Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНУ СМАРТФОНІВ З ВИКОРИСТАННЯМ DART/FLUTTER

DEVELOPMENT OF AN ONLINE SMARTPHONE STORE USING DART/FLUTTER

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

Виконав: здобувач вищої освіти
групи ІПЗз-41
Бондаренко Назар Олександрович

Керівник:
Хвищун Микола В'ячеславович

Кваліфікаційну роботу
допущено до захисту
«10» 06 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна
_____

Луцьк – 2025 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» 12 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Бондаренку Назару Олександровичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «*Розробка інтернет-магазину смартфонів з використанням Dart/Flutter*»
Керівник роботи: *к. ф-м.н., доцент Хвищун М. В.*
затверджені наказом закладу вищої освіти від «*19» грудня 2024 р.* № *474/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра *«10» червня 2025 р.*
3. Вихідні дані до роботи: *Dart/Flutter, методичні вказівки до виконання кваліфікаційної роботи бакалавра.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):
Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення, опис функціонального наповнення об'єкта проектування, практична реалізація об'єкта проектування.
5. Перелік графічного матеріалу: *12 рисунків.*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Хвищун М. В.*		
Специфікація вимог до розробленої системи	*Хвищун М. В.*		
Розробка об'єкта проектування	*Хвищун М. В.*		
Нормоконтроль	*Вознюк А.В.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	1,39 %		
Академічна доброчесність	*Хвищун М. В.*		

7. Дата видачі завдання *«20» грудня 2024 р.*

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 04.02.2025 р.	Викон.
2	Аналіз проблеми розробки та впровадження об'єкту проектування	до 01.03.2025 р.	Викон.
3	Обгрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 15.03.2025 р.	Викон.
4	Розробка функціонально-структурної схеми роботи об'єкта проектування	до 29.03.2025 р.	Викон.
5	Практична реалізація об'єкта проектування	до 26.04.2025 р.	Викон.
6	Тестування та налагодження об'єкта проектування	до 03.05.2025 р.	Викон.
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру	до 10.06.2025 р.	Викон.

Здобувач вищої освіти _____ (підпис) Бондаренко Н.О. (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (підпис) Хвищун М.В. (прізвище, ініціали)

АНОТАЦІЯ

Бондаренко Н. О. Розробка інтернет-магазину смартфонів з використанням Dart/Flutter. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

У кваліфікаційній роботі розглянуто процес розробки кросплатформного мобільного додатку інтернет-магазину смартфонів із використанням фреймворку Flutter та мови Dart. У першому розділі здійснено аналіз предметної області, огляд аналогічних рішень і визначено доцільність розробки власного застосунку. У другому розділі сформульовано вимоги до системи, побудовано UML-діаграми, описано функціональні сценарії роботи користувача та здійснено проєктування структури застосунку. Третій розділ присвячено практичній реалізації додатку, опису програмного коду, використаних бібліотек, засобів налагодження, а також підготовці технічної та користувацької документації.

Ключові слова: Flutter, Dart, інтернет-магазин, кросплатформний мобільний додаток, смартфони, UX/UI, мобільна розробка, UML.

ABSTRACT

Bondarenko N. Development of an Online Smartphone Store Using Dart/Flutter. Manuscript. Qualification work of the bachelor's program "Software engineering" in the specialty "Software engineering". Lutsk National Technical University. Lutsk, 2025.

The qualification thesis explores the development process of a cross-platform mobile application for a smartphone online store using the Flutter framework and Dart programming language. The first chapter presents an analysis of the subject domain, a review of similar solutions, and the justification for creating a custom application. The second chapter formulates system requirements, constructs UML diagrams, describes user interaction scenarios, and covers the application' architectural design. The third chapter focuses on the practical implementation of the app, including code structure, used libraries, debugging tools, and the preparation of technical and user documentation.

Keywords: Flutter, Dart, online store, cross-platform mobile application, smartphones, UX/UI, mobile development, UML.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному суспільстві стрімко зростає попит на мобільні технології, що кардинально змінюють підходи до організації торгівлі та взаємодії з клієнтами. В умовах цифрової трансформації бізнесу електронна комерція стала не лише популярним інструментом продажу, а й необхідною умовою конкурентоспроможності компаній. Особливе місце серед каналів електронної комерції посідають мобільні застосунки, які забезпечують доступ до товарів і послуг у будь-який час та з будь-якого місця.

Смартфони стали головним інструментом повсякденного користування, і значна частина онлайн-покупок сьогодні здійснюється саме з мобільних пристроїв. Користувачі очікують від застосунків швидкого завантаження, інтуїтивно зрозумілого інтерфейсу, безперебійної роботи та широкого функціоналу. У цьому контексті особливої актуальності набуває використання сучасних інструментів для розробки кросплатформних мобільних додатків, які дозволяють створити єдину кодову базу для Android та iOS і, таким чином, зменшити витрати на розробку та підтримку.

Серед таких технологій вирізняється фреймворк Flutter, створений компанією Google. Flutter активно розвивається і вже сьогодні застосовується в багатьох комерційних і стартап-проєктах у всьому світі. Його поєднання з мовою програмування Dart дозволяє швидко створювати гнучкі, масштабовані та привабливі мобільні інтерфейси, використовуючи систему візуальних компонентів (віджетів), що забезпечують нативний досвід для кінцевого користувача.

Тема створення інтернет-магазину смартфонів обрана не випадково. По-перше, це актуальна сфера для багатьох підприємств, що працюють у галузі рітейлу електроніки. По-друге, структура такого застосунку включає ключові для електронної комерції елементи: каталог продукції, картки товарів, кошик покупця, навігацію, функції пошуку, фільтрації та оформлення замовлення. Це дозволяє реалізувати повний цикл розробки типового комерційного додатку.

Таким чином, розробка мобільного додатку інтернет-магазину смартфонів із використанням Dart/Flutter є актуальною як у науково-прикладному аспекті, так і як практична реалізація сучасних підходів до створення програмних продуктів. Такий проєкт дозволяє не лише продемонструвати професійні навички з кросплатформного програмування, а й сформувати базу для майбутнього впровадження в реальні бізнес-процеси.

Мета кваліфікаційної роботи – розробити мобільний додаток інтернет-магазину смартфонів з використанням Dart/Flutter, що забезпечує зручний інтерфейс, ефективне управління каталогом товарів і функцію оформлення замовлення.

Об’єкт кваліфікаційної роботи – процес розробки кросплатформного мобільного додатку.

Предмет роботи – архітектура, інтерфейс та функціональність інтернет-магазину смартфонів, реалізованого з використанням технологій Dart/Flutter.

Завдання кваліфікаційної роботи:

– провести аналіз сучасного стану розробки мобільних застосунків у сфері електронної комерції;

– обґрунтувати вибір середовища розробки, мов програмування та технологій;

– сформулювати вимоги до функціональності, структури та дизайну інтернет-магазину;

– реалізувати додаток із використанням Flutter, враховуючи ключові функції: перегляд товарів, додавання в кошик, навігація, оформлення замовлення;

– провести тестування та UX/UI-аналіз розробленого програмного продукту;

– підготувати документацію для кінцевих користувачів та оцінити ефективність реалізованого рішення.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

У сучасних умовах розвитку цифрової економіки ринок електронної комерції демонструє стійке зростання. За даними міжнародних досліджень, частка мобільних пристроїв у структурі онлайн-продажів постійно зростає, що зумовлює необхідність створення ефективних, кросплатформних рішень для реалізації інтернет-магазинів. Серед користувачів надзвичайно популярними є мобільні додатки, які забезпечують високий рівень комфорту під час перегляду товарів, здійснення покупок і взаємодії з магазином [1].

Проаналізувавши доступні літературні джерела, наукові публікації та технічні огляди, можна зробити висновок, що одним із найбільш перспективних підходів до розробки мобільних застосунків є використання фреймворків із підтримкою кросплатформної архітектури [2]. У цьому контексті особливу увагу дослідників і практиків привертає Flutter, розроблений компанією Google.

У статті «Reasons Why Flutter is the Future of Mobile App Development?» автори зазначають, що ключовими перевагами Flutter є висока швидкість рендерингу інтерфейсів за допомогою вбудованого графічного рушія Skia, підтримка адаптивного дизайну, багата бібліотека візуальних компонентів (віджетів), а також можливість використання функції «гарячого перезапуску» (Hot Reload), що дозволяє розробникам миттєво переглядати зміни в коді без повної перекомпіляції застосунку.

Офіційна документація Flutter.dev та Dart.dev також підкреслює, що фреймворк Flutter спроектований для створення високопродуктивних і масштабованих мобільних застосунків із привабливим інтерфейсом. Стаття під назвою «An In-Depth Analysis of Flutter for Cross-Platform Mobile App Development», опублікована в журналі International Research Journal of Modernization in Engineering, Technology and Science (IRJMETS, листопад 2024),

присвячена детальному аналізу фреймворку Flutter, розробленого компанією Google для кросплатформної розробки мобільних додатків.

Крім того, велика кількість тематичних дописів та рецензій на технічних платформах (наприклад, стаття «10 Reasons Why You Should Use Flutter in 2023»), – підтверджують стабільне зростання інтересу до Flutter серед світової спільноти розробників. На GitHub-репозиторії Flutter нараховується понад 160 тисяч зірок (stars), що свідчить про високу популярність і активну підтримку фреймворку спільнотою.

Узагальнюючи вищезазначене, можна стверджувати, що Flutter сьогодні є одним із найефективніших рішень для реалізації мобільних додатків із кросплатформною архітектурою. Його застосування забезпечує високу якість інтерфейсу, спрощення процесу розробки та ефективну підтримку життєвого циклу мобільного застосунку.

Серед існуючих аналогів систем, функціональність яких наближена до проєктованого інтернет-магазину смартфонів, можна виокремити Rozetka, Comfy, Allo та Citrus. Усі ці платформи мають мобільні додатки, орієнтовані на користувачів Android та iOS, і широко використовуються в Україні для придбання електроніки, зокрема мобільних телефонів.

Мобільний застосунок Rozetka (рис. 1.1) дозволяє зручно переглядати широкий асортимент продукції, зокрема електроніки, побутової техніки, товарів для дому та інших категорій. Реалізовано розширений функціонал пошуку за категоріями, брендами, ціновими діапазонами, рейтингом, популярністю, що дозволяє швидко знаходити потрібні товари. Додаток підтримує авторизацію, збереження історії замовлень, функцію «Улюблені товари», бонусні програми та push-сповіщення про акції. Проте, незважаючи на функціональність, інтерфейс Rozetka іноді перевантажений великою кількістю рекламних банерів, акційних пропозицій, спливаючих вікон та вкладених меню, що ускладнює користування.

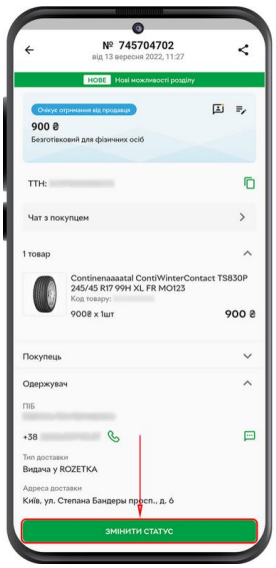

Рисунок 1.1 – Інтерфейс мобільного додатку Rozetka [3]

Інтерфейс мобільного додатку Comfy (рис. 1.2) реалізований у сучасному стилі з простим горизонтальним меню та розділами для швидкого доступу до популярних категорій товарів. Сторінка товару містить фотографії, основні характеристики, відгуки та кнопку для додавання в кошик. Слабким місцем додатку є обмежена кількість фільтрів у деяких категоріях та необхідність переходу через декілька екранів для оформлення покупки [4].

Рисунок 1.2 – Домашня сторінка мобільного застосунку Comfy [4]

У застосунку Allo (рис. 1.3) акцент зроблено на візуальний контент – великі банери, вітрини зі знижками, розширені картки товарів. Хоча це може бути ефективним з точки зору маркетингу, надмірне використання графіки уповільнює роботу програми на слабших пристроях. Крім того, існують труднощі з навігацією – користувач змушений прокручувати довгі списки, щоб знайти необхідну категорію або фільтр [5].

Рисунок 1.3 – Каталог товарів у мобільному додатку Allo [5]

Citrus (рис. 1.4) має приємний мінімалістичний інтерфейс, орієнтований на швидкий вибір смартфонів за брендом, новизною чи популярністю. Водночас, у мобільній версії спостерігається менша деталізація картки товару у порівнянні з веб-версією. Деякі користувачі також скаржаться на затримки в оновленні інформації про наявність товару або ціни [6].

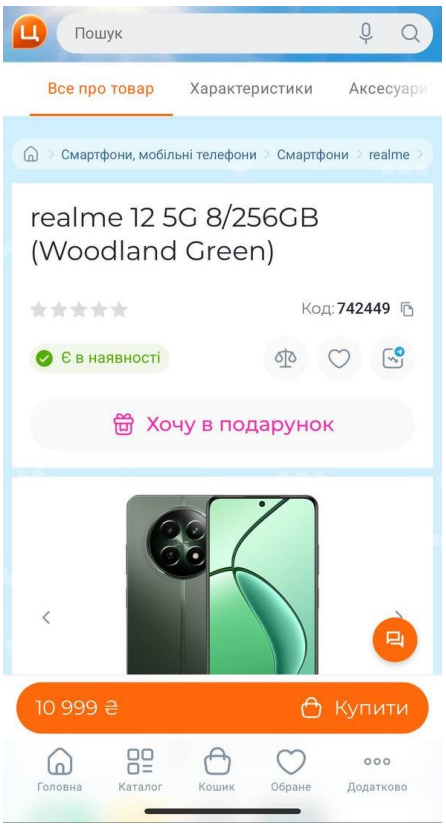

Рисунок 1.4 – Інтерфейс сторінки смартфонів у додатку Citrus [6]

Загалом, проаналізовані додатки мають схожі функціональні можливості: зручний пошук, візуальні елементи каталогу товарів, підтримку кошика, перегляд історії замовлень, тощо. Однак, спільною рисою є те, що всі вони мають окремі реалізації для Android та iOS, що ускладнює обслуговування, оновлення та супровід програмного забезпечення. Крім того, не всі додатки є однаково оптимізованими щодо продуктивності, швидкості завантаження або спрощеності інтерфейсу.

Ці особливості створюють передумови для розробки нового, легкого, зручного і швидкого у реалізації додатку на основі кросплатформного фреймворку Flutter, який дозволяє об’єднати переваги гнучкого дизайну, високої продуктивності й уніфікованого коду для різних платформ.

У процесі патентного пошуку не було виявлено зареєстрованих програмних продуктів, які б одночасно реалізовували інтернет-магазин із мобільним інтерфейсом, орієнтованим виключно на продаж смартфонів, побудований на основі Dart/Flutter. Це свідчить про доцільність створення нової

розробки, яка б поєднувала найкращі практики сучасного UI/UX-дизайну з технологічною ефективністю кросплатформенних фреймворків.

Застосування Flutter дозволяє спростити розробку, скоротити час на реалізацію основного функціоналу (від вітальної сторінки до кошика замовлень), уніфікувати структуру коду та забезпечити швидке розгортання застосунку. Особливо важливо, що Flutter підтримує гнучке масштабування та інтеграцію з різними бекенд-сервісами, що відкриває перспективи подальшого розвитку системи [7].

Таким чином, створення нового мобільного інтернет-магазину смартфонів з використанням Dart/Flutter є обґрунтованим рішенням, що враховує як технічні, так і практичні аспекти, та відповідає сучасним тенденціям у сфері програмної інженерії і мобільної торгівлі.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

На основі проведеного аналізу предметної області, літературних джерел, сучасних інженерних рішень та існуючих програмних аналогів сформульовано завдання на розробку мобільного застосунку – інтернет-магазину смартфонів, побудованого з використанням кросплатформного фреймворку Flutter та мови програмування Dart.

У межах кваліфікаційної роботи необхідно реалізувати повноцінну клієнтську частину застосунку, яка забезпечуватиме інтуїтивно зрозумілу навігацію, ефективну організацію каталогу товарів, відображення детальної інформації про кожен смартфон, можливість додавання товарів до кошика, а також оформлення замовлення. Застосунок має бути зручним, швидкодіючим, адаптивним до різних розмірів екранів та відповідати сучасним принципам UX/UI-дизайну.

Завдання розробки полягає не лише у створенні функціонального інтерфейсу, але й у впровадженні архітектурних рішень, які забезпечують масштабованість і зручність підтримки програмного коду. Важливо передбачити

можливість подальшої інтеграції із серверною частиною (API), що дозволить у майбутньому реалізувати облік користувачів, реєстрацію, авторизацію, збереження історії замовлень, тощо.

Загальна постановка завдання передбачає виконання таких основних етапів:

– аналіз предметної області та функціональних вимог до мобільного інтернет-магазину;

– вибір мови програмування та середовища розробки (Dart, Flutter, Android Studio);

– проєктування структури застосунку та інтерфейсу користувача;

– реалізація основних функцій (каталог товарів, пошук, перегляд картки товару, кошик, оформлення замовлення);

– проведення тестування працездатності та зручності використання;

– оформлення пояснювальної записки з описом архітектури рішення, інтерфейсу, функціоналу та інструкцією користувача.

Результатом виконання роботи має стати прототип кросплатформного мобільного застосунку інтернет-магазину смартфонів, що відповідає функціональним вимогам, та супровідна документація, яка містить опис реалізованих рішень і результати тестування.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

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

Обґрунтуємо вибір моделі та способів аналізу. Для моделювання програмного забезпечення було обрано об’єктно-орієнтований підхід, що є найкращим варіантом для реалізації програмного продукту у вигляді клієнтського застосунку. Основою стала модель UML (Unified Modeling Language), яка дозволяє створити наочні структурні та поведінкові діаграми. Такий підхід підтримує повторне використання компонентів, модульність, спрощене масштабування та тестування, що є критично важливим для мобільного додатку [8].

У процесі аналізу вимог було використано методи:

– аналіз аналогів (дослідження подібних мобільних застосунків Rozetka, Comfy тощо);

– аналіз користувацьких сценаріїв (вивчення поведінки користувачів у подібних системах);

– структуризація функціональних вимог;

– таблиці подія–реакція.

Функціональні вимоги:

– перегляд каталогу смартфонів (назва, зображення, рейтинг, ціна);

– перехід до детальної картки товару;

– додавання товарів у кошик;

– зміна кількості одиниць товару;

– видалення товарів з кошика;

– пошук за назвою;

– відображення банера з акціями;

– виведення діалогового вікна з підтвердженням дії;

– перехід між сторінками (вітальна, головна, кошик, детальна).

Нефункціональні вимоги:

– підтримка Android та iOS (кросплатформеність);

– відповідність UX/UI принципам;

– стабільна робота при мінімальному обсязі пам’яті;

– час відгуку інтерфейсу – не більше 100 мс;

– можливість масштабування архітектури (розширення функціоналу API, реєстрація користувачів тощо).

У процесі моделювання програмної системи було реалізовано кілька UML-діаграм, що дозволили формалізувати логіку роботи додатку, структуру його компонентів та взаємодію користувача з функціональністю. Зокрема, для візуалізації поведінки користувача із застосунком була побудована діаграма варіантів використання (Use Case Diagram) (рис. 2.1). Ця діаграма описує типові сценарії взаємодії між зовнішнім актором, яким у цьому випадку виступає користувач, та системою. Основні варіанти використання, передбачені в рамках додатку, включають перегляд каталогу товарів, відкриття детальної інформації про вибраний смартфон, додавання обраного товару до кошика, перегляд вмісту кошика з можливістю редагування, а також видалення товару з кошика. Кожен із цих варіантів є ключовим з погляду реалізації бізнес-логіки електронної комерції в мобільному середовищі.

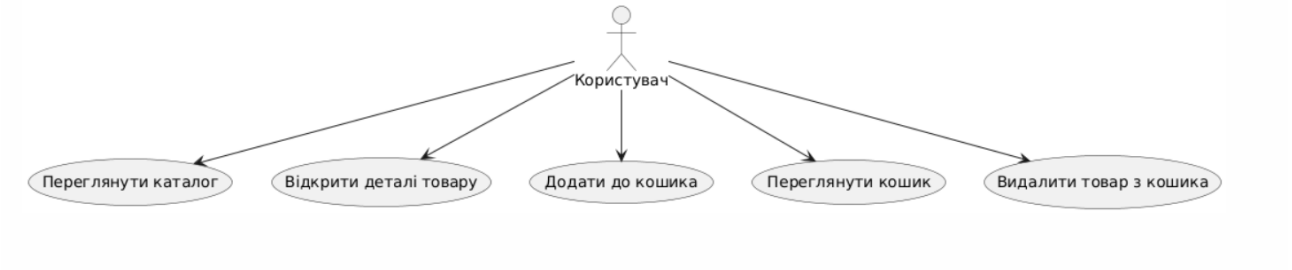

Рисунок 2.1 – Діаграма варіантів використання

Для відображення послідовності дій користувача, починаючи з запуску додатку і до завершення оформлення замовлення, було розроблено діаграму

діяльності (Activity Diagram) (рис. 2.2). Ця діаграма дозволила наочно подати логічний ланцюжок подій, який охоплює етапи перегляду каталогу, вибору певного товару, вивчення його деталей, додавання до кошика, переходу до списку замовлень та підтвердження покупки. Кожен етап моделюється як окрема активність, що дає змогу простежити логіку переходу між станами інтерфейсу та визначити потенційні точки для обробки винятків або помилок.

Рисунок 2.2 – Діаграма діяльності

Структура самої програмної системи представлена у вигляді діаграми класів (Class Diagram) (рис. 2.3). Ця діаграма демонструє, з яких об’єктів складається застосунок, які атрибути і методи вони мають, а також які між ними існують зв’язки. У системі було виділено ключовий клас Smartphone, що інкапсулює основні властивості товару – назву, ціну, зображення, рейтинг. Також визначено клас Cart, який відповідає за формування вмісту кошика. Він містить масив товарів, що були додані користувачем, а також функціональність для додавання, видалення товарів і розрахунку загальної суми. Для координації дій та управління даними в додатку створено клас Shop, який виконує роль

менеджера та посередника між візуальним інтерфейсом та логікою даних, зокрема обробляє запити на отримання товарів, відображення банерів та взаємодію з локальним сховищем.

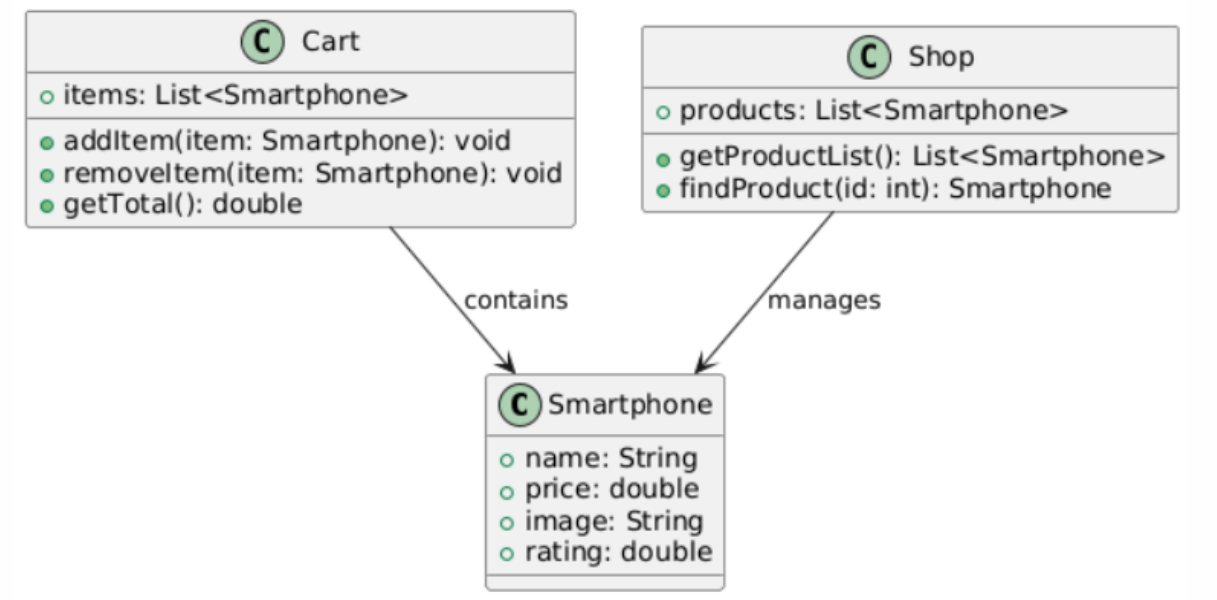

Рисунок 2.3 – Діаграма класів

Узагальнення таких моделей дозволяє отримати чітке бачення архітектури застосунку ще до початку реалізації, а також є важливим етапом формалізації технічного завдання, що сприяє якісній розробці та тестуванню програмного забезпечення.

UX/UI-дизайн. Застосовано принципи Material Design, адаптовані до Flutter:

– головна сторінка з банером, пошуком, горизонтальним списком товарів;

– інтерфейс кошика – список, іконка видалення;

– кнопки дій стилізовані з використанням GestureDetector, Container, Row, SizedBox;

– респонсивний дизайн – підтримка різних розмірів екранів.

Кожна сторінка реалізована через окремий виджет, що забезпечує модульність інтерфейсу. Було використано принципи логічної ієрархії, візуальної чіткості, передбачуваності елементів керування.

Програма працює за схемою:

– введення: взаємодія з інтерфейсом (натискання, пошук);
– обробка: локальна обробка в логіці класу Shop;
– зберігання: стан додатку зберігається у змінних стану;
– виведення: відображення нових сторінок або діалогових вікон.

Розроблена модель програмного забезпечення дозволяє реалізувати гнучку та масштабовану систему інтернет-магазину смартфонів. Застосування Flutter як фреймворку забезпечує швидку розробку та ефективне відображення інтерфейсу. Аналіз вимог і проектування показали доцільність обраного архітектурного рішення.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Процес розробки програмного забезпечення для інтернет-магазину смартфонів вимагає ретельного вибору інструментів, які не лише забезпечать реалізацію всіх необхідних функцій, а й відповідатимуть вимогам сучасної мобільної розробки, таким як кросплатформність, продуктивність, масштабованість, зручність у використанні та підтримці. Саме тому на початковому етапі проєкту було проведено детальний аналіз доступних фреймворків і мов програмування, які можуть бути використані для створення клієнтської частини мобільного застосунку.

Серед основних технологій, які були розглянуті, слід виділити Flutter, React Native, Xamarin, Apache Cordova та Kotlin Multiplatform Mobile. Аналіз показав, що хоча кожна з цих технологій має свої сильні сторони, саме Flutter продемонстрував найкраще поєднання продуктивності, гнучкості в побудові інтерфейсів, швидкості розробки та активної підтримки з боку спільноти та компанії-розробника – Google.

Серед основних технологій, які були розглянуті для розробки кросплатформного мобільного застосунку інтернет-магазину смартфонів,

найбільш релевантними виявилися Flutter, React Native, Xamarin, Apache Cordova та Kotlin Multiplatform Mobile. Кожна з цих платформ має свої унікальні можливості, архітектурні підходи та обмеження, тому на етапі технічного аналізу доцільним було здійснити їх порівняння за основними критеріями: швидкодія, простота розробки, підтримка UI-компонентів, спільнота, документація, продуктивність і рівень нативності.

Flutter є розробкою компанії Google і, на момент реалізації проєкту, вважався одним із найбільш перспективних фреймворків для створення інтерфейсів користувача. Його основною перевагою є власний рушій рендерингу Skia, який дозволяє створювати кросплатформні інтерфейси без опори на нативні елементи Android або iOS. Це означає, що весь інтерфейс візуалізується однаково на різних платформах, без необхідності писати платформоспецифічний код. Flutter відзначається підтримкою реактивного програмування, великою бібліотекою готових віджетів, високою продуктивністю завдяки компіляції Ahead-of-Time та зручністю розробки завдяки функції hot reload. Його офіційна документація добре структурована, а активна спільнота швидко розв’язує питання на GitHub, Stack Overflow та форумах [9].

React Native, розроблений Facebook (Meta), є ще одним популярним фреймворком для кросплатформної розробки. На відміну від Flutter, React Native використовує JavaScript і працює через міст між JavaScript і нативними компонентами операційної системи. Це дозволяє використовувати нативні UI-компоненти, але водночас створює затримки через необхідність постійної взаємодії між JavaScript-логікою і нативним рівнем. Перевагами React Native є велика кількість готових компонентів, багата екосистема, підтримка Redux, а також легка інтеграція з існуючими Android/iOS-проєктами. Однак для складних анімацій чи графіки, де потрібна максимальна продуктивність, React Native часто поступається Flutter [10].

Xamarin – це фреймворк компанії Microsoft, який дозволяє створювати мобільні застосунки за допомогою мови C# та платформи .NET. Xamarin компілюється в нативний код і надає змогу створювати як повністю

кросплатформні застосунки, так і такі, що мають спільну бізнес-логіку з окремими UI для Android і iOS. Основною перевагою Xamarin є інтеграція з екосистемою Microsoft, включно з Visual Studio, Azure, DevOps та іншими сервісами. Проте цей фреймворк має складніший поріг входу, більш важку конфігурацію і значно меншу спільноту, ніж Flutter чи React Native. Крім того, застосунки, створені на Xamarin, іноді мають більший розмір пакету та вищу залежність від платформонативних SDK [11].

Apache Cordova – це платформа з відкритим кодом, яка дозволяє створювати гібридні мобільні застосунки з використанням HTML, CSS та JavaScript. У Cordova застосунок працює як веб-сторінка всередині WebView компонента. Це забезпечує максимальну гнучкість у верстці, але негативно впливає на продуктивність та інтеграцію з нативними функціями платформи. Cordova підходить для простих додатків з мінімальною графікою, але не є найкращим вибором для складних мобільних систем із високими вимогами до інтерфейсу або анімації. Хоча існує великий набір плагінів, їх підтримка не завжди стабільна, що ускладнює масштабування проєкту.

Kotlin Multiplatform Mobile (KMM) – це сучасна технологія від JetBrains, яка дозволяє ділити бізнес-логіку між Android та iOS, залишаючи UI частину платформоспецифічною. Вона базується на мові Kotlin і забезпечує високу продуктивність, оскільки частина коду виконується як нативна. Проте KMM ще знаходиться в активній фазі розвитку і має обмежену підтримку в частині інструментів, готових бібліотек і прикладів. Ця платформа є доцільною для команд, які вже працюють із Kotlin, особливо якщо пріоритет – це максимальна продуктивність і контроль над кожною частиною застосунку. Проте вона вимагає розробки окремих UI для Android та iOS, що збільшує обсяг роботи [12].

Загалом, проведене порівняння показало, що найбільш збалансованим рішенням для реалізації цільового проєкту є Flutter, оскільки він забезпечує нативну продуктивність, дозволяє швидко створювати адаптивні інтерфейси, має потужну підтримку інструментів розробки та активну спільноту, а також дає можливість писати спільний код для Android і iOS без потреби створення

окремих UI. Це дозволило значно пришвидшити розробку, зменшити витрати та забезпечити якісне користувацьке враження від взаємодії з додатком.

Flutter є фреймворком із відкритим кодом, який дозволяє створювати високопродуктивні, візуально привабливі мобільні застосунки для платформ Android та iOS з використанням єдиного коду. Це значно скорочує час та ресурси, необхідні для розробки і тестування. Його основною перевагою є використання власного рушія рендерингу Skia, що забезпечує плавне та нативне відображення UI-компонентів незалежно від платформи. Flutter підтримує реактивний підхід до створення інтерфейсу та повністю синхронізований із принципами Material Design і Cupertino для Android та iOS відповідно [13-15].

Крім фреймворку, ключовим компонентом у реалізації застосунку є мова програмування Dart, яка поєднує в собі об’єктно-орієнтовану модель, підтримку узагальнень (generics), а також синтаксис, подібний до JavaScript та Java. Вона має вбудовану підтримку реактивного програмування, що дозволяє зручно працювати зі станом додатку та асинхронною логікою (через async/await, Future, Stream).

Для розробки середовищем було обрано Android Studio – інтегровану IDE з офіційною підтримкою Flutter, яка забезпечує засоби автозаповнення коду, інтеграцію емуляторів пристроїв, інструменти налагодження та візуалізації, зручне керування залежностями проєкту через файл pubspec.yaml.

Під час реалізації мобільного інтернет-магазину з використанням Flutter було використано низку сторонніх бібліотек і пакетів, які значно розширили функціональні можливості застосунку, спростили реалізацію окремих компонентів та підвищили зручність взаємодії з користувачем. Кожен із застосованих пакетів мав конкретне функціональне призначення та був інтегрований у відповідні модулі системи. Наведемо детальний опис кожного з них [16-17].

Одним із ключових інструментів став пакет provider, який використовується для організації управління станом (state management). Він реалізує патерн «спостерігач» (Observer), що дозволяє розділити логіку

управління даними та візуальний інтерфейс. Це дає змогу централізовано зберігати дані (наприклад, вміст кошика або список обраних товарів) і передавати їх між екранами без дублювання коду. Provider забезпечує автоматичне оновлення віджетів, коли стан змінюється, що покращує реактивність додатку та підтримку принципу «Single Source of Truth».

Для візуалізації оцінки товарів було використано пакет flutter_rating_bar, який дозволяє відображати рейтинг у вигляді графічних зірок. Цей елемент інтерфейсу є звичним для користувачів і відіграє важливу роль при виборі товару. Пакет надає гнучкі параметри налаштування вигляду зірок, розміру, кольору, кількості та типу взаємодії (пасивне відображення або активне оцінювання). В даному застосунку компонент рейтингу використовувався у картці товару для демонстрації середнього значення на основі відгуків.

Для реалізації інтерактивного банеру на головній сторінці, який демонструє акційні товари або новинки, було обрано пакет carousel_slider. Цей компонент дозволяє виводити серію зображень або віджетів у вигляді горизонтального слайдера з анімацією, автоматичним прокручуванням та індикаторами. Карусель є важливим елементом UI, оскільки вона привертає увагу користувача до ключових пропозицій. Використання цього пакету дозволило створити адаптивну карусель, що коректно працює на різних розмірах екранів.

Зважаючи на потребу використання якісної графіки, яка не втрачає чіткості при масштабуванні, до проєкту було додано підтримку векторних зображень через бібліотеку flutter_svg. Векторні формати (SVG) є особливо корисними для логотипів, іконок та елементів інтерфейсу, оскільки дозволяють уникнути пікселізації та підтримують гнучку зміну кольору. Пакет flutter_svg забезпечує парсинг SVG-файлів та їх інтеграцію як стандартних віджетів Flutter без значного навантаження на продуктивність.

Для підвищення зручності взаємодії з користувачем було застосовано бібліотеку fluttertoast, яка використовується для виведення коротких текстових повідомлень – так званих «toast notifications». Вони виникають при певних діях

користувача, таких як додавання товару в кошик, очищення списку або помилки при заповненні форми. Перевагою toast є ненав’язливість – повідомлення зникає через кілька секунд і не потребує додаткових дій від користувача. Це покращує зворотній зв’язок у застосунку.

Ще однією важливою бібліотекою став пакет intl, що забезпечує локалізацію (переклад інтерфейсу) та форматування числових значень, дат, валют відповідно до культурних стандартів. У рамках проєкту intl використовувався для форматування цін товарів у вигляді, прийнятному для українських користувачів, із врахуванням локалі, десяткових роздільників та символу валюти. Крім того, ця бібліотека дозволяє у майбутньому легко масштабувати застосунок для мультимовної підтримки, що є важливим з огляду на потенційну комерціалізацію продукту.

Щодо методів реалізації логіки, в додатку використовувався принцип компонентного підходу, за якого кожен екран – окремий stateful або stateless widget, із відповідною логікою ініціалізації, побудови, взаємодії з користувачем та оновлення. Дані про товари були інкапсульовані в об'єкти класу Smartphone, які містили такі поля, як назва, ціна, зображення та рейтинг. Операції додавання, вилучення, підрахунку загальної суми в кошику реалізовані через методи відповідного класу Cart.

З алгоритмічної точки зору в системі застосовувалися лінійні алгоритми пошуку, сортування масивів об’єктів, агрегації даних та фільтрації. Наприклад, обчислення вартості товарів у кошику здійснювалося простим проходом по списку обраних товарів і підсумовуванням їх цін. Видалення товару реалізоване через фільтрацію масиву за унікальним ідентифікатором, а динамічна побудова списків – через виджети ListView.builder та GridView.count.

У розробці інтерфейсу також було враховано принципи UX/UI-дизайну. Всі елементи розміщено з урахуванням зручності використання, адаптивності до різних розмірів екранів, логічної ієрархії блоків. Інтерфейс не перевантажено інформацією – всі елементи згруповано у блоки: банер, каталог, кошик, деталі товару.

Для візуального моделювання структури програми було розроблено діаграму класів, що ілюструє зв’язки між основними об’єктами системи. У ній показано, що об’єкти типу Smartphone зберігаються у класі Shop, а об’єкти типу Cart оперують ними для формування замовлення. Також побудовано діаграму варіантів використання, яка демонструє основні сценарії взаємодії користувача з системою, а діаграма діяльності описує основні кроки процесу замовлення товару – від першого перегляду каталогу до підтвердження покупки.

Таким чином, обрані засоби, інструменти й методи повністю відповідають поставленому завданню – створити сучасний, легкий у використанні, зручний та продуктивний інтернет-магазин смартфонів, реалізований у вигляді кросплатформного мобільного додатку з використанням найкращих практик сучасної розробки програмного забезпечення.

РОЗДІЛ 3
РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНУ СМАРТФОНІВ

3.1 Практична реалізація об’єкта проектування

Реалізація програмного продукту – мобільного інтернет-магазину смартфонів – виконувалась із використанням мови програмування Dart та фреймворку Flutter у середовищі IntelliJ IDEA. Основна мета полягала у створенні сучасного, зручного для користувача додатку з підтримкою кросплатформної роботи – тобто запуску на Android і iOS з єдиною кодовою базою.

Програмна реалізація була структурована у вигляді багатофайлової архітектури, де логіка та інтерфейс були розділені між окремими модулями. Загальна структура коду включала основні файли: main.dart, intro_page.dart, menu_page.dart, phone_details_page.dart, cart_page.dart, а також допоміжні – shop.dart, smartphone.dart, button.dart, phone_tile.dart, colors.dart. Такий поділ дозволив чітко організувати код та забезпечити його масштабованість.

Одним із базових елементів головного екрану є рекламний банер, що реалізований через Container з кольоровим фоном, текстовим повідомленням та кнопкою MyButton, яка може бути переорієнтована на рекламну акцію. Окрім банеру, головна сторінка містить пошукове поле, список товарів, віджет PhoneTile, який відображає кожен смартфон із зображенням, назвою, ціною та рейтингом. Для переходу до сторінки деталей товару використовувався метод navigateToPhoneDetails, що передає об'єкт смартфона у відповідний екран через Navigator.push.

Функціонал кошика реалізовано через окремий клас Cart, який зберігає масив обраних товарів і надає методи для додавання та видалення. Доступ до цього об’єкта реалізовано через бібліотеку provider, яка дозволяє організувати глобальне управління станом додатку. Усі дії користувача – додавання товару до кошика, перегляд деталей, видалення позицій – оновлюють UI завдяки реактивному зв’язку між станом і віджетами.

Для реалізації зворотного зв’язку застосовано пакет fluttertoast, який відображає повідомлення про успішне виконання дії (наприклад, додавання товару). Усі товари формуються з моделі Smartphone, яка містить поля назва, ціна, зображення, рейтинг.

У лістингу 3.1 наведено реалізацію уніфікованої кнопки, яка змінює поведінку залежно від контексту.

Лістинг 3.1 – Кнопка, що має однаковий вигляд і різний функціонал

```
return GestureDetector(
      onTap: onTap,
      child: Container(
        decoration: BoxDecoration(
          color: Colors.grey[400],
          borderRadius: BorderRadius.circular(40),
        ),
        padding: EdgeInsets.all(20),
        child: Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text(
              text,
              style: TextStyle(color: Colors.black),
            ),
            const SizedBox(width: 10),
            Icon(
              Icons.arrow_forward,
              color: Colors.black,
            ),
          ],
        ),
      ),
    );
```

кінець лістингу 3.1

Цей фрагмент коду формує кнопку, яка включає в себе текст та іконку, дозволяючи виконати певну дію при клацанні на неї.

GestureDetector – цей віджет призначений для реагування на різні дії користувача, включаючи операцію натискання.

Container є контейнерним об’єктом, що вміщує кнопку та відповідає за її стиль та розміри.

Елементи в рядку будуть вирівняні по центру позначає mainAxisAlignment: MainAxisAlignment.center.

SizedBox – це відступ між текстом та іконкою.

Icon – іконка, що розташована праворуч від тексту, використовуючи Icons.arrow_forward, яка вказує напрямок руху вперед.

В лістингу 3.2 реалізується логіка додавання нового товару до таблиці DataGridView на основі введ.ених користувачем значень.

Лістинг 3.2 – Методи для зменшення та збільшення кількості товарів

```
int quantityCount = 0;

// зменшення кількості
void decrementQuantity() {
  setState(() {
    if (quantityCount > 0) {
      quantityCount--;
    }
  });
}

// збільшення кількості
void incrementQuantity() {
  setState(() {
    quantityCount++;
  });
}
```

кінець лістингу 3.2

Змінна quantityCount створена для зберігання кількості одиниць товару у кошику. Функція decrementQuantity викликається для зменшення обсягу продуктів у кошику на одиницю. У блоку setState перевіряється наявність товарів: якщо поточна кількість товарів більше 0, значення quantityCount зменшується на одиницю. Цей метод пов’язаний з функціоналом кнопки, що дозволяє зменшити кількість товару у кошику. У випадку, якщо quantityCount вже має значення 0, додаткове зменшення не відбувається.

Лістинг 3.3 демонструє метод обчислення загальної вартості всіх товарів, доданих до таблиці, з урахуванням кількості та ціни.

Лістинг 3.3 – Метод для додавання товару до кошика

```
void addToCart() {
  if (quantityCount > 0) {
    final shop = context.read<Shop>();
    // додати в кошик
    shop.addToCart(widget.phone, quantityCount);
    // let the user know it was successful
    showDialog(
      context: context,
      barrierDismissible: false,
      builder: (context) => AlertDialog(
        backgroundColor: primaryColor,
        content: const Text(
          "Successfully added to cart",
          style: TextStyle(color: Colors.white),
          textAlign: TextAlign.center,
        ),
        actions: [
          IconButton(onPressed: () {
            Navigator.pop(context);
            Navigator.pop(context);
          },
            icon: const Icon(
              Icons.done,
              color: Colors.white,
            ),
          )
        ],
      ),
    );
  }
}
```

кінець лістингу 3.3

Спочатку перевіряється, чи кількість одиниць товару більша за 0. Це гарантує, що товар буде доданий до кошика лише в разі наявності хоча б однієї одиниці. Якщо кількість товару перевищує 0, отримується доступ до класу Shop, який містить інформацію про кошик покупок. Далі викликається метод addToCart() цього класу, де передаються дані про товар (widget.phone) і кількість цього товару (quantityCount). Це викликає додавання товару до кошика в потрібній кількості.

Після успішного додавання товару в кошик, виводиться спливаюче вікно (AlertDialog), яке повідомляє користувача про успішне виконання. Кнопка «okay» внизу цього вікна слугує для закриття діалогового вікна та повернення до попереднього екрану.

Обидва Navigator.pop(context) виклики потрібні для закриття діалогового вікна та повернення користувача на попередній екран.

Наведений код (ліст. 3.4) відповідає за збереження вмісту таблиці до текстового файлу з використанням класу SaveFileDialog та методу File.WriteAllText.

Лістинг 3.4 – Список товарів

```
// phone menu
final List<Smartphone> _phoneMenu = [

  Smartphone(
    name: "Redmi Note 12 Pro",
    price: "277.74",
    imagePath: "lib/images/Xiaomi Redmi Note 12Pro.png",
    rating: "4.7",
  ),

  Smartphone(
    name: "Xiaomi 13T",
    price: "555.49",
    imagePath: "lib/images/Xiaomi 13T.png",
    rating: "4.9",
  ),
];
```

кінець лістингу 3.4

Змінна _phoneMenu є списком об’єктів класу Smartphone. Ці дані (назва, ціна, зображення та рейтинг) використовуються для подальшого відображення телефонів у списку товарів, доступних для покупки в додатку.

У лістингу 3.5 реалізовано механізм завантаження даних з файлу у таблицю DataGridView з розбором рядків і додаванням записів.

Лістинг 3.5 – Управління вмістом кошика покупця

```
  List<Smartphone> _cart = [];

List<Smartphone> get phoneMenu => _phoneMenu;
List<Smartphone> get cart => _cart;

void addToCart(Smartphone phoneItem, int quantity) {
  for (int i = 0; i < quantity; i++) {
    _cart.add(phoneItem);
  }
```

```
  notifyListeners();
}

void removeFromCart(Smartphone phone) {
  _cart.remove(phone);
  notifyListeners();
}
```

кінець лістингу 3.5

Список _cart – це кошик, який спочатку не містить жодного товару.

Геттери, які надають доступ до _phoneMenu (списку доступних товарів для покупки) та _cart (кошик покупця) – це відповідно get phoneMenu і get cart. Геттери дозволяють іншим частинам програми отримувати доступ до цих списків.

Функція addToCart додає товари до кошика покупця. Параметр phoneItem представляє товар Smartphone, який потрібно додати, а quantity вказує, скільки його одиниць слід додати. Метод використовує цикл for для додавання вказаної кількості товару до кошика.

Метод removeFromCart видаляє товар із корзини покупця. Видаляється товар зазначеного типу Smartphone, який був доданий раніше до кошика.

В лістингу 3.6 реалізовано динамічний список для відображення товарів, що знаходяться у кошику покупця.

Лістинг 3.6 – Відображення товарів у кошику

```
  Expanded(
              child: ListView.builder(
                itemCount: value.cart.length,
                itemBuilder: (context, index) {

                  final Smartphone phone = value.cart[index];

                  final String phoneName = phone.name;

                  final String phonePrice = phone.price;
```

кінець лістингу 3.6

Expanded – розширює свій вміст, щоб зайняти доступний простір всередині батьківського віджету.

ListView.builder – це віджет, який дозволяє створювати список елементів за допомогою конструктора, який створює елементи списку за запитом.

Параметр itemCount вказує віджету скільки елементів він має будувати. Кількість елементів визначається через довжину списку value.cart.

Змінна phone отримує конкретний смартфон зі списку value.cart на основі index. Вона використовується для отримання доступу до властивостей смартфона, таких як його назва та ціна.

Після запуску програми, перед користувачем з’являється вітальна сторінка, яка зображена на рисунку 3.1.

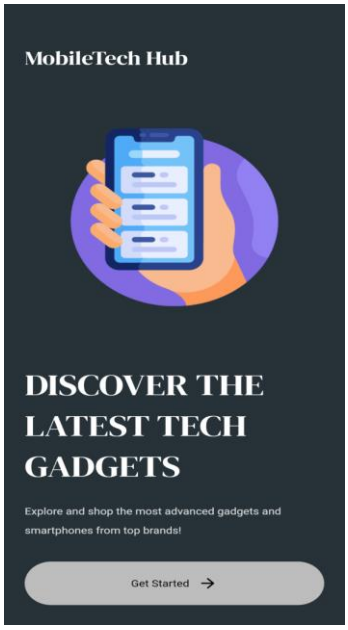

Рисунок 3.1 – Вітальна сторінка

Після натиснення кнопки, клієнта переводить на головну сторінку. Тут він має змогу переглянути та вибрати потрібний йому товар. На рисунку 3.2 зображено головну сторінку.

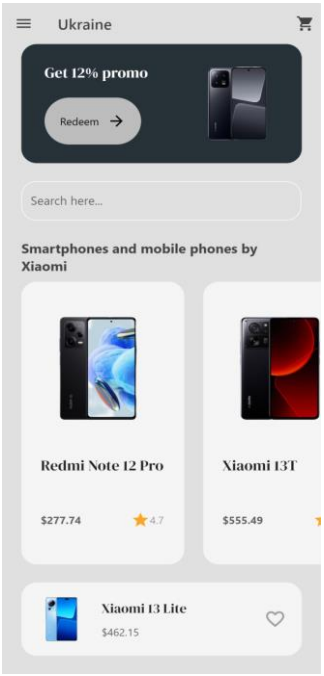

Рисунок 3.2 – Головна сторінка

Після того, коли користувач натисне на вибраний ним товар, його перенаправить на детальну сторінку із цим продуктом. На ній він може більш детально ознайомитись та зробити покупку, натиснувши спочатку на кнопку вибору кількості, а потім додати в кошик. На рисунку 3.3 наведено детальну сторінку.

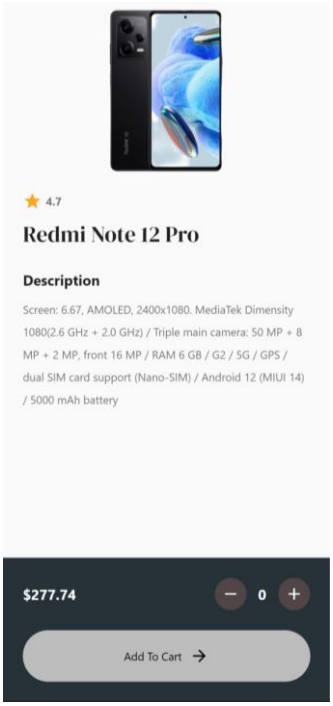

Рисунок 3.3 – Детальна сторінка

Щойно клієнт здійснить додавання товару в кошик, перед ним з’явиться діалогове вікно про успішне виконання, яке можна переглянути на рисунку 3.4.

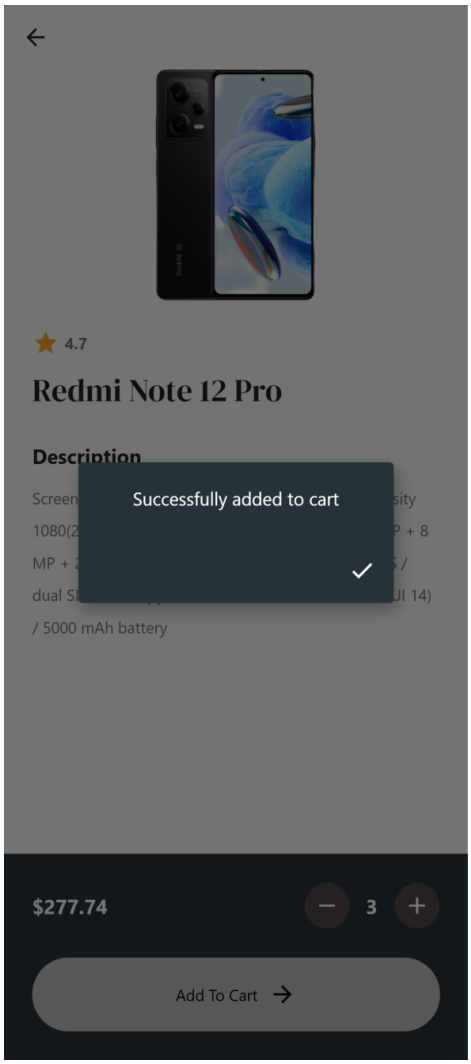

Рисунок 3.4 – Діалогове вікно про успішне виконання

На головній сторінці розміщена іконка кошика, яка дозволяє переглянути всі додані покупки. У цьому списку кожен товар представлений із назвою, ціною, зображенням та кількістю. Для кожної позиції передбачена кнопка видалення, що дає змогу користувачеві оперативно видалити будь-який товар із кошика. Після натискання на цю кнопку товар миттєво зникає зі списку, а загальна сума автоматично оновлюється. Такий підхід забезпечує зручність управління замовленням та покращує взаємодію з інтерфейсом. На рисунку 3.5 наведено сторінку із доданим товаром.

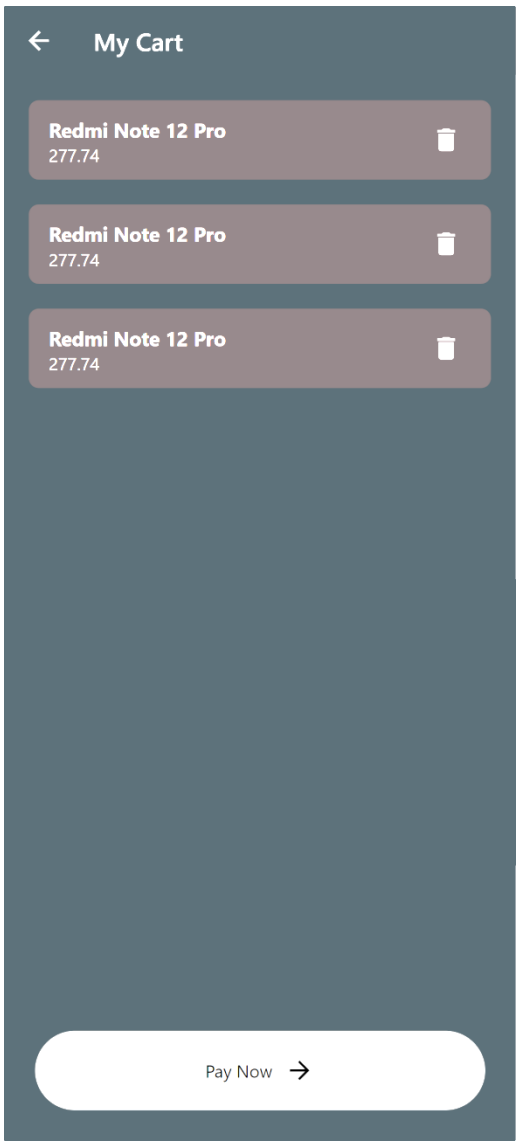

Рисунок 3.5 – Сторінка кошика

3.2 Тестування та налагодження системи

Процес тестування та налагодження є важливим етапом життєвого циклу розробки програмного забезпечення, що дозволяє виявити логічні, синтаксичні, функціональні та інтерфейсні помилки до моменту впровадження програмного продукту у використання. Для забезпечення стабільної роботи мобільного додатку інтернет-магазину було проведено поетапне тестування, яке включало ручне (мануальне) тестування, тестування інтерфейсу користувача, перевірку роботи бізнес-логіки та функціонального навантаження додатку.

На етапі налагодження застосовувалося інтерактивне тестування у середовищі Android Studio, яке підтримує зручну перевірку коду завдяки інтегрованому емулятору мобільного пристрою, системі журналювання (логів) та дебагеру. За допомогою механізму hot reload стало можливим миттєво застосовувати зміни в коді без перезапуску всього застосунку, що значно прискорило процес усунення помилок, зокрема пов’язаних із відображенням віджетів, переходами між екранами та оновленням стану інтерфейсу.

Під час тестування була перевірена коректність роботи таких ключових функцій:

– виведення головного екрана із банером, списком товарів та пошуковим полем;

– перехід до детальної картки товару з повною інформацією;

– додавання товару до кошика та його відображення у відповідному розділі;

– можливість змінювати кількість одиниць товару або видаляти позиції з кошика;

– відображення системних повідомлень (toast) після виконання дій;

– збереження стану при переході між сторінками;

– адаптивна верстка на екранах різних розмірів (емуляція різних пристроїв).

Програмний код також пройшов статичний аналіз засобами Android Studio – виявлені помилки типізації, невикористані змінні, дублювання коду були своєчасно виправлені. Було оптимізовано структуру проєкту: дублікати логіки винесені у функції, повторювані компоненти – у окремі віджети (PhoneTile, MyButton).

При налагодженні було виявлено низку типових недоліків:

– некоректне відображення товарів при використанні GridView на екранах із низьким DPI;

– відсутність обробки ситуацій, коли список товарів порожній;

– неможливість видалення товару при одиничній кількості (було виправлено через валідацію в методі decrementQuantity);

– неправильне форматування ціни – усунено через використання пакету intl.

Усі ці помилки були усунені шляхом модифікації відповідних методів, додавання перевірок та покращення логіки обробки подій. У результаті застосунок став стабільним, передбачуваним у поведінці та стійким до помилок введення.

Мінімальні системні вимоги до запуску застосунку:

– операційна система: Android 7.0+ або iOS 11.0+;

– оперативна пам’ять: від 2 ГБ;

– вільне місце на пристрої: не менше 150 МБ;

– процесор: 4-ядерний ARM або x86 з підтримкою JIT-компіляції;

– наявність інтернет-з’єднання – бажана, але не обов’язкова (у поточній реалізації всі дані локальні).

Рекомендовані параметри:

– операційна система: Android 10+ або iOS 13+;

– оперативна пам’ять: 4 ГБ і більше;

– дисплей: 5,5 або більше з роздільною здатністю не менше 720x1280;

– наявність Google Play Services (для подальшого розширення функціоналу).

Таким чином, тестування і налагодження додатку підтвердили його працездатність на базовому рівні функціональності, забезпечили стабільну взаємодію з інтерфейсом користувача та дозволили усунути виявлені помилки до моменту розгортання застосунку. Результати тестування свідчать про готовність системи до подальшого вдосконалення, зокрема інтеграції з базою даних або зовнішнім сервером API.

3.3 Розробка документації для програмного продукту

Розробка технічної та користувацької документації є важливим етапом завершення розробки інформаційно-комп'ютерної системи. Документація дозволяє не лише забезпечити належну експлуатацію програмного продукту кінцевими користувачами, але й полегшує підтримку та розвиток системи в майбутньому. Для створеного мобільного застосунку інтернет-магазину смартфонів було підготовлено комплект супровідних матеріалів, що охоплюють вимоги до системи, інструкції користувача, опис основного інтерфейсу та опис можливих помилок.

Готовий застосунок збирається в інсталяційний файл .apk за допомогою інструментів Flutter SDK. Інсталяція відбувається стандартним способом для Android-пристроїв:

– увімкнути дозвіл на встановлення програм із зовнішніх джерел;

– завантажити файл smartphone_store.apk з локального носія або хмарного сховища;

– запустити файл на пристрої;

– після інсталяції – відкрити застосунок через головне меню пристрою.

В майбутньому застосунок може бути завантажений до Google Play після додаткової підготовки release-збірки та проходження перевірки на відповідність політикам платформи.

Мінімальні системні вимоги:

– операційна система: Android 7.0 або новіше;

– оперативна пам'ять: від 2 ГБ;

– процесор: ARM 32/64 або x86;

– вільне місце: не менше 150 МБ;

– підключення до інтернету: опціонально (на поточному етапі – не обов'язкове).

Опишемо інструкцію користувача. Після запуску додатку користувач потрапляє на головну сторінку, яка містить:

– банер із рекламою акційних товарів;

– поле для пошуку товарів;

– список доступних смартфонів у вигляді плиток (назва, ціна, зображення, рейтинг);

– нижнє меню з переходами: «Головна», «Кошик».

При натисканні на плитку товару відкривається сторінка деталей, де можна ознайомитися з характеристиками пристрою та додати його до кошика. Кількість одиниць регулюється за допомогою кнопок «+» і «–». Після додавання виводиться повідомлення-підтвердження (toast).

У розділі «Кошик» відображаються всі додані товари, їхня кількість, ціна, загальна вартість замовлення. Тут можна змінювати кількість, видаляти позиції та очищати кошик повністю. Оформлення покупки реалізовано як кнопка-фінал, яка виводить повідомлення про успішне формування замовлення (функціонал моделюється, оскільки реального підключення до сервера немає) (табл. 3.1).

Таблиця 3.1 – Опис функціональних кнопок у розділі «Кошик» мобільного застосунку

Назва кнопки	Призначення
+ / –	Збільшення або зменшення кількості товару
Add to cart	Додати обраний смартфон до кошика
Remove	Видалити товар з кошика
Clear cart	Очистити весь вміст кошика
Buy now	Підтвердити оформлення замовлення (з демонстраційною дією)
Back	Повернення до попереднього екрану

У поточній версії застосунку реалізовано обробку таких ситуацій:

– спроба зменшити кількість товару нижче 1 – блокується на рівні логіки;

– порожній кошик при переході до оформлення – повідомлення про необхідність додати товар;

– невалідні введення в пошуку – обробка через порожній результат без помилок;

– повідомлення користувачеві подаються у вигляді коротких текстів за допомогою пакету fluttertoast.

Отже, розроблена документація забезпечує легкість у використанні програмного продукту, полегшує ознайомлення нових користувачів із функціоналом додатку, а також є важливим елементом для супроводу системи в майбутньому.

3.4 UX/UI-аналіз розробленої системи

UX/UI-аналіз (User Experience / User Interface) є критично важливим етапом оцінки якості мобільного застосунку, особливо у сфері електронної комерції, де зручність і привабливість інтерфейсу безпосередньо впливають на поведінку користувачів, рівень довіри та ймовірність повторного використання додатку. У процесі розробки інтернет-магазину смартфонів основна увага приділялась досягненню інтуїтивно зрозумілого, адаптивного та візуально привабливого інтерфейсу з мінімалістичним дизайном, що відповідає сучасним стандартам [18].

Обґрунтування структури та навігації. Інтерфейс застосунку побудований на основі принципів спрощеної ієрархії та логічної послідовності взаємодії, що відповідає моделі поведінки користувача в типових мобільних торгових платформах. Основні елементи інтерфейсу згруповані на окремих екранах:

– головна сторінка – банер, пошук, каталог товарів (плитки);

– сторінка деталей товару – опис, рейтинг, ціна, кнопки керування;

– кошик – список доданих товарів із кількістю, сумою і кнопкою оформлення.

Нижнє меню навігації забезпечує швидкий доступ до головних розділів. Такий підхід дозволяє уникнути надмірного вкладення екранів та зменшує час досягнення цільових дій (наприклад, купівля товару).

Основна кольорова гама застосунку – світлі нейтральні відтінки з акцентами на кнопках і банері. Цей вибір дозволяє зберігати візуальний баланс і

не відволікати увагу користувача від основного контенту – зображень товарів, цін і кнопок дій.

Тексти виконані стандартним шрифтом Flutter (Roboto), що забезпечує читабельність на будь-якому пристрої. Розміри шрифтів адаптовано відповідно до значущості інформації (назва – більша, рейтинг/ціна – середня, кнопки – компактна типографіка).

Інтерфейс побудовано відповідно до принципів Material Design, рекомендованих Google для Android-застосунків. Водночас завдяки Flutter інтерфейс автоматично адаптується і до iOS-платформи з використанням елементів Cupertino.

Застосовані компоненти (Scaffold, AppBar, ListView, GridView, Container, ElevatedButton) забезпечують коректне відображення інтерфейсу незалежно від розміру екрана. Ретельно протестовано поведінку UI на пристроях з різною діагоналлю та роздільною здатністю – всі елементи залишаються читабельними, інтуїтивно зрозумілими і не перекриваються.

Для підвищення ефективності взаємодії реалізовано:

– жести свайпу та натискання, зокрема через GestureDetector;

– візуальний зворотний зв’язок при виконанні дій (додавання в кошик, зміна кількості товару) – за допомогою fluttertoast;

– спрощене керування кількістю товарів – «+» і «–» з обмеженням нижньої межі;

– підтвердження виконаних дій – повідомлення у форматі Toast та зміна стану інтерфейсу.

Час, необхідний на здійснення основної дії (вибір → додавання до кошика → перегляд кошика → покупка), не перевищує 4 кліки, що відповідає сучасним критеріям швидкого UX.

Попри загальну ефективність інтерфейсу, було виявлено кілька потенційних напрямків для вдосконалення:

– додавання темної теми оформлення для зниження навантаження на зір;

– реалізація плавної анімації переходів між екранами;

- покращення UX за допомогою фільтрації та сортування товарів;
- впровадження розділу «Обране» для збереження вибраних товарів.

UX/UI-аналіз засвідчив, що створений інтерфейс відповідає основним вимогам зручності, естетики, простоти навігації та адаптивності. Система орієнтована на кінцевого користувача, забезпечує інтуїтивну логіку взаємодії та комфортне користування незалежно від технічних характеристик пристрою. Реалізовані рішення заклали основу для подальшого масштабування функціоналу без втрати якості користувацького досвіду.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було повністю реалізовано повноцінний кросплатформний мобільний додаток інтернет-магазину, орієнтованого на продаж смартфонів, із використанням сучасного фреймворку Flutter та мови програмування Dart. Розробка охоплювала повний цикл створення програмного продукту – від аналізу предметної області й постановки вимог до реалізації, тестування, UX/UI-аналізу та підготовки супровідної документації.

У ході дослідження було проаналізовано актуальний стан проблеми створення інтернет-магазинів, проведено огляд існуючих аналогічних мобільних застосунків (Rozetka, Comfy, Allo, Citrus) та визначено їхні переваги й недоліки. На основі цього було обґрунтовано доцільність створення власного, полегшеного додатку з фокусом на мінімалізм, кросплатформність та простоту використання.

У кваліфікаційній роботі бакалавра:

– здійснено аналіз сучасного стану розробки мобільних застосунків у сфері електронної комерції. Проаналізовано тенденції цифрової торгівлі, вивчено функціональні особливості популярних мобільних додатків (Rozetka, Comfy, Allo, Citrus). Оцінено переваги кросплатформних рішень, обґрунтовано актуальність створення нового застосунку;

– здійснено обґрунтування вибору середовища розробки, мов програмування та технологій. Проведено порівняльний аналіз сучасних фреймворків (Flutter, React Native, Xamarin, Cordova, KMM). Обрано Flutter як оптимальний інструмент для створення інтерфейсів, а Dart – як зручну мову з підтримкою реактивного програмування. Визначено Android Studio як базове середовище розробки. Описано використані бібліотеки: provider, flutter_rating_bar, carousel_slider, flutter_svg, intl;

– сформульовано вимоги до функціональності, структури та дизайну інтернет-магазину. Визначено функціональні та нефункціональні вимоги до додатку. Сформовано UML-діаграми (використання, діяльності, класів), які

описують поведінку користувача та архітектуру системи. UX/UI-дизайн побудовано за принципами Material Design з адаптивною верстkою та модульністю;

– реалізовано додаток із використанням Flutter з урахуванням ключових функцій. Створено мобільний інтернет-магазин із підтримкою Android та iOS. Реалізовано: вітальна сторінка, головний екран з банером, каталог смартфонів, картка товару, кошик, навігація, оформлення замовлення. Забезпечено гнучкість та масштабованість коду. Використано власні віджети, реалізовано керування станом через Provider;

– проведено тестування та UX/UI-аналіз розробленого програмного продукту. Здійснено ручне тестування на емуляторі Android Studio. Перевірено основні функції додатку, виправлено типові помилки (форматування цін, адаптивність GridView тощо). Оцінено інтерфейс на зручність використання, відповідність принципам UX. Проведено налагодження та оптимізацію структури коду;

– підготовлено документацію для кінцевих користувачів та оцінено ефективність реалізованого рішення. Описано інструкції з інсталяції, використання функціоналу, вимоги до пристроїв. Зібрано APK-файл для Android. Здійснено опис функціональних кнопок та взаємодії користувача з додатком. Визначено потенціал для розширення функціоналу (підключення до API, реєстрація користувачів, база даних).

У результаті виконаної роботи розроблений мобільний додаток відповідає поставленим вимогам: забезпечує зручний доступ до товарів, дозволяє працювати з кошиком, реалізує просту навігацію, а також може бути основою для подальшого розвитку – зокрема підключення бази даних, системи авторизації, або виходу в публічний доступ через Google Play.

Результати дослідження демонструють ефективність використання Dart/Flutter як сучасного засобу для створення адаптивних, продуктивних і кросплатформних мобільних застосунків, що особливо актуально в умовах високих вимог до швидкості розробки та багатоплатформності продуктів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Заполочний А., Левикін В. Дослідження методів розроблення вебсайту інтернет-магазину / Збірник тез доповідей Міжнародної науково-практичної конференції «Інформаційні системи та технології». ХНУРЕ, 2024. С. 68-71. URL: https://ist-conf-nure.com.ua/IST-2024_part-2.pdf (дата звернення: 10.04.2025).

2. Сокрут А. А. Розробка інтернет-магазину роздрібного продажу товарів: кваліфікаційна робота бакалавра. 2021. 61с. URL: https://krs.chmnu.edu.ua/jspui/bitstream/123456789/2298/1/607%20Сокрут%20Альбіна%20Андріївна.pdf (дата звернення: 10.04.2025).

3. Rozetka. URL: https://rozetka.com.ua (дата звернення: 10.04.2025).

4. Comfy. URL: https://comfy.ua (дата звернення: 10.04.2025).

5. Allo. URL: https://allo.ua (дата звернення: 10.04.2025).

6. Citrus. URL: https://www.citrus.ua (дата звернення: 10.04.2025).

7. Flutter documentation. URL: https://flutter.dev/docs (дата звернення: 15.04.2025).

8. Бородкіна І. Л., Бородкін Г. О. Інженерія програмного забезпечення : навч. посіб. Київ : ЦУЛ, 2019. 204 с.

9. Dart programming language documentation. URL: https://dart.dev/guides (дата звернення: 15.04.2025).

10. React Native Documentation. Офіційна документація React Native – фреймворку для розробки кросплатформних мобільних застосунків з використанням JavaScript та React. URL: https://reactnative.dev/docs/getting-started (дата звернення: 15.04.2025).

11. Xamarin Documentation. Офіційна документація Xamarin – платформи для розробки кросплатформних мобільних застосунків з використанням C# та .NET. URL: https://dotnet.microsoft.com/en-us/apps/xamarin (дата звернення: 15.04.2025).

12. Kotlin Multiplatform Documentation. URL: https://kotlinlang.org/docs/multiplatform.htmlKotlin+7 (дата звернення: 15.04.2025).

13. Pub.dev. Flutter and Dart packages library. URL: https://pub.dev (дата звернення: 20.04.2025).

14. GitHub. Flutter Samples and Open Source Projects. URL: https://github.com/flutter (дата звернення: 20.04.2025).

15. Medium. Flutter development articles. URL: https://medium.com/flutter (дата звернення: 20.04.2025).

16. Stack Overflow. Q&A platform for Dart and Flutter development. URL: https://stackoverflow.com (дата звернення: 23.04.2025).

17. Навчальний посібник з Flutter для початківців. URL: https://spacelab.ua/lessons/uchebnoe-posobie-po-flutter-dlya-nachinayushchih/ (дата звернення: 23.04.2025).

18. Babich N. Mobile UX design: trends and best practices. UX Planet, 2023. URL: https://uxplanet.org/mobile-ux-trends-2023 (дата звернення: 23.04.2025).