням Hive (ліст. 3.11). Вона містить інформацію про ім'я користувача, рівень, статистику проходження тестів та відкриті досягнення.

Лістинг 3.11 – Модель профілю користувача для Hive

```
@HiveType(typeId: 10)
class UserProfile extends HiveObject {
    @HiveField(0)
    String username;

    @HiveField(1)
    int xp;

    @HiveField(2)
    int level;

    @HiveField(3)
    int testsPassed;

    @HiveField(4)
    int correctAnswers;

    @HiveField(5)
    int wrongAnswers;

    @HiveField(6)
    List<String> unlockedAchievements;

    void addXP(int amount) {
        xp += amount;
        recalcLevel();
    }
}
```

кінець лістингу 3.11

У наведеному фрагменті коду анотації `@HiveType` [14] та `@HiveField` [15] використовуються для налаштування збереження об'єкта і його полів у локальному сховищі Hive, а метод `addXP()` реалізує механізм нарахування досвіду та автоматичного перерахунку рівня після його зміни.

Для представлення тестових питань у застосунку використовується модель `TestQuestionModel`, яка описує структуру одного питання: ідентифікатор, розділ, текст питання, список відповідей, індекс правильної відповіді, шлях до зображення та пояснення. Саме ця модель використовується під час формування застосунком тренувального або екзаменаційного режиму тестування (ліст. 3.12).

Лістинг 3.12 – Модель тестового питання

```

class TestQuestionModel {
  final String id;
  final String sectionId;
  final String question;
  final List<String> answers;
  final int correctIndex;
  final String? imagePath;
  final String? explanation;

  factory TestQuestionModel.fromJson(Map<String, dynamic> json) {
    return TestQuestionModel(
      id: json["id"].toString(),
      sectionId: json["sectionId"].toString(),
      question: json["question"],
      answers: List<String>.from(json["answers"]),
      correctIndex: json["correctIndex"],
      imagePath: json["imagePath"],
      explanation: json["explanation"],
    );
  }
}

```

кінець лістингу 3.12

У даному лістингу показано використання фабричного конструктора `fromJson`, який перетворює дані з JSON-файлу у Dart-об'єкт, що дозволяє зручно працювати з тестовими питаннями у коді застосунку, не звертаючись безпосередньо до структури JSON під час кожної операції. Поле `correctIndex` використовується для перевірки відповіді користувача, а поле `explanation` – для відображення пояснення після помилки у тренувальному режимі.

Завантаження та обробка тестових питань із локального JSON-файлу реалізовано за допомогою класу `PdrTestsLoader`, який наведено в лістингу 3.13.

Лістинг 3.13 – Завантаження тестових питань із JSON-файлу

```

class PdrTestsLoader {
  static const String _testsPath = 'assets/pdr/tests.json';

  static Future<List<TestQuestionModel>> loadAll() async {
    final raw = await rootBundle.loadString(_testsPath);
    final List list = json.decode(raw) as List;

    return list
      .map((e) => TestQuestionModel.fromJson(e))
  }
}

```

```

        .toList();
    }

    static Future<List<TestQuestionModel>> loadBySection(
        String sectionId,
    ) async {
        final all = await loadAll();
        return all.where((e) => e.sectionId == sectionId).toList();
    }
}

```

кінець лістингу 3.13

Метод `loadAll()` завантажує всі тестові питання з локального JSON-файлу, після чого кожен елемент перетворюється у модель `TestQuestionModel`, а метод `loadBySection()` використовується для фільтрації питань за конкретним розділом правил дорожнього руху. Такий підхід дозволяє використовувати одну локальну базу тестових питань для різних режимів роботи застосунку.

Окремо в застосунку реалізовано завантаження теоретичних розділів правил дорожнього руху. Для обробки теоретичних матеріалів використовуються моделі `SectionModel` та `TheoryItem`, а їх завантаження здійснюється за допомогою класу `PdrLoader`, фрагмент якого наведено в лістингу 3.14.

Лістинг 3.14 – Завантаження теоретичних розділів ПДР

```

static Future<List<SectionModel>> loadSections() async {
    final jsonString =
        await rootBundle.loadString('assets/pdr/sections.json');

    final List data = json.decode(jsonString);

    return data.map((sectionData) {
        final theoryList = (sectionData['theory'] as List).map((t) {
            return TheoryItem(
                number: t['number'],
                title: t['title'],
                text: t['text'],
                imagePath: t['imagePath'],
            );
        }).toList();

        return SectionModel(
            id: sectionData['id'],
            title: sectionData['title'],
            description: sectionData['description'],

```

```

        theory: theoryList,
    );
  }).toList();
}

```

кінець лістингу 3.14

Наведений лістинг демонструє процес читання теоретичних матеріалів із локального JSON-файлу. Кожен розділ містить заголовок, опис та список навчальних пунктів, які перетворюються у відповідні об'єкти `TheoryItem`. Після цього формується список об'єктів `SectionModel`, який використовується екраном теоретичних матеріалів для відображення навчального контенту користувачу. Крім тестових питань і теоретичних матеріалів, у локальних ресурсах застосунку зберігаються дані дорожніх знаків, для представлення яких використовуються моделі `SignCategory` та `RoadSign`, що описують категорії знаків, назви, зображення та текстові пояснення.

3.4 Розробка документації для програмного продукту

Документація є важливою складовою програмного продукту, оскільки вона забезпечує користувачів необхідною інформацією щодо встановлення, налаштування та використання застосунку. Наявність зрозумілої документації спрощує процес початку роботи із застосунком, підвищує зручність його використання та сприяє ефективній експлуатації програмного забезпечення.

Розроблений мобільний застосунок призначений для самостійного вивчення правил дорожнього руху, підготовки до теоретичного іспиту та перевірки знань користувачів у тренувальному й екзаменаційному режимах. Програмний продукт орієнтований на широке коло користувачів та не потребує спеціальних навичок для початку роботи. Однією з ключових особливостей додатку є можливість повноцінної роботи без підключення до мережі Інтернет, оскільки всі необхідні дані зберігаються локально на пристрої користувача.

Для забезпечення стабільної роботи застосунку пристрій користувача повинен відповідати мінімальним системним вимогам, наведеним у таблиці 3.1.

Таблиця 3.1 – Мінімальні та рекомендовані системні вимоги до пристрою для стабільної роботи мобільного застосунку

Параметр	Мінімальні вимоги	Рекомендовані вимоги
Операційна система	Android 8.0	Android 10 і вище
Оперативна пам'ять	2 ГБ	4 ГБ і більше
Вільне місце на диску	100 МБ	200 МБ і більше
Процесор	4-ядерний	8-ядерний
Роздільна здатність екрана	720×1280	Full HD і вище
Підключення до інтернету	Не обов'язкове	Не обов'язкове

Під час першого запуску застосунку відображається екран початкового налаштування, наведений на рисунку 3.14. На цьому екрані необхідно ввести власне ім'я, яке надалі використовується для персоналізації інтерфейсу та відображення інформації про користувача в профілі застосунку.

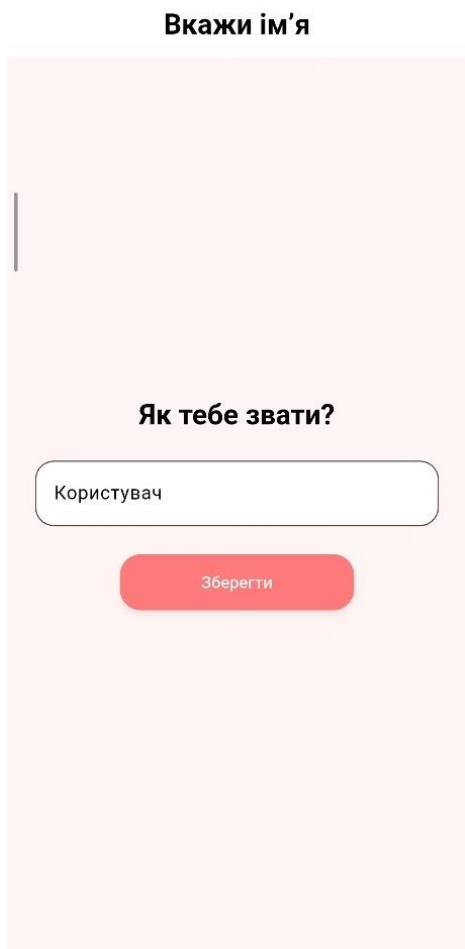


Рисунок 3.14 – Екран першого запуску застосунку

Після завершення початкового налаштування відкривається головне меню застосунку, з якого можна перейти до розділів теоретичних матеріалів, тестування, дорожніх знаків і профілю, який містить статистику та досягнення користувача. Основний сценарій використання передбачає ознайомлення з теоретичними матеріалами, перегляд дорожніх знаків і проходження тестів для перевірки отриманих знань. Під час тестування система автоматично зберігає результати, оновлює рівень досвіду та відстежує прогрес навчання.

Для забезпечення зручності використання застосунків містить окремий екран налаштувань, зображений на рисунку 3.15, у якому можна персоналізувати роботу системи відповідно до власних потреб.

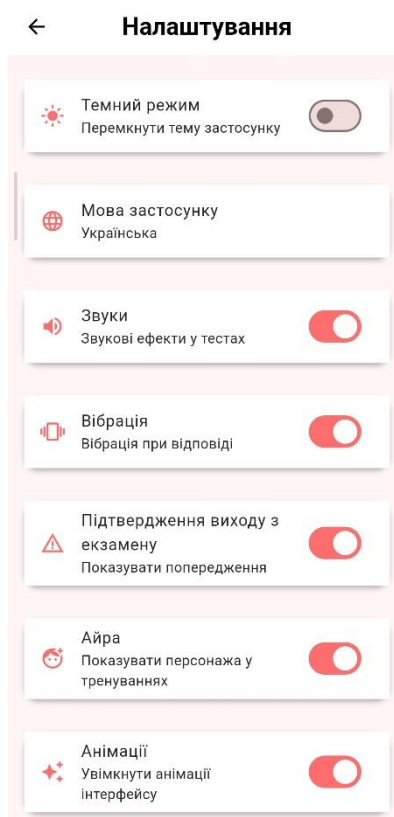


Рисунок 3.15 – Екран налаштувань застосунку

У налаштуваннях можна змінювати тему оформлення застосунку, обираючи світлий або темний режим інтерфейсу. Також передбачено можливість увімкнення або вимкнення персонального помічника Айри, який супроводжує користувача під час проходження тренувального режиму тестування. Додатково

доступне керування звуковими ефектами та вібрацією, що дозволяє адаптувати роботу застосунку до власних уподобань і умов використання.

Встановлення програмного продукту може здійснюватися двома способами: через магазин Google Play або за допомогою APK-файлу. Основним способом поширення застосунку є використання магазину Google Play, що забезпечує зручне встановлення та автоматичне отримання оновлень. Сторінку застосунку в Google Play наведено на рисунку 3.16.

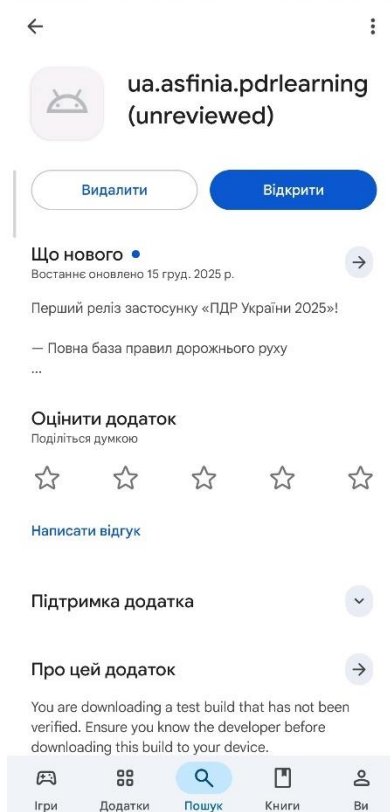


Рисунок 3.16 – Сторінка застосунку в Google Play

Для встановлення застосунку через Google Play необхідно відкрити сторінку програмного продукту в магазині, натиснути кнопку «Установити» та дочекатися завершення процесу завантаження. Після цього застосунок автоматично з’явиться у списку встановлених програм пристрою.

Альтернативним способом встановлення є використання APK-файлу. Для цього потрібно завантажити інсталяційний файл на пристрій, відкрити його за допомогою файлового менеджера та підтвердити встановлення. У разі

необхідності система Android може запропонувати надати дозвіл на встановлення застосунків із зовнішніх джерел. Після завершення інсталяції застосунок готовий до використання та початкового налаштування.

Отже, розроблена документація містить основні відомості щодо використання програмного продукту, системних вимог та процесу встановлення застосунку. Наведені інструкції дозволяють швидко розпочати роботу із системою та ефективно використовувати її можливості для вивчення правил дорожнього руху й підготовки до теоретичного іспиту.

3.5 Розробка інсталяційного пакета

Після завершення реалізації та тестування мобільного застосунку було підготовлено інсталяційну версію програмного продукту для його встановлення на пристроях користувачів. Для Android-платформи інсталяційний пакет сформований у вигляді APK та AAB файлів. APK-файл використовується для прямого встановлення застосунку на пристрій [16], а AAB-файл для публікації програмного продукту в Google Play [17].

Інсталяційний пакет розробленого мобільного застосунку містить скомпільований програмний код, Android-обгортку Flutter-проекту, ресурси інтерфейсу, іконку застосунку, зображення персонального помічника, графічні матеріали досягнень, JSON-файли з навчальними матеріалами та зображення дорожніх знаків. Підключення необхідних залежностей та ресурсів виконується у файлі `pubspec.yaml`, фрагмент якого наведено в лістингу 3.15.

Лістинг 3.15 – Підключення залежностей та ресурсів застосунку

```
dependencies:  
  flutter:  
    sdk: flutter  
  
  sqflite: ^2.3.2  
  shared_preferences: ^2.2.2  
  intl: ^0.19.0  
  hive: ^2.2.3  
  hive_flutter: ^1.1.0
```

```
flutter:
  uses-material-design: true

  assets:
    - assets/images/
    - assets/pdr/
    - assets/pdr/signs/
```

кінець лістингу 3.15

У наведеному лістингу показано основні залежності, які використовуються для роботи застосунку, зокрема бібліотеки для локального збереження даних, роботи з налаштуваннями та форматуванням інформації. У секції `assets` зазначено директорії з графічними ресурсами, JSON-файлами навчальних матеріалів та зображеннями дорожніх знаків. Завдяки цьому всі необхідні ресурси входять до складу інсталяційного пакета та доступні для застосунку після встановлення.

Для коректного відображення застосунку в системному меню Android було налаштовано іконку програмного продукту. Для автоматичного створення іконок необхідних розмірів використовується пакет `flutter_launcher_icons`, конфігурацію якого наведено в лістингу 3.16.

Лістинг 3.16 – Налаштування іконки застосунку

```
flutter_launcher_icons:
  android: true
  ios: true
  image_path: "assets/images/app_icon.png"
  min_sdk_android: 21
```

кінець лістингу 3.16

Наведений фрагмент визначає шлях до початкового зображення іконки та параметри її генерації для Android і iOS. Після виконання відповідної команди Flutter автоматично створює набір іконок для різних роздільних здатностей екрана та розміщує їх у директоріях Android-проекту, що дозволяє забезпечити коректне відображення іконки застосунку на різних пристроях.

Основні параметри Android-застосунку задаються у файлі `AndroidManifest.xml`, де визначаються назва застосунку, системна іконка, головна активність, режим запуску та параметри відображення інтерфейсу. У цьому файлі для розробленого застосунку задано назву «ПДР – Навчання», іконку `@mipmap/ic_launcher` та головну активність `MainActivity`, яка запускається під час відкриття додатку користувачем.

Для створення release-збірки Android-застосунку згенеровано ключ підпису, який необхідний для встановлення release-версії та подальшого поширення програмного продукту через магазин Google Play. Генерація ключа здійснюється за допомогою утиліти `keytool` [18], яка входить до складу Android SDK. Фрагмент команди генерації ключа наведено в лістингу 3.17.

Лістинг 3.17 – Генерація ключа для підпису release-збірки

```
set "KEYTOOL=C:\Program Files\Android\Android
Studio\jbr\bin\keytool.exe"

"%KEYTOOL%" -genkey -v ^
    -keystore release-keystore.jks ^
    -keyalg RSA ^
    -keysize 2048 ^
    -validity 10000 ^
    -alias upload
```

кінець лістингу 3.17

У наведеному фрагменті команди задаються основні параметри генерації ключа підпису, зокрема назва файлу сховища, алгоритм RSA, розмір ключа, термін його дії та псевдонім. У результаті успішного виконання створюється файл `release-keystore.jks`, який надалі використовується для підпису release-збірки застосунку.

Після налаштування ресурсів застосунку, іконки та ключа підпису виконується формування інсталяційного пакета. Для створення release-збірки Flutter надає можливість генерувати APK-файл для встановлення застосунку на Android-пристрої або Android App Bundle для подальшої публікації в магазині Google Play. Команди створення інсталяційних пакетів наведено в лістингу 3.18.

Лістинг 3.18 – Команди створення release-збірки застосунку

```
flutter build apk --release
```

```
flutter build appbundle --release
```

кінець лістингу 3.18

Наведені команди використовуються для формування release-збірки застосунку та отримання інсталяційних пакетів, готових до встановлення на Android-пристроях або подальшої публікації в магазині Google Play. У результаті їх виконання створюються файли збірки, зображені на рисунку 3.17, які можуть використовуватися для поширення програмного продукту.

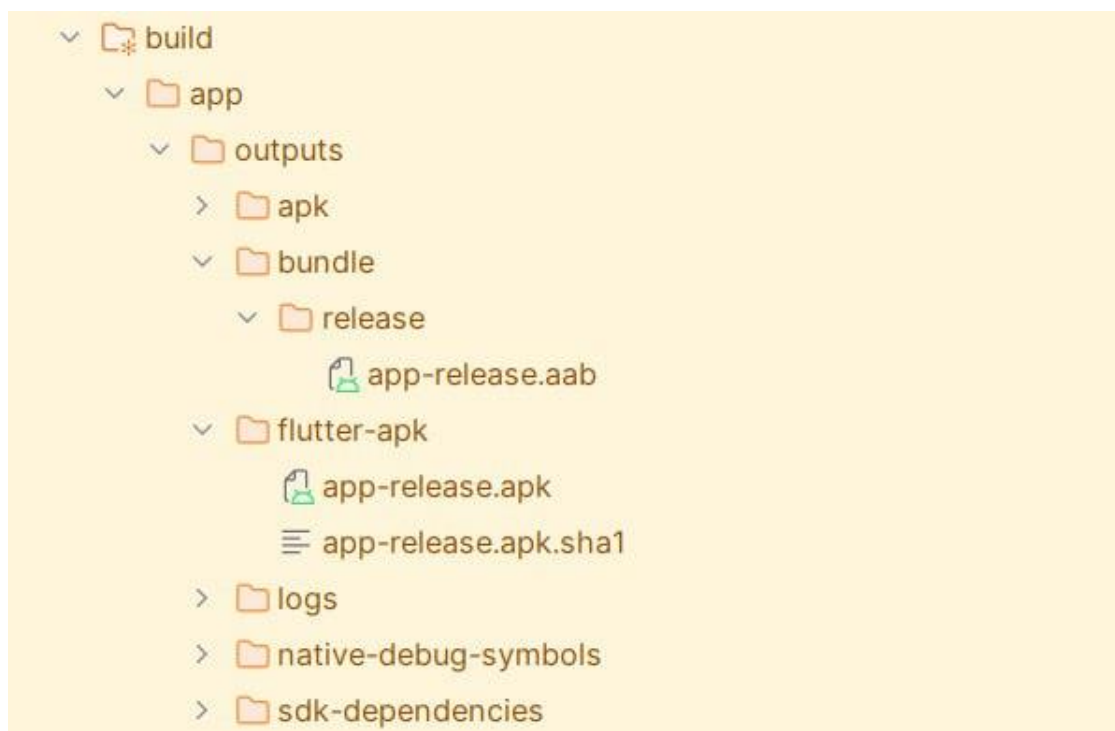


Рисунок 3.17 – Сформований інсталяційний пакет застосунку

Таким чином, у процесі розробки інсталяційного пакета було виконано підключення необхідних ресурсів, налаштування іконки, визначення Android-параметрів застосунку, підготовку ключа підпису та формування release-збірки. Отриманий інсталяційний пакет містить усі необхідні компоненти для автономної роботи застосунку та може бути використаний як для прямого встановлення через APK-файл, так і для поширення через Google Play.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи бакалавра було досягнуто поставленої мети, яка полягала у розробці кросплатформного мобільного застосунку для вивчення правил дорожнього руху з використанням сучасних технологій мобільної розробки та інтерактивних методів навчання.

У межах роботи було проведено аналіз предметної області, сучасних мобільних застосунків для вивчення правил дорожнього руху та наукових досліджень у сфері мобільної розробки і цифрових технологій навчання. Визначено основні переваги та недоліки існуючих програмних рішень, а також обґрунтовано необхідність створення нового програмного продукту.

Виконано проектування програмного забезпечення із застосуванням UML-нотації та розроблено діаграми, які відображають структуру застосунку, взаємодію користувача із додатком та основні алгоритми його функціонування.

Розроблено кросплатформний мобільний застосунок із використанням фреймворку Flutter та мови програмування Dart. Реалізоване рішення забезпечує можливість використання єдиної кодової бази та підтримує роботу на різних мобільних пристроях.

Створено модуль навчання, який забезпечує перегляд теоретичних матеріалів, категорій правил дорожнього руху та дорожніх знаків.

Також реалізовано тренувальний та екзаменаційний режими тестування, де для формування тестових завдань використано алгоритми випадкового вибору запитань і відповідей. Крім цього розроблено механізми перевірки правильності відповідей та обчислення результатів проходження тестування.

У ході роботи було розроблено систему персонального помічника, статистики користувача, нарахування досвіду та досягнень. Для локального збереження даних використано технології Hive та SharedPreferences, що дозволило забезпечити роботу застосунку без підключення до мережі Інтернет.

Крім цього, створено адаптивний користувацький інтерфейс із підтримкою світлої та темної тем оформлення засобами Flutter Material Design, який

забезпечує зручну взаємодію користувача із системою та відповідає сучасним вимогам до мобільних застосунків.

Під час виконання роботи проведено тестування та налагодження програмного продукту, що дозволило виявити та усунути недоліки, пов'язані з відображенням інформації під час проходження тестів і коректністю відображення їх результатів. Також сформовано інсталяційні пакети APK і AAB та забезпечено можливість поширення застосунку через магазин Google Play.

Отримані результати роботи демонструють повне виконання поставлених завдань, а розроблений програмний продукт відповідає сучасним тенденціям розвитку мобільних освітніх систем, має практичне значення та може бути використаний для самостійного вивчення правил дорожнього руху й підготовки до складання теоретичного іспиту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Green-way. Правила дорожнього руху України (ПДР 2026), вчити онлайн. URL: <https://green-way.com.ua/uk> (дата звернення: 09.02.2026).
2. Штрафи UA. Перевірка штрафів ПДР. ПДР. URL: <https://splata.shtrafy.ua/> (дата звернення: 09.02.2026).
3. Jošt G., Taneski V. State-of-the-Art Cross-Platform Mobile Application Development Frameworks: A Comparative Study of Market and Developer Trends. Informatics. 2025. Vol. 12, no. 2. P. 45. URL: <https://doi.org/10.3390/informatics12020045> (дата звернення: 17.02.2026).
4. Diansyah H., Syafrinal S. Design and Development of a Mobile Application Using Android Studio and Flutter. Journal Mobile Technologies (JMS). 2025. Vol. 3, no. 2. P. 69-76. URL: <https://doi.org/10.59431/jms.v3i2.646> (дата звернення: 22.02.2026).
5. Nawaz I., Cuenen A., Wets G., Paul R., Janssens D. Advancing Online Road Safety Education: A Gamified Approach for Secondary School Students in Belgium. Applied Sciences. 2025. Vol. 15, no. 15. P. 8557. URL: <https://doi.org/10.3390/app15158557> (дата звернення: 27.02.2026).
6. React Native. Introduction. Learn once, write anywhere. URL: <https://reactnative.dev/docs/getting-started> (дата звернення: 02.03.2026).
7. Flutter documentation. URL: <https://docs.flutter.dev/> (дата звернення: 02.03.2026).
8. Dart documentation. Dart programming language. URL: <https://dart.dev/docs> (дата звернення: 02.03.2026).
9. Hive – Dart API docs. The official repository for Dart and Flutter packages. URL: <https://pub.dev/documentation/hive/latest/> (дата звернення: 07.03.2026).
10. Shared_preferences. Flutter package. Dart packages. URL: https://pub.dev/packages/shared_preferences (дата звернення: 07.03.2026).
11. Guide to app architecture. Flutter documentation. URL: <https://docs.flutter.dev/app-architecture/guide> (дата звернення: 11.03.2026).

12. EnsureInitialized method – WidgetsFlutterBinding class – widgets library. Flutter – Dart API docs. URL: <https://api.flutter.dev/flutter/widgets/WidgetsFlutterBinding/ensureInitialized.html> (дата звернення: 23.03.2026).

13. Image.asset constructor – Image – widgets library. Flutter – Dart API docs. URL: <https://api.flutter.dev/flutter/widgets/Image/Image.asset.html> (дата звернення: 05.04.2026).

14. HiveType class – hive library – Dart API. The official repository for Dart and Flutter packages. URL: <https://pub.dev/documentation/hive/latest/hive/HiveType-class.html> (дата звернення: 15.04.2026).

15. HiveField class – hive library – Dart API. The official repository for Dart and Flutter packages. URL: <https://pub.dev/documentation/hive/latest/hive/HiveField-class.html> (дата звернення: 15.04.2026).

16. App manifest overview. App architecture. Android Developers. URL: <https://developer.android.com/guide/topics/manifest/manifest-intro#apk> (дата звернення: 26.04.2026).

17. The Android App Bundle format. Other Play guides. Android Developers. URL: <https://developer.android.com/guide/app-bundle/app-bundle-format> (дата звернення: 26.04.2026).

18. The keytool Command. Moved. URL: <https://docs.oracle.com/en/java/javase/25/docs/specs/man/keytool.html> (дата звернення: 02.05.2026).