Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: Розробка інтерактивного програмного застосунку для онлайн-книгарні

Виконав: студент 4 курсу
групи ЗПІ-20Б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Бірюк Назар Федорович
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Бабюк Н.П.
(прізвище та ініціали)
Рецензент:. к.т.н., доц. каф. ОТ Войцеховська О.В.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри ПЗ Романюк О.Н.
« 10 » червня 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
« 12 » березня 2024 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Бірюку Назару Федоровичу

1. Тема роботи – розробка інтерактивного програмного застосунку для онлайн-книгарні.

Керівник роботи: Бабюк Наталя Петрівна, к.т.н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 р. № 80.

2. Строк подання студентом роботи 3 червня 2024 р.

3. Вихідні дані до роботи: базові алгоритми для обробки даних – алгоритм обробка мережевих HTTP-запитів, алгоритм валіадації введених даних, алгоритм для визначення рекомендованих книг; використані бібліотеки – android.util.Log, androidx.compose.runtime.mutableStateOf, dagger.hilt.android.lifecycle, kotlinx.coroutines.flow.MutableStateFlow, javax.inject.Inject, com.fahad.auth_firebase; середовище розробки – Android Studio; мова програмування – Kotlin.

4. Зміст текстової частини: вступ, аналіз актуальності розробки програмного застосунку для онлайн-книгарні, порівняльний аналіз аналогів, постановка завдань до роботи, аналіз структури алгоритмів та формування проблематики програмного забезпечення, розробка загальної моделі застосунку, дослідження алгоритмів роботи системи, розробка програмних модулів інтерактивного програмного застосунку для онлайн-книгарні, обґрунтування вибору засобів для реалізації програмного застосунку, обґрунтування вибору середовища розробки програмного застосунку, розробка модулів програмного застосунку, висновки

5. Перелік ілюстративного матеріалу: титульний слайд; мета, об'єкт та предмет дослідження; задачі дослідження; новизна і практична цінність одержаних результатів; загальний алгоритм роботи системи; розробка модулю реєстрації та входу користувача до профілю; розробка модулю додавання книг у кошик та список улюбленого; демонстрація інтерфейсу застосунку «головної сторінки додатку»; демонстрація інтерфейсу застосунку «вибору зацікавленої

книги»; демонстрація інтерфейсу застосунку «корзина»; демонстрац
інтерфейсу застосунку «списку улюблених книг»; демонстрація інтерфей
застосунку «профілю користувача»; Апробація та публікація матеріал
бакалаврської дипломної роботи; фінальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Бабюк Н.П., к.н.т., доцент каф. ПЗ	13.03.24 р.	03.06.24 р.

7. Дата видачі завдання 13 березня 2024 р

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану онлайн-книгарень у сучасному світу	14.03.24 – 30.03.24	вик
2	Аналіз аналогів інтерактивного програмного застосунку онлайн-книгарні	01.04.24 – 15.04.24	вик
3	Розробка алгоритмів застосунку	16.04.24 – 20.04.24	вик
4	Розробка модулів програмного застосунку	21.04.24 – 15.05.24	вик
5	Тестування програмного застосунку	16.05.24 – 22.05.24	вик
6	Оформлення матеріалів до захисту БДР	22.05.24 – 03.06.24	вик

Студент _____ (підпис) **Н.Ф. Бірюк** (ініціали та прізв

Керівник бакалаврської дипломної роботи _____ (підпис) **Н. П. Бабюк** (ініціали та прізв

АНОТАЦІЯ

УДК 004.04

Бірюк Н.Ф. Розробка інтерактивного програмного застосунку для онлайн-книгарні: бакалаврська дипломна робота зі спеціальності 121 Інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця : ВНТУ, 2024. 59 с.

На укр. мові. Бібліогр. : 20 назв.; рис. : 19; табл. : 5.

Бакалаврська дипломна робота спрямована на розробку інтерактивного програмного застосунку для онлайн-книгарні. У ході дослідження було здійснено аналіз актуальності онлайн-книгарень, проведено порівняльний аналіз існуючих аналогів та визначено ключові завдання для розробки додатку.

Реалізовано програмні модулі, такі як модуль для реєстрації, додавання книг у список улюблених та інші ключові функції. Було обґрунтовано вибір засобів та середовища для реалізації програмного продукту, а саме Kotlin, як мову програмування та Android Studio, як середовище для розробки.

Також було проведено тестування застосунку, а також визначено технічні вимоги.

Ключові слова: онлайн-книгарня, інтерактивний додаток, програмне забезпечення, розробка, алгоритми, функціональність, технічні вимоги, тестування, користувацький інтерфейс, зручність.

ABSTRACT

UDC 004.04

Biriuk N.F. Development of an interactive software application for an online bookstore: bachelor's thesis in the specialty 121 Software Engineering, educational program – software engineering. Vinnytsia: VNTU, 2024. 59 p.

In Ukrainian. Language. Refs. : 20 titles; rice. : 19; Table. : 5.

The bachelor's thesis focuses on the development of an interactive application for an online bookstore. The study involved analyzing the relevance of online bookstores, conducting a comparative analysis of existing analogs, and identifying key tasks for the application development.

Software modules, such as user registration, adding books to the favorites list, and other key functions, have been implemented. The choice of tools and development environment was substantiated, specifically Kotlin as the programming language and Android Studio as the development environment.

The application was also tested, and technical requirements were defined.

Keywords: online bookstore, interactive application, software development, algorithms, functionality, technical requirements, testing, user interface, usability.

ЗМІСТ

ВСТУП

Розвиток інформаційних технологій значно вплинув на всі сфери життя, зокрема на бізнес та торгівлю [1]. Однією з яскравих змін є перехід до електронної комерції, яка забезпечує зручність, швидкість та доступність покупок для користувачів. Онлайн-книгарні стають дедалі популярнішими завдяки можливості купувати книги з будь-якої точки світу, не виходячи з дому. Це зумовлено зростаючим попитом на цифрові сервіси та електронну комерцію, що робить їх невід'ємною частиною сучасного ринку.

Проте, більшість існуючих платформ для онлайн-продажу книг мають певні обмеження в функціональності та зручності використання [2]. Серед основних проблем, з якими стикаються користувачі, можна виділити: обмежений вибір книг, складність навігації, недостатня інтерактивність, відсутність персоналізованих рекомендацій та недосконала система пошуку. Ці недоліки створюють потребу у вдосконаленні таких систем, щоб забезпечити кращий користувацький досвід і відповідати сучасним вимогам ринку.

Тому розробка інтерактивного програмного застосунку для онлайн-книгарні є актуальною темою. Такий застосунок має забезпечити зручний та інтуїтивно зрозумілий інтерфейс, який дозволить користувачам легко знаходити необхідні книги, отримувати рекомендації на основі їхніх уподобань, читати рецензії інших користувачів та залишати власні відгуки. Крім того, інтеграція системи оповіщення про новинки та акції, а також можливість створення власних книжкових списків, зробить користування застосунком ще більш комфортним та приємним.

Враховуючи швидкий темп розвитку інформаційних технологій та зростаючий попит на електронну комерцію, розробка сучасного інтерактивного застосунку для онлайн-книгарні є необхідністю для успішного функціонування на ринку [3]. Такий підхід дозволить не тільки задовольнити потреби користувачів, але й підвищити конкурентоспроможність компанії, що надає ці послуги.

Сучасні аналоги програмних застосунків для онлайн-книгарень, хоча й виконують базові функції продажу та управління асортиментом, часто мають значні недоліки. Зокрема, багато з них характеризуються незручним та застарілим інтерфейсом, що ускладнює навігацію та пошук потрібних товарів.

Крім того, відсутність інтерактивних елементів та персоналізованих рекомендацій обмежує можливості покращення користувацького досвіду. Також існує проблема недостатньої інтеграції з сучасними платіжними системами та іншими сервісами, що знижує ефективність комерційних процесів.

Всі ці недоліки створюють потребу у розробці нових, більш вдосконалених рішень, які б відповідали сучасним вимогам та очікуванням користувачів.

Метою роботи є розробка інтерактивного програмного застосунку для онлайн-книгарні, який забезпечить зручне користування, ефективну організацію продажів та підвищення рівня задоволеності клієнтів. Для досягнення поставленої мети необхідно вирішити наступні задачі:

– аналіз існуючих рішень для онлайн-продажу книг та виявлення їх недоліків;

– визначення основних функціональних вимог до програмного застосунку;

– розробка архітектури та дизайну користувацького інтерфейсу застосунку;

– реалізація програмного застосунку з використанням сучасних технологій;

– тестування та оцінка ефективності розробленого застосунку;

– впровадження інтерактивних елементів для покращення користувацького досвіду.

Об'єктом дослідження виступає процеси розробки та організації онлайн-продажу книг через власне ПЗ.

Предметом дослідження виступає програмний застосунок для онлайн-книгарні, який включає в себе функціональні можливості для здійснення покупок, управління асортиментом та взаємодії з користувачами.

Новизна роботи полягає у наступного:

Створений інтерактивний програмний застосунок для онлайн-книгарні, поєднує в собі сучасні технології та підходи до розробки інтерактивних користувацьких інтерфейсів, що значно покращує користувацький досвід для більш ефективного та зручного користування системою порівняно з існуючими аналогами.

Практичне значення отриманих результатів полягає у можливості застосування розробленого програмного продукту в реальних умовах функціонування онлайн-книгарень. Це сприятиме підвищенню ефективності продажів, розширенню клієнтської бази та покращенню загальної конкурентоспроможності бізнесу. Крім того, розроблений застосунок може служити основою для подальших досліджень та вдосконалення в галузі електронної комерції.

Апробація та публікація результатів курсового проєкту. Результати курсового проєкту доповідалися на конференції Молодь в науці: дослідження, проблеми, перспективи (МН-2024) і опубліковані в тезах доповіді цієї конференції [4].

1 ФОРМУЛЮВАННЯ ЗАВДАНЬ ТА АНАЛІЗ АКТУАЛЬНОСТІ РОЗРОБКИ ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ ОНЛАЙН-КНИГАРНІ

1.1 Аналіз актуальності онлайн-книгарень

В останні десятиліття інформаційні технології радикально змінили спосіб, яким люди отримують доступ до товарів і послуг. Онлайн-книгарні стали невід'ємною частиною сучасного ринку книг, пропонуючи численні переваги як для споживачів, так і для видавців та авторів. Актуальність онлайн-книгарень зумовлена кількома ключовими факторами.

Сучасні технології, такі як машинне навчання, хмарні обчислення та інтерактивні елементи, відкривають нові можливості для розвитку онлайн-книгарень. Впровадження персоналізованих рекомендацій дозволяє створювати індивідуальний підхід до кожного клієнта, підвищуючи задоволеність користувачів та їх лояльність. Використання хмарних технологій забезпечує стабільність та масштабованість системи, дозволяючи обробляти великий обсяг даних та забезпечувати безперебійний доступ до інформації.

По-перше, зручність та доступність. Онлайн-книгарні надають можливість купувати книги в будь-який час та з будь-якого місця, де є доступ до Інтернету [5]. Це особливо важливо для людей з обмеженим часом або тих, хто проживає в регіонах з обмеженим доступом до фізичних книгарень.

По-друге, широкий асортимент. Онлайн-книгарні можуть пропонувати набагато ширший вибір книг порівняно з традиційними магазинами, оскільки не обмежені фізичним простором. Це дозволяє користувачам знаходити рідкісні видання, зарубіжні публікації та інші спеціалізовані матеріали.

По-третє, ціни та акції. Завдяки відсутності витрат на оренду приміщень та утримання фізичних магазинів, онлайн-книгарні можуть пропонувати конкурентоспроможні ціни. Крім того, вони часто проводять акції, розпродажі та надають знижки, що робить купівлю книг ще вигіднішою для споживачів.

По-четверте, персоналізація та рекомендації. Використання сучасних технологій, таких як алгоритми машинного навчання, дозволяє онлайн-

книгарням аналізувати вподобання та поведінку користувачів, пропонуючи персоналізовані рекомендації. Це не лише покращує користувацький досвід, але й сприяє збільшенню продажів.

По-п'яте, екологічність. Поширення електронних книг зменшує потребу у виробництві паперових видань, що позитивно впливає на збереження навколишнього середовища.

Актуальність онлайн-книгарень продовжує зростати, відповідаючи на виклики та можливості сучасного світу. Інтерактивні програмні застосунки для онлайн-книгарень, що використовують новітні технології, здатні ще більше покращити користувацький досвід та задовольнити потреби сучасних споживачів.

Аналіз актуальності онлайн-книгарень вказує на важливість постійного вдосконалення їх функціональності та користувацького досвіду. З огляду на це, розробка інноваційного програмного застосунку для онлайн-книгарні стає не лише актуальним, а й необхідним завданням для забезпечення конкурентоспроможності на сучасному ринку.

Враховуючи всі зазначені фактори, стає очевидним, що розробка інтерактивного програмного застосунку для онлайн-книгарні не тільки задовольняє поточні потреби ринку, але й прокладає шлях до майбутніх інновацій. Такий застосунок повинен включати в себе передові технології, які дозволять здійснювати персоналізацію пропозицій, інтегрувати зручні та безпечні платіжні системи, а також забезпечувати високу продуктивність та надійність сервісу. Успішна реалізація потребує міждисциплінарного підходу, включаючи знання в галузі програмування, дизайну користувацького інтерфейсу, маркетингу та аналізу даних.

Розробка цього застосунку може служити основою для подальших досліджень та вдосконалення в області електронної комерції. Крім того, вона може бути застосована не лише в книгарнях, але й адаптована для інших видів онлайн-торгівлі, що робить її універсальною та перспективною інвестицією. Таким чином, впровадження новітніх технологій в онлайн-книгарнях сприятиме

не тільки підвищенню їх конкурентоспроможності, але й створенню нового стандарту якості обслуговування в електронній комерції.

1.2 Порівняльний аналіз аналогів

У цьому розділі проведено порівняльний аналіз існуючих програмних застосунків для онлайн-книгарень, що використовуються на ринку, з метою виявлення їх переваг та недоліків. Аналіз допомагає зрозуміти, які саме аспекти можуть бути вдосконалені в процесі розробки нового інтерактивного програмного забезпечення для онлайн-книгарні.

Для аналізу обрано кілька відомих платформ: Amazon, Barnes & Noble та Kobo. Кожна з цих платформ має свої унікальні риси, які роблять їх популярними серед користувачів, проте вони також мають певні недоліки, що можуть бути виправлені в новому програмному продукті.

Amazon є одним з найбільших онлайн-рітейлерів у світі, пропонуючи широкий асортимент книг у різних форматах, включаючи друковані та електронні видання [6]. На рисунку 1.1 наведено приклад інтерфейму Amazon.

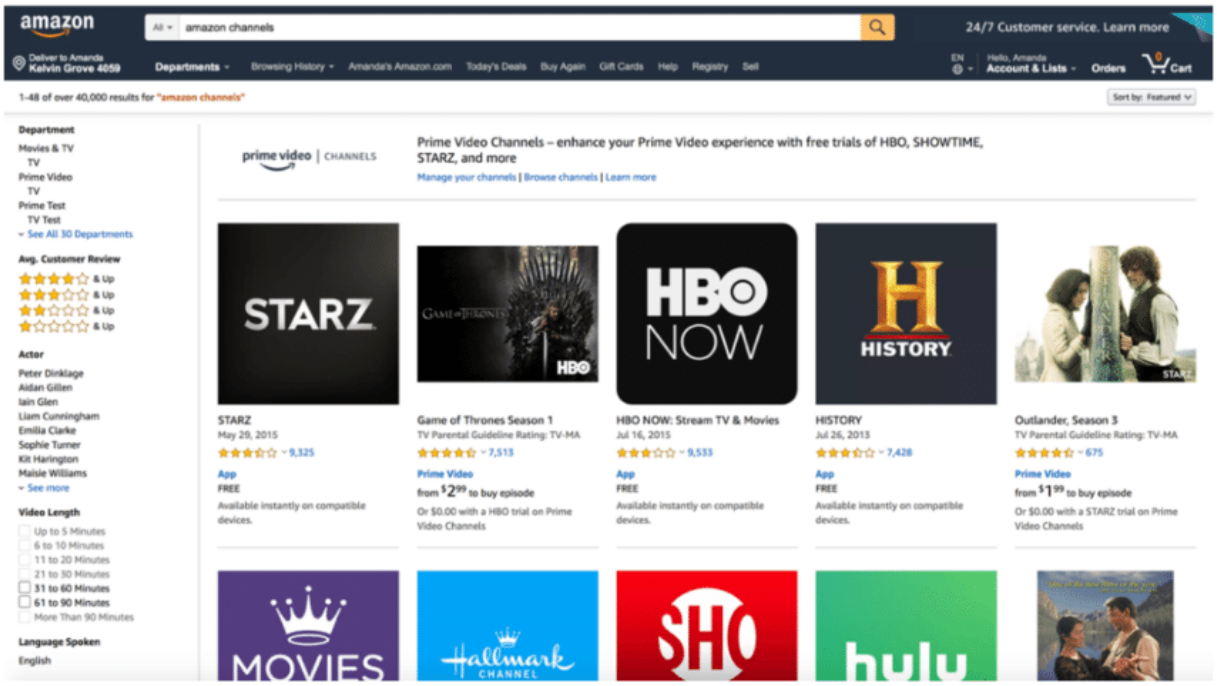

Рисунок 1.1 – Приклад інтерфейсу Amazon

Основні переваги Amazon полягають у великому виборі книг, що включає рідкісні та зарубіжні видання, а також у зручності користування завдяки простому інтерфейсу, швидкому пошуку та численним опціям фільтрації. Крім того, алгоритми персоналізованих рекомендацій дозволяють аналізувати історію покупок та вподобання користувачів, пропонуючи їм релевантні книги.

Важливою перевагою є також сервіс Prime Membership, який надає доступ до безкоштовної доставки та інших ексклюзивних пропозицій для підписників.

Проте, висока конкуренція серед продавців може ускладнювати пошук унікальних пропозицій, а відсутність інтерактивних елементів обмежує можливості покращення користувацького досвіду.

Barnes & Noble є однією з найбільших книжкових мереж у США з сильною онлайн-присутністю [7]. Переваги цієї платформи включають надійність бренду, що забезпечує довіру з боку великої кількості постійних клієнтів (див. рисунок 1.2).

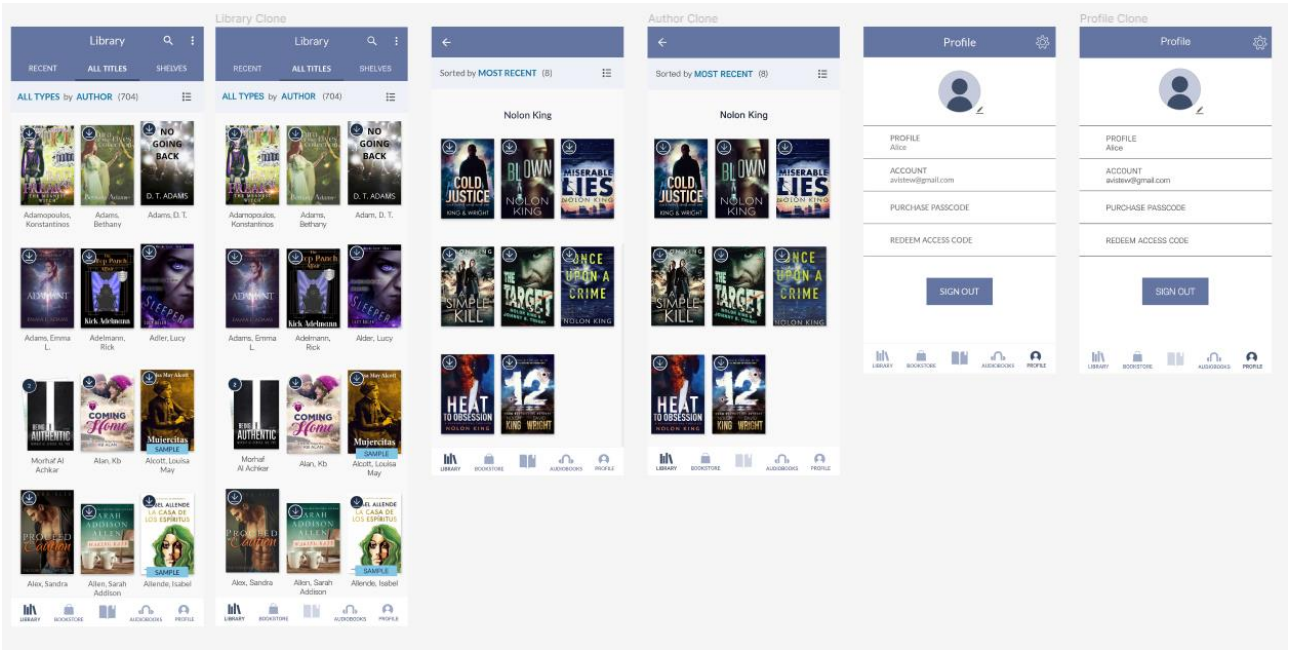

Рисунок 1.2 – Приклад інтерфейсу «Barnes & Noble»

Важливою особливістю є наявність власного електронного рідера NOOK, який створює цілісну екосистему для купівлі та читання електронних книг. Крім

того, Barnes & Noble організовує онлайн-заходи та книжкові клуби для користувачів, що сприяє створенню спільноти читачів.

Однак, у порівнянні з Amazon, Barnes & Noble має обмежений асортимент книг, а інтерфейс потребує модернізації для більш зручного використання. Крім того, алгоритми рекомендацій менш ефективні, що знижує рівень персоналізації.

Kobo є популярним вибором серед користувачів, які віддають перевагу електронним книгам [8]. Основною перевагою цієї платформи є широкий вибір електронних книг, що охоплюють різноманітні жанри та видання (див. рисунок 1.3).

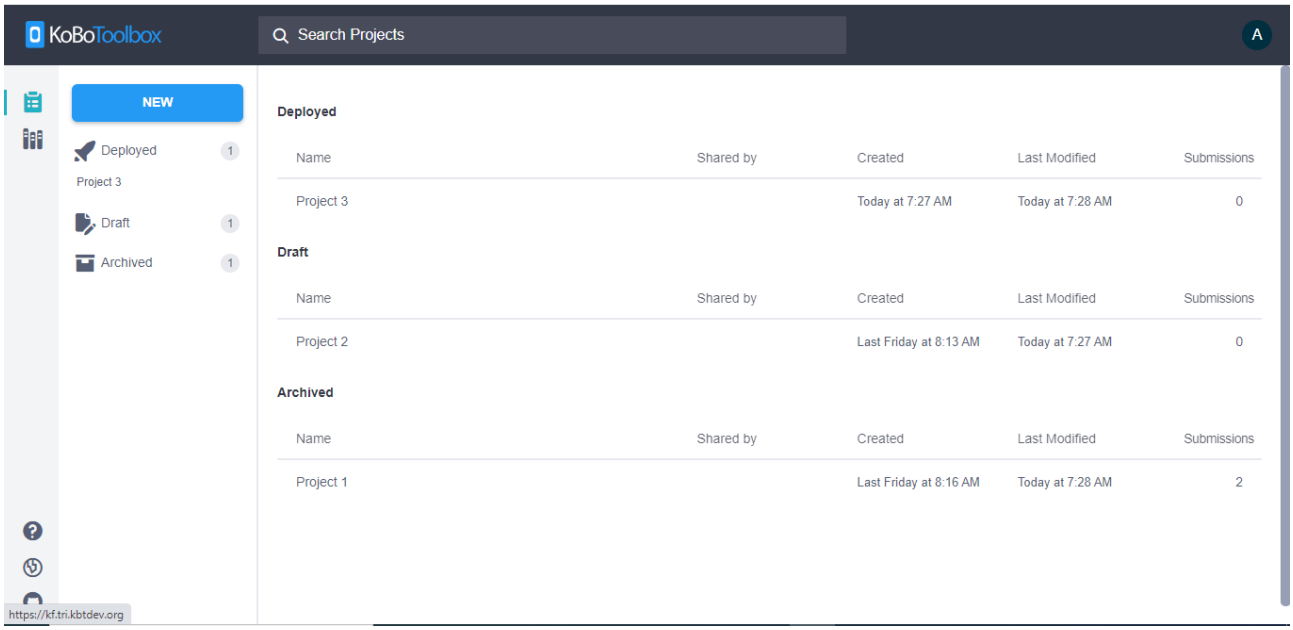

Рисунок 1.3 – Приклад інтерфейсу «Kobo»

Kobo також пропонує можливість синхронізації між різними пристроями, що дозволяє читачам продовжувати читання на будь-якому зручному для них пристрої. Інтерактивні функції, такі як інтеграція з соціальними мережами та можливість ділитися цитатами й нотатками, роблять користування платформою більш привабливим.

Проте, фокус на електронні видання обмежує вибір друкованих книг, а недостатня локалізація у деяких регіонах зменшує доступність різноманітних

мов та локальних авторів. Також алгоритми рекомендацій потребують вдосконалення для більш точної персоналізації.

Аналіз показує, що кожен із розглянутих застосунків має свої унікальні переваги та недоліки. Amazon пропонує найбільший асортимент та зручність, але може бути перенасичений товарами. Barnes & Noble надає надійність бренду та додаткові функції, проте потребує оновлення інтерфейсу. Kobo спеціалізується на електронних книгах та пропонує інтерактивні можливості, але має обмежений вибір друкованих видань.

Результати аналізу були занесені до таблиці 1.1, що дозволяє наочно порівняти основні переваги та недоліки існуючих платформ, а також виявити ключові можливості для вдосконалення в новому програмному застосунку.

Таблиця 1.1 – Порівняльний аналіз аналогів

Характеристика	**Amazon**	**Barnes & Noble**	**Kobo**	**БДР**
Широкий асортимент	+	–	–	+
Зручність користування	+	+/–	+	+
Персоналізація рекомендацій	+	+/–	+/–	+
Інтерактивні функції	–	–	+	+
Синхронізація між пристроями	+	–	+	+
Локалізація	+	+	–	+
Надійність бренду	+	+	+/–	+
Ціни та акції	+/–	+	+/–	+
Екологічність	+/–	–	+	+

Amazon пропонує великий вибір книг, включаючи рідкісні та зарубіжні

видання. Barnes & Noble та Kobo мають обмежений асортимент, тоді як новий додаток забезпечує широкий вибір книг.

Amazon має простий інтерфейс та зручну навігацію. Barnes & Noble потребує модернізації інтерфейсу, а Kobo забезпечує зручність користування. Новий додаток також буде зручним у користуванні і відповідатиме всім правилам зручного інтерфейсу.

Розглядаючи Amazon, він ефективно використовує алгоритми рекомендацій. Barnes & Noble та Kobo мають менш ефективні алгоритми, тоді як новий додаток забезпечує високу якість персоналізації.

Інтерактивні функції Amazon та Barnes & Noble мають певні обмеження. Kobo пропонує інтеграцію з соціальними мережами та інші інтерактивні можливості. Новий додаток також включає різноманітні інтерактивні функції.

Amazon та Kobo дозволяють синхронізувати прогрес читання між різними пристроями, що забезпечує зручність для користувачів. Barnes & Noble не має цієї функції. Новий додаток буде підтримувати синхронізацію між пристроями.

Amazon та Barnes & Noble підтримують різні мови та локальних авторів, тоді як Kobo має обмежену локалізацію. Новий додаток забезпечить широку підтримку мов та локальних авторів.

Amazon та Kobo пропонують конкурентоспроможні ціни та періодичні акції, тоді як Barnes & Noble також проводить акції, знижки. Новий додаток забезпечить вигідні пропозиції для споживачів.

Amazon та Kobo сприяють зменшенню використання паперу через поширення електронних видань. Barnes & Noble не акцентує на цьому аспекті. Новий додаток буде враховувати екологічні аспекти.

Ці висновки вказують на необхідність розробки нового інтерактивного програмного застосунку для онлайн-книгарні, який би поєднував переваги існуючих рішень та усував їх недоліки. Такий застосунок має забезпечити широкий асортимент, зручність користування, інноваційні інтерактивні функції та високу якість персоналізації для задоволення потреб сучасних користувачів.

1.3 Постановка завдань до роботи

Постановка завдань є одним з ключових етапів у розробці програмного забезпечення, оскільки вона визначає чіткі цілі та конкретні кроки, необхідні для досягнення успіху проекту.

Це дозволяє систематизувати процес розробки, забезпечити злагоджену роботу команди та ефективне використання ресурсів. Чітке визначення завдань дозволяє уникнути непорозумінь, зменшити ризики та забезпечити своєчасне виконання проекту.

Нижче наведено конкретні завдання, які необхідно вирішити для досягнення мети розробки інтерактивного програмного застосунку для онлайн-книгарні.

Метою бакалаврської дипломної роботи є розробка інтерактивного програмного застосунку для онлайн-книгарні, який поєднує в собі сучасні технології та підходи до розробки користувацьких інтерфейсів, що забезпечить більш ефективне та зручне користування системою порівняно з існуючими аналогами.

Для досягнення цієї мети необхідно вирішити наступні завдання:

– визначити основні функціональні вимоги, які повинні бути реалізовані у новому програмному застосунку;

– розробити загальну архітектуру системи, що включатиме всі модулі системи;

– створити базу даних для зберігання інформації про книги, користувачів, замовлення тощо;

– реалізувати механізми безпеки для захисту даних користувачів та запобігання несанкціонованому доступу;

– розробити модуль додавання книги у улюблене для підвищення юзабіліті інтерфейсу.

1.4 Порівняльний аналіз методів розв'язання поставлених завдань

У процесі розробки інтерактивного програмного застосунку для онлайн-книгарні було розглянуто кілька методів, кожен з яких має свої особливості, переваги та недоліки. Три основні методи, які було взято до уваги, включають використання готових платформ для створення онлайн-магазинів, розробку додатку за допомогою фреймворків для створення мобільних додатків, а також створення власного програмного забезпечення на Kotlin. У цьому розділі буде детально описано кожен із цих методів, після чого буде обрано найбільш підходящий для реалізації проекту.

Перший метод передбачає використання готових платформ для створення онлайн-магазинів, таких як Shopify, WooCommerce або Magento. Ці платформи надають широкий спектр інструментів та функцій, які дозволяють швидко і легко створювати інтернет-магазини з мінімальними знаннями програмування.

Shopify є однією з найпопулярніших платформ для створення онлайн-магазинів. Вона надає користувачам можливість створювати повністю функціональні інтернет-магазини з інтуїтивно зрозумілим інтерфейсом та багатим набором шаблонів. Shopify підтримує інтеграцію з різними платіжними системами, а також пропонує різноманітні маркетингові інструменти для залучення клієнтів [9]. Однак, використання Shopify пов'язане з регулярними витратами на підписку та додаткові функції.

WooCommerce є плагіном для WordPress, який дозволяє перетворити сайт на повноцінний інтернет-магазин [10]. WooCommerce надає багато безкоштовних та платних розширень, що дозволяють налаштувати магазин під конкретні потреби. Цей метод є економічно вигідним, проте потребує певних технічних знань для налаштування та підтримки.

Magento є потужною платформою для створення інтернет-магазинів, яка пропонує широкий спектр функцій для управління великими каталогами товарів [11]. Magento підходить для великих компаній та має високу гнучкість у налаштуваннях. Проте, використання Magento може вимагати значних технічних ресурсів та часу на налаштування і підтримку.

Другий метод полягає у використанні фреймворків для створення мобільних додатків, таких як React Native, Flutter або Xamarin. Ці фреймворки дозволяють розробляти кросплатформені додатки з використанням єдиного коду, що може значно скоротити час та витрати на розробку.

React Native, розроблений компанією Facebook, дозволяє створювати мобільні додатки для iOS та Android з використанням JavaScript. Цей фреймворк забезпечує високу продуктивність та надає можливість використовувати рідні компоненти для створення інтуїтивно зрозумілих інтерфейсів [12]. Основною перевагою React Native є велика спільнота розробників та багатий вибір бібліотек.

Flutter, розроблений компанією Google, дозволяє створювати кросплатформені додатки з використанням мови програмування Dart [13]. Flutter забезпечує високу продуктивність та підтримку інтерактивних елементів, що робить його ідеальним вибором для розробки складних та візуально привабливих додатків. Однак, Flutter є відносно новим фреймворком, і деякі бібліотеки та інструменти можуть бути менш зрілими порівняно з іншими фреймворками.

Xamarin, розроблений компанією Microsoft, дозволяє створювати кросплатформені додатки з використанням мови програмування C#. Xamarin забезпечує високу продуктивність та доступ до рідних API, що дозволяє створювати додатки з рідним виглядом та відчуттям [14]. Основною перевагою Xamarin є інтеграція з екосистемою Microsoft, що може бути корисним для компаній, які вже використовують технології Microsoft. Однак, використання Xamarin може вимагати додаткових витрат на ліцензії.

Третій метод передбачає створення власного програмного забезпечення з використанням мови програмування Kotlin [15]. Kotlin є сучасною мовою, розробленою компанією JetBrains, яка отримала офіційну підтримку від Google для розробки додатків під Android. Використання Kotlin для створення додатку забезпечує високу продуктивність, безпеку та зручність у написанні коду.

Однією з основних переваг Kotlin є його сумісність з Java, що дозволяє використовувати існуючі бібліотеки та інструменти для розробки додатків.

Kotlin має більш лаконічний синтаксис порівняно з Java, що зменшує кількість помилок та підвищує продуктивність розробників. Крім того, Kotlin підтримує функціональне програмування, що дозволяє створювати більш гнучкий та масштабований код.

Створення власного програмного забезпечення на Kotlin дозволяє повністю контролювати процес розробки та налаштування додатку під конкретні потреби. Це забезпечує високу гнучкість та можливість впровадження нових функцій та інтерактивних елементів, які недоступні в готових платформах та фреймворках. Однак, цей метод вимагає значних ресурсів та часу на розробку, а також високої кваліфікації розробників.

Розглянувши всі три методи, найбільш доцільним для реалізації проекту є створення власного програмного забезпечення на Kotlin. Цей метод надає декілька ключових переваг:

1. Створення власного ПЗ дозволяє контролювати кожен аспект додатку, від дизайну до функціональності, що забезпечує максимальну гнучкість та можливість швидкого реагування на потреби користувачів.
2. Використання Kotlin дозволяє інтегрувати новітні технології та підходи до розробки додатків, забезпечуючи високу продуктивність, безпеку та зручність користування.
3. Створення власного додатку дозволяє додавати унікальні функції, такі як персоналізовані рекомендації, інтерактивні елементи та синхронізація між пристроями, що підвищує задоволеність користувачів.
4. Власне ПЗ можна налаштувати під конкретні потреби онлайн-книгарні, забезпечуючи найкращий користувацький досвід та відповідність бізнес-вимогам.
5. Створення власного додатку зменшує залежність від зовнішніх платформ та їх обмежень, що забезпечує більшу незалежність та стабільність бізнесу.

Таким чином, створення власного програмного забезпечення на Kotlin є найкращим методом для розробки інтерактивного програмного застосунку для

онлайн-книгарні, що забезпечить конкурентні переваги на ринку та задовольнить потреби сучасних користувачів.

1.5 Висновок

У першому розділі було проведено комплексний аналіз актуальності онлайн-книгарень, їх сучасного стану та перспектив розвитку. Розглянуто основні фактори, що визначають значимість онлайн-книгарень у сучасному світі, зокрема зручність, доступність, широкий асортимент, конкурентоспроможні ціни, персоналізація рекомендацій та екологічність.

Також виконано порівняльний аналіз існуючих аналогів на ринку, таких як Amazon, Barnes & Noble та Kobo. Було виявлено, що кожен з цих застосунків має свої переваги та недоліки, однак жоден з них не поєднує всі необхідні функціональні можливості, які забезпечили б максимально ефективний та зручний користувацький досвід.

На основі цього аналізу було обґрунтовано необхідність розробки нового інтерактивного програмного застосунку для онлайн-книгарні, який враховує сучасні технологічні тренди та потреби користувачів. Розробка власного застосунку дозволяє впровадити інноваційні рішення, такі як персоналізовані рекомендації, інтерактивні елементи та високий рівень безпеки даних.

У розділі також було детально визначено завдання, які необхідно вирішити для досягнення мети роботи. Виконання завдань забезпечить створення високоякісного та конкурентоспроможного програмного застосунку для онлайн-книгарні.

2 АНАЛІЗ СТРУКТУРИ АЛГОРИТМІВ ТА ФОРМУВАННЯ ПРОБЛЕМАТИКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз та дослідження проблематики

Для успішної розробки інтерактивного програмного застосунку для онлайн-книгарні необхідно ретельно вивчити та проаналізувати наявні проблеми в цій галузі. Важливо зрозуміти основні виклики, з якими стикаються онлайн-книгарні, та виявити шляхи їх вирішення. Це включає всебічне дослідження технічних, функціональних та ринкових аспектів.

Перше, що потрібно ретельно проаналізувати, це функціональні труднощі, з якими можуть стикатися користувачі. Наприклад, нестабільна робота застосунку, недостатня зручність інтерфейсу чи обмежена функціональність можуть відлякати потенційних користувачів. Дослідження цих проблем дозволить виявити основні недоліки та пропонувати ефективні шляхи їх усунення.

Однією з ключових проблем є забезпечення зручного та інтуїтивно зрозумілого інтерфейсу користувача. Багато користувачів стикаються з труднощами під час навігації по сайту або пошуку конкретних книг. Це часто призводить до фрустрації та зниження задоволеності користувачів. Тому особлива увага має бути приділена зручності використання та швидкості доступу до потрібної інформації. Інтерфейс повинен бути адаптивним, відповідати різним типам пристроїв і підтримувати різноманітні мовні опції. Це забезпечить легкість у користуванні незалежно від пристрою або місця знаходження користувача.

Друге, важливо дослідити технічні аспекти. Нестабільна робота застосунку, довгий час завантаження сторінок або проблеми з безпекою даних можуть значно вплинути на задоволення користувачів. Тому вирішення технічних проблем є надзвичайно важливим етапом у розробці програмного продукту.

Важливою проблемою є технічна надійність та стабільність роботи

застосунку. Часті збої, довге завантаження сторінок та інші технічні недоліки можуть значно знизити задоволеність користувачів і призвести до втрати клієнтів. Тому під час розробки необхідно врахувати всі технічні аспекти, щоб забезпечити стабільну та безперебійну роботу онлайн-книгарні. Також важливо забезпечити можливість масштабування системи для обробки великої кількості одночасних запитів під час пікових навантажень. Регулярні оновлення та технічна підтримка мають бути пріоритетами для розробників.

Безпека даних є критичним аспектом для будь-якого онлайн-сервісу, особливо для тих, що обробляють фінансову інформацію. Онлайн-книгарні повинні забезпечити захист персональних даних користувачів та безпечність транзакцій. Це включає використання сучасних методів шифрування та захисту від кіберзагроз. Крім того, важливо впроваджувати регулярні оновлення безпеки та здійснювати аудит безпеки для виявлення та усунення потенційних вразливостей. Програми захисту даних повинні відповідати міжнародним стандартам та законодавчим вимогам.

Усі ці аспекти потребують ретельного дослідження та аналізу, щоб розробити інтерактивний програмний застосунок для онлайн-книгарні, який буде якісним, функціональним та конкурентоспроможним на ринку електронної комерції.

Конкурентне середовище є ще одним викликом для онлайн-книгарень. Існує велика кількість конкурентів, які пропонують схожі послуги, тому важливо знайти способи виділитися на ринку.

Це може включати унікальні функції, програми лояльності для клієнтів, персоналізовані рекомендації тощо. Наприклад, інтеграція з соціальними мережами, організація віртуальних книжкових клубів або ексклюзивні авторські заходи можуть стати конкурентними перевагами.

Крім того, важливо аналізувати ціни конкурентів та пропонувати конкурентоспроможні ціни, а також розширювати асортимент продукції, додаючи рідкісні та спеціалізовані видання. Також можна розглянути можливість співпраці з видавництвами для ексклюзивного продажу новинок.

Для більш глибокого розуміння проблематики важливо проаналізувати існуючі рішення на ринку. Це дозволить визначити їх сильні та слабкі сторони і на основі цього розробити власний продукт, який буде максимально відповідати потребам користувачів. Аналіз ринку також включає вивчення відгуків користувачів та їхніх побажань щодо функціональності та зручності використання. Вивчення нових технологій та тенденцій в області електронної комерції також може надати корисні інсайти для вдосконалення продукту.

Підсумовуючи, дослідження та аналіз проблематики є важливим етапом розробки інтерактивного програмного застосунку для онлайн-книгарні. Це дозволить не тільки виявити основні виклики, але й знайти ефективні шляхи їх подолання, що в свою чергу забезпечить успішність проекту.

2.2 Дослідження функціональних можливостей застосунку

Розробка інтерактивного програмного застосунку для онлайн-книгарні потребує ретельного дослідження функціональних можливостей, які забезпечать зручність та ефективність його використання. Основні функції застосунку мають бути орієнтовані на задоволення потреб користувачів, забезпечення зручної навігації та приємного досвіду користування.

Ось детальний опис функціональних можливостей, які будуть реалізовані в застосунку.

Однією з ключових функцій застосунку є можливість користувачів входити в свої облікові записи або реєструвати нові. Ця функція забезпечує персоналізацію досвіду користування, дозволяючи зберігати налаштування, переглядати історію покупок та улюблених книг. Для реєстрації нові користувачі вводять основні дані, такі як ім'я, електронна пошта та пароль. Верифікація електронної пошти може бути додатковим кроком для забезпечення безпеки облікового запису. Після успішної реєстрації користувачі можуть увійти в свій аккаунт, використовуючи електронну пошту та пароль.

Застосунок надає користувачам можливість переглядати книги за різними категоріями, такими як жанр, автор, новинки, бестселери тощо. Це дозволяє

користувачам швидко знаходити книги, які їх цікавлять. Інтерфейс категорій має бути інтуїтивно зрозумілим і зручним для навігації. Користувачі можуть фільтрувати результати пошуку за різними критеріями, що полегшує процес вибору книг.

При натисканні на обрану книгу, користувачі можуть переглянути детальну інформацію про неї. Ця інформація включає опис книги, інформацію про автора, рік видання, жанр, рейтинг та відгуки інших користувачів.

Детальна сторінка книги також може містити зображення обкладинки та можливість перегляду уривка книги. Це допомагає користувачам прийняти обґрунтоване рішення про покупку.

Користувачі можуть додавати книги до кошика, що дозволяє їм збирати книги для подальшої покупки. Функція кошика забезпечує зручне керування покупками: користувачі можуть переглядати всі обрані книги, змінювати кількість примірників, видаляти непотрібні книги, а також переходити до оформлення замовлення.

Кошик зберігає свій вміст навіть після виходу з облікового запису, що дозволяє користувачам повертатися до своїх покупок у зручний для них час.

Застосунок дозволяє користувачам додавати книги до списку улюблених. Ця функція корисна для зберігання книг, які користувачі хочуть придбати або прочитати в майбутньому.

Список улюблених книг доступний в особистому профілі користувача, що дозволяє швидко і легко знаходити та переглядати збережені книги.

Користувачі можуть переглядати та редагувати свій профіль. Профіль містить інформацію про користувача, включаючи ім'я, електронну пошту, історію покупок, списки улюблених книг та налаштування облікового запису. Можливість редагування профілю дозволяє користувачам змінювати свої персональні дані, паролі та інші налаштування, забезпечуючи персоналізований досвід користування.

Для забезпечення комфорту користувачів в різний час доби застосунок пропонує можливість зміни інтерфейсу зі світлої на нічну тему. Нічний режим

зменшує яскравість екрану та використовує темні кольори, що знижує навантаження на очі під час користування застосунком в умовах низької освітленості. Користувачі можуть перемикатися між темами вручну або налаштувати автоматичне переключення залежно від часу доби.

Ці функціональні можливості створюють інтерактивний, зручний та приємний досвід користування застосунком для онлайн-книгарні, що сприяє задоволенню потреб користувачів та ефективному управлінню їхніми книжковими колекціями та покупками.

2.3 Аналіз та розробка інтерактивних можливостей програмного застосунку

Розробка інтерактивних можливостей є ключовим елементом для створення ефективного та зручного програмного застосунку для онлайн-книгарні. Інтерактивні елементи забезпечують інтуїтивну навігацію, покращують користувацький досвід та підвищують загальну задоволеність користувачів. Впровадження інноваційних функцій та технологій стає найбільш необхідними для інтерактивного застосунку.

Основні інтерактивні можливості, які необхідно реалізувати в застосунку:

Тапання (Tap). Вибір книг та категорій: Користувач може тапнути на обкладинку книги, щоб переглянути деталі, або на категорію, щоб побачити всі доступні книги в цій категорії. Це спрощує навігацію та робить процес пошуку книг більш інтуїтивним.

Активні елементи інтерфейсу: Кнопки, посилання та інші інтерактивні елементи реагують на тапання, надаючи користувачам миттєвий відгук, що підтверджує виконання дії. Наприклад, при тапанні на кнопку "Купити" користувач побачить підтвердження додавання книги до кошика.

Прокручування (Scroll). Перегляд додаткового контенту: Прокручування дозволяє користувачам переглядати довгі списки книг, рекомендації, відгуки та інші елементи. Це забезпечує доступ до великого обсягу інформації без необхідності змінювати сторінки.

Динамічне завантаження контенту: При прокручуванні вниз можна автоматично завантажувати нові елементи, такі як нові книги або відгуки, що покращує користувацький досвід і зменшує час завантаження.

Мультитач-жести (Multitouch Gestures). Збільшення та зменшення масштабу: Користувачі можуть використовувати жест "пінчування", щоб збільшити або зменшити масштаб обкладинок книг або зображень, що покращує перегляд деталей.

Потягування (Swipe). Навігація по сторінках: Потягування дозволяє користувачам швидко переходити між сторінками, наприклад, з головної сторінки на сторінку категорій або на сторінку деталей книги.

Додавання книгу в обране за використанням трьох пальців. Користувач може додати книгу в улюблене вибравши відповідну книгу та потягнувши вверх трьома пальцями. Ця інтерактивність водночас відноситься як до мультитач-жестів так і до потягування.

Введення тексту (Text Input) Пошук книг: Користувачі можуть вводити текст для пошуку книг за назвою, автором або ключовими словами, що спрощує процес знаходження потрібних книг.

Вибір зі списку (Selection from List) Фільтрація та сортування: Користувачі можуть вибирати фільтри та сортування зі списку, щоб знайти книги за жанром, що полегшує процес пошуку.

Взаємодія з сповіщеннями (Notification Interaction) Оповіщення про знижки та акції: Користувачі можуть взаємодіяти зі сповіщеннями про знижки, акції або нові надходження, натискаючи на них, щоб перейти до відповідних розділів застосунку.

Розробка цих інтерактивних можливостей не лише покращує користувацький досвід, але й підвищує ефективність використання застосунку, роблячи його зручним, привабливим та функціональним для сучасних користувачів.

2.4 Розробка загальної моделі застосунку

У сучасній розробці програмного забезпечення UML (Unified Modeling Language) діаграми є невід'ємною частиною проектування складних систем. Для створення інтерактивного програмного застосунку для онлайн-книгарні використання UML-діаграм стає критично важливим інструментом, що забезпечує чітке уявлення про структуру, поведінку та взаємодію різних компонентів системи.

Візуалізація структури та архітектури системи є однією з основних переваг використання UML-діаграм. Класові діаграми дозволяють розробникам та аналітикам побачити структуру системи, зокрема її класи, атрибути, методи та взаємозв'язки між ними. Це є основою для створення об'єктно-орієнтованого дизайну, що сприяє кращому розумінню та управлінню складними проектами.

Аналіз поведінки системи також значно полегшується завдяки UML-діаграмам. Діаграми послідовностей та діаграми активностей відображають динамічний аспект системи, демонструючи взаємодію між об'єктами та послідовність операцій. Це особливо важливо для інтерактивних застосунків, де користувацький інтерфейс та бізнес-логіка тісно пов'язані.

UML-діаграми слугують універсальною мовою, зрозумілою всім учасникам проекту: від розробників до менеджерів і замовників. Це забезпечує ефективне спілкування та зменшує ризик виникнення непорозумінь. Спрощення комунікації між учасниками проекту є ще однією важливою перевагою, яка дозволяє уникати багатьох проблем на етапі розробки та впровадження.

На рисунку 2.1 наведено розроблену UML-діаграму для розроблюваного інтерактивного програмного застосунку.

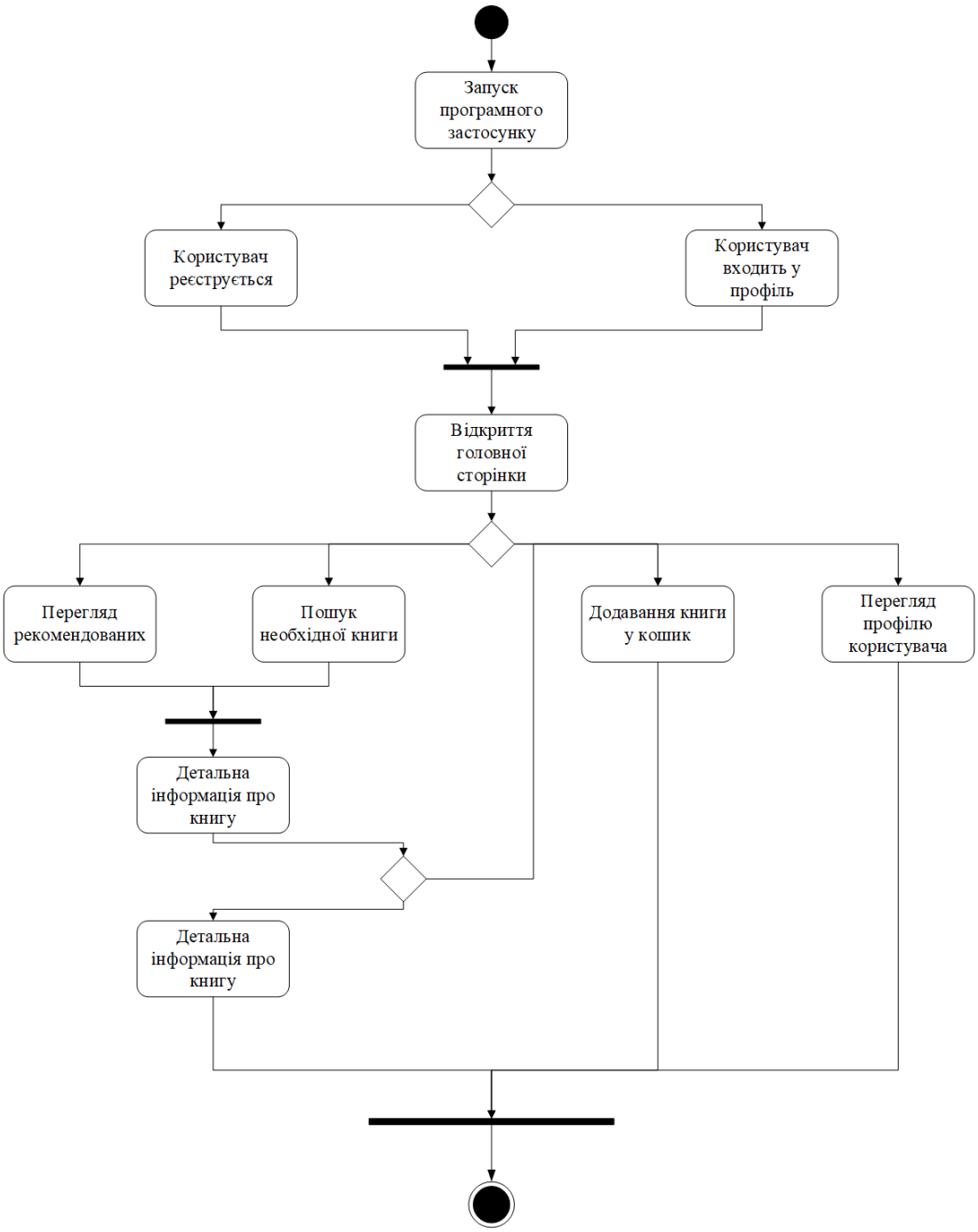

Рисунок 2.1 – UML-діаграма активності застосунку

Процес тестування та валідації системи також стає більш ефективним завдяки UML-діаграмам. Діаграми станів дозволяють детально моделювати різні стани об'єктів у системі та переходи між ними, що допомагає у визначенні тестових сценаріїв та перевірці коректності поведінки системи в різних умовах. Це значно підвищує якість кінцевого продукту та його відповідність початковим вимогам.

Документування системи за допомогою UML-діаграм виконує роль детальної технічної документації, яка може бути використана як під час розробки, так і після завершення проекту. Це значно полегшує процес підтримки та модифікації застосунку в майбутньому, забезпечуючи збереження знань про систему та її структуру.

Таким чином, використання UML-діаграм у проектуванні інтерактивного програмного застосунку для онлайн-книгарні не лише покращує загальне розуміння системи, але й сприяє ефективній організації роботи, забезпечуючи чіткість, надійність та масштабованість розробки. Це дозволяє створювати якісне програмне забезпечення, що відповідає потребам користувачів та вимогам ринку.

2.5 Дослідження алгоритмів роботи системи

У процесі розробки інтерактивного програмного застосунку для онлайн-книгарні важливе місце займає дослідження алгоритмів роботи системи. Це дослідження дозволяє забезпечити коректне функціонування різних модулів, оптимізувати продуктивність і забезпечити високу якість користувацького досвіду. Важливість дослідження алгоритмів полягає в тому, що воно дає змогу зрозуміти логіку роботи системи, виявити можливі помилки та недоліки, а також знайти шляхи їх усунення до етапу впровадження.

Одним із ключових модулів онлайн-книгарні є модуль реєстрації користувачів. Важливо дослідити алгоритм цього модуля для забезпечення безпеки та простоти реєстраційного процесу. На рисунку 2.2 наведено блок-схему цього алгоритму.

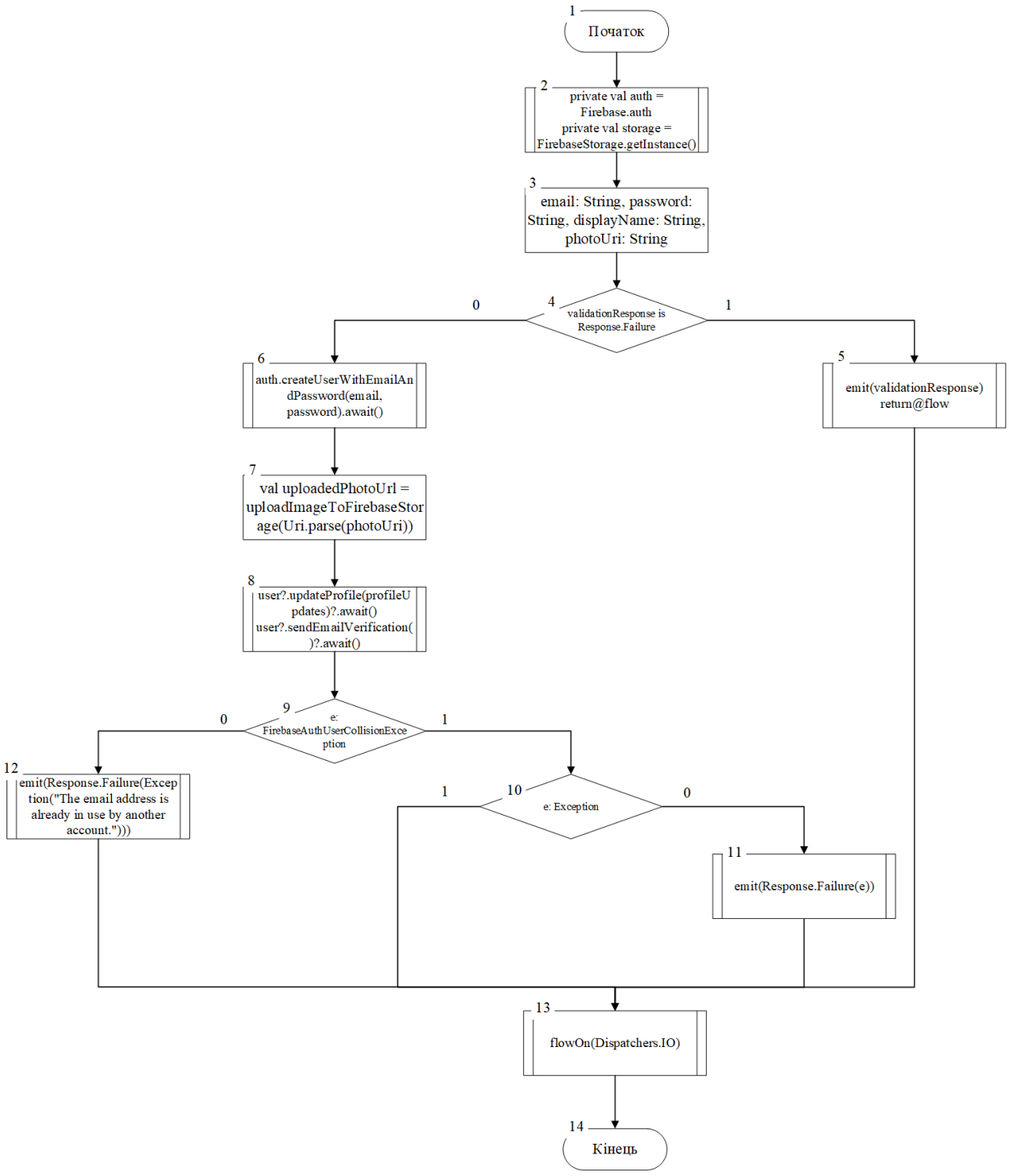

Рисунок 2.2 – Блок-схема алгоритму реєстрації користувача

Алгоритм має враховувати перевірку введених даних, уникнення дублікатів, забезпечення надійного захисту особистих даних користувачів за допомогою шифрування та використання надійних протоколів. Крім того, необхідно передбачити функціональність для підтвердження електронної пошти та відновлення паролю, що підвищить зручність і безпеку для користувачів.

Модуль додавання книг в улюблене є важливою частиною функціоналу онлайн-книгарні, оскільки він покращує користувацький досвід, дозволяючи зберігати обрані книги для подальшого перегляду або покупки. На рисунку 2.3 наведено блок-схему цього алгоритму.

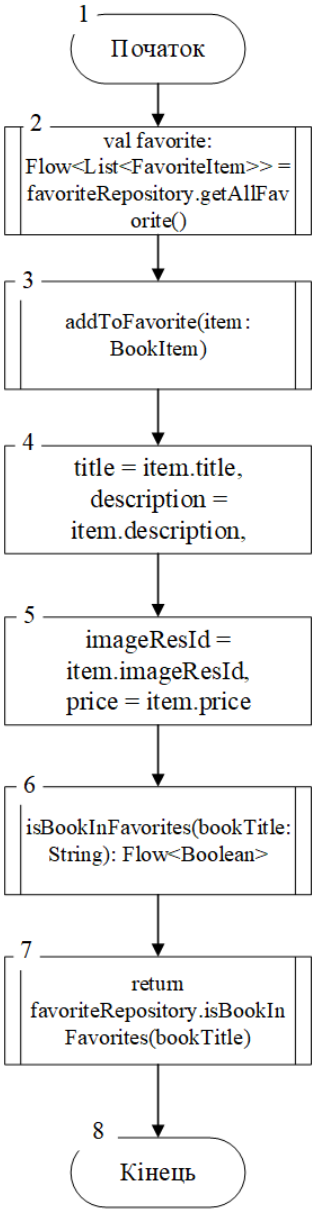

Рисунок 2.3 – Блок-схема модулю для додавання книги в улюблені

Алгоритм цього модуля має забезпечити ефективне зберігання і швидкий доступ до списку улюблених книг. Важливо оптимізувати процес додавання та видалення книг зі списку, а також синхронізацію з обліковим записом

користувача, щоб інформація була доступна з різних пристроях. На рисунку 2.4 наведено блок-схему цього модуля.

Модуль оформлення покупки є центральним елементом системи, оскільки саме він забезпечує завершення комерційних транзакцій.

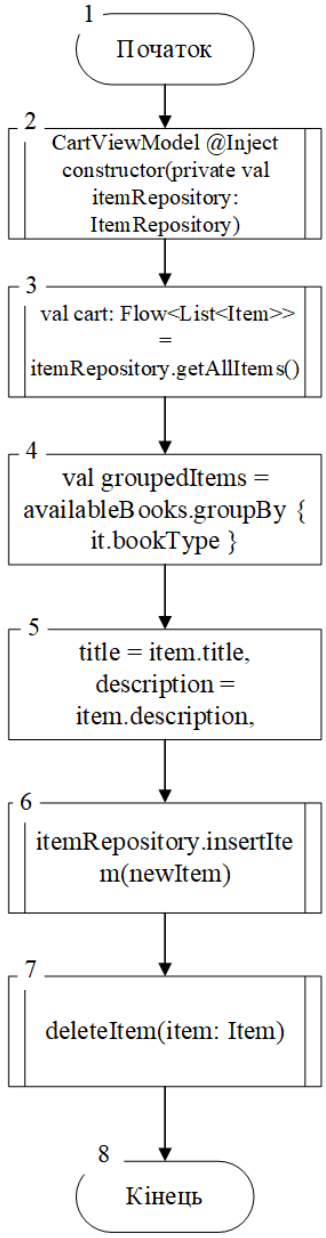

Рисунок 2.4 – Модуль оформлення купівлі книги

Алгоритм роботи цього модуля повинен враховувати обробку кошика, перевірку наявності товару на складі, розрахунок вартості з урахуванням знижок та податків, а також обробку платежів.

Безпека є ключовим аспектом цього модуля, тому необхідно інтегрувати надійні платіжні шлюзи і забезпечити шифрування фінансових даних. Крім того, алгоритм повинен включати відправлення підтвердження замовлення на електронну пошту користувача і оновлення статусу замовлення в обліковому записі.

Дослідження алгоритмів для модулів реєстрації, додавання в улюблене та оформлення покупки є критично важливим етапом у розробці інтерактивного програмного застосунку для онлайн-книгарні. Воно дозволяє забезпечити коректне функціонування системи, оптимізувати користувацький досвід та гарантувати безпеку даних. Це, у свою чергу, сприяє підвищенню довіри користувачів до системи та її успішному функціонуванню на ринку.

2.5 Висновки

У другому розділі було розроблено модель роботи за допомогою UML-діаграм, які відображають основні процеси та взаємозв'язки в програмі. Це включало моделювання ключових елементів системи, таких як реєстрація користувачів, додавання книг до улюблених та оформлення покупки.

Окрім цього, у розділі були створені основні алгоритми для різних модулів системи. Для модуля реєстрації був розроблений алгоритм, що забезпечує безпечну і зручну процедуру реєстрації, включаючи перевірку введених даних та захист особистої інформації. Для модуля додавання книг в улюблене було створено алгоритм, який оптимізує процес зберігання та доступу до списку улюблених книг користувача, з урахуванням синхронізації даних між різними пристроями.

Для модуля оформлення покупки було розроблено алгоритм, що забезпечує коректну обробку замовлень, включаючи перевірку наявності товару, розрахунок вартості та обробку платежів з високим рівнем безпеки.

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ ІНТЕРАКТИВНОГО ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ ОНЛАЙН-КНИГАРНІ

3.1 Обґрунтування вибору засобів для реалізації програмного застосунку

При виборі мови програмування для розробки інтерактивного програмного застосунку для онлайн-книгарні було розглянуто кілька популярних мов, таких як Java, JavaScript і Swift. Кожна з них має свої переваги і недоліки, однак після детального аналізу було вирішено обрати Kotlin. Нижче наведено порівняння розглянутих мов програмування.

Java є однією з найпопулярніших мов програмування, що має велику кількість бібліотек і фреймворків. Вона традиційно використовується для розробки Android-застосунків, що робить її знайомою для багатьох розробників мобільних додатків [16].

Java також добре підходить для створення масштабованих додатків завдяки своїй продуктивності та стабільності. Однак, Java не має багатьох сучасних функцій, які є у Kotlin, що може ускладнити розробку більш гнучких і зручних інтерфейсів. Крім того, програмування на Java часто вимагає написання більшої кількості коду, що може призвести до збільшення часу розробки та підтримки.

JavaScript є універсальною мовою програмування, яка використовується як для фронтенду, так і для бекенду (з використанням Node.js), що дозволяє створювати повноцінні веб-додатки. JavaScript має величезну спільноту розробників та широкий вибір бібліотек і фреймворків, таких як React, Angular і Vue.js [17]. Мова надає велику гнучкість та динамічні можливості для розробки інтерфейсів.

Однак, JavaScript не завжди забезпечує високу продуктивність порівняно з мовами, які працюють на JVM, що може бути критичним для великих застосунків. Крім того, JavaScript менш безпечний у порівнянні з мовами, які мають сувору типізацію і механізми контролю доступу, що може підвищувати

ризик вразливостей.

Swift є сучасною мовою програмування з багатьма корисними функціями для розробки мобільних додатків. Swift забезпечує високу продуктивність і оптимізований для розробки під iOS [18].

Мова розроблена для легкого засвоєння і надає чистий і зрозумілий синтаксис. Однак, Swift використовується переважно для розробки додатків під iOS, що робить його менш універсальним для кросплатформної розробки.

Хоча Swift популярний серед розробників iOS, його використання поза цією платформою є обмеженим.

Kotlin включає багато сучасних функцій, таких як безпечність від NullPointerException, розширення функцій, корутини для асинхронного програмування, що робить його ідеальним для розробки мобільних додатків.

Kotlin без проблем взаємодіє з Java, що дозволяє використовувати наявні Java-бібліотеки та інфраструктуру [19].

Програмування на Kotlin відзначається меншою кількістю коду, що значно спрощує розробку та підтримку застосунків. Це велика перевага, особливо в умовах швидкозмінного індустріального середовища. Оскільки Kotlin офіційно підтримується для розробки Android-застосунків, розробники можуть бути впевнені у його актуальності та регулярних оновленнях, що сприяє підтримці сучасних стандартів і технологій.

Незважаючи на те, що Kotlin є відносно молодою мовою порівняно з Java, його зростаюча популярність і широке прийняття в спільноті розробників свідчать про його успішну інтеграцію в індустріальне середовище програмування. Крім того, активна підтримка та розвиток Kotlin сприяють поширенню його використання в різних галузях програмування.

Всі результати порівняння між Kotlin і іншими мовами програмування можна знайти в таблиці 3.1, яка надає наочне уявлення про аналітичні дані та переваги Kotlin у порівнянні з іншими мовами програмування.

Таблиця 3.1 – Порівняльна таблиця

Характеристики	**Java**	**JavaScript**	**Swift**	**Kotlin**
Сучасні можливості	+/–	+/–	+	+
Взаємодія з Java	+	–	–	+
Обсяг коду	–	+	+	+
Офіційна підтримка Android	+	-	–	+
Універсальність	+/–	+	–	+
Продуктивність	+	–	+	+
Стабільність	+	–	+	+
Гнучкість розробки інтерфейсів	–	+	+	+

Сучасні можливості мов програмування визначають їх здатність підтримувати новітні парадигми та інструменти, які полегшують розробку, тестування і підтримку програмного забезпечення. Kotlin пропонує такі сучасні функції, як безпечність від NullPointerException, корутини для асинхронного програмування і розширення функцій, що значно покращує ефективність розробників. Впровадження цих можливостей дозволяє створювати більш надійні та масштабовані додатки з меншими витратами часу та зусиль.

Взаємодія з Java є важливою характеристикою, оскільки велика кількість існуючого програмного забезпечення і бібліотек написані на Java. Kotlin, будучи повністю сумісним з Java, дозволяє розробникам легко інтегрувати наявні Java-бібліотеки та код у нові проекти, зменшуючи витрати на переписування та забезпечуючи безперервність роботи. Це робить Kotlin привабливим вибором для розробників, які бажають використовувати сучасні можливості без втрати доступу до перевірених інструментів Java.

Менший обсяг коду сприяє зниженню складності програмного забезпечення та зменшенню кількості помилок. Kotlin дозволяє писати менш громіздкий і більш читабельний код порівняно з Java, завдяки використанню

лаконічних синтаксичних конструкцій та функцій, таких як data-класи і лямбда-вирази. Це прискорює процес розробки, зменшує навантаження на підтримку коду і покращує загальну якість програмного забезпечення.

Офіційна підтримка Android від Google забезпечує регулярні оновлення, підтримку і документацію для мови Kotlin. Це гарантує, що розробники мають доступ до найновіших інструментів і ресурсів для створення Android-додатків. Така підтримка також забезпечує більшу стабільність і довгострокову перспективу для використання Kotlin у мобільній розробці, що робить його стратегічно важливим вибором для проектів, орієнтованих на Android-платформу.

Універсальність мови програмування дозволяє використовувати її для різних типів проектів і платформ. Kotlin підтримує розробку для Android, серверів, веб-додатків та навіть iOS через Kotlin Multiplatform. Це робить Kotlin дуже гнучким і дозволяє розробникам використовувати одну мову для різних проектів, зменшуючи витрати на навчання і підтримку різних мов програмування.

Продуктивність мови програмування визначає її здатність виконувати код швидко і ефективно. Kotlin компілюється в байт-код, який виконується на JVM, забезпечуючи високу продуктивність і оптимізацію. Це особливо важливо для мобільних додатків, де обмежені ресурси пристрою вимагають максимальної ефективності від програмного забезпечення.

Стабільність програмного забезпечення є критично важливою для забезпечення надійної роботи додатків. Kotlin, завдяки своїй сумісності з Java і підтримці від Google, забезпечує високу стабільність і надійність для мобільних додатків. Це дозволяє розробникам зосередитися на функціональності, а не на усуненні неочікуваних збоїв або помилок.

Гнучкість у розробці користувацьких інтерфейсів дозволяє створювати привабливі, зручні та інтуїтивно зрозумілі додатки. Kotlin пропонує сучасні інструменти та бібліотеки для розробки інтерфейсів, які спрощують роботу з UI-елементами і забезпечують кращий користувацький досвід. Це включає

підтримку DSL (Domain Specific Language) для декларативного опису UI, що робить процес розробки більш природним і ефективним.

Для зберігання та управління даними було вирішено використати базу даних – Firebase**.**

Firebase – це платформа для розробки мобільних та веб-додатків, яка надає різноманітні сервіси та інструменти для зручної роботи з різними аспектами додатків, такими як зберігання даних, аутентифікація користувачів, аналітика, хмарні функції. Firebase забезпечує комплексний підхід до розробки, дозволяючи зосередитися на функціоналі застосунку, а не на інфраструктурних питаннях. Завдяки своїй гнучкості та потужним інструментам, Firebase значно спрощує процес створення високоякісних, надійних та масштабованих застосунків.

Основні переваги використання:

1. Простота інтеграції та швидкість розробки:

- Швидке розгортання: Firebase пропонує готові рішення для аутентифікації, баз даних, хостингу тощо, що дозволяє швидко створити повнофункціональний додаток.
- Мінімальний код: Завдяки SDK та API, багато завдань можна виконати з мінімальною кількістю коду.

2. Аутентифікація та управління користувачами:

- Firebase Authentication: Підтримка різних методів аутентифікації (електронна пошта, соціальні мережі тощо), що забезпечує безпеку та зручність для користувачів.
- Управління користувачами: Легке управління обліковими записами користувачів, включаючи відновлення паролів, верифікацію електронної пошти та інші функції.

3. Зберігання та керування файлами

- Firebase Storage: Безпечне зберігання книг, обкладинок, аудіофайлів та інших медіафайлів з можливістю швидкого доступу та завантаження.

4. Аналітика та відстеження користувацької поведінки

- Google Analytics for Firebase: Вбудована аналітика, що дозволяє відстежувати, як користувачі взаємодіють з бібліотекою, які книги популярні, які функції використовуються найчастіше тощо.

Також для аутентифікації користувача при реєстрації для безпеки та надійності був вибраний популярний сервіс SendGrid - це популярний сервіс для відправки електронних листів, що пропонує надійне рішення для транзакційних та маркетингових розсилок. Він забезпечує простий API для інтеграції, а також ряд інструментів для управління та аналітики розсилок.

1. Надійність та масштабованість.

- Висока доставляємість: SendGrid має високу репутацію серед провайдерів електронної пошти, що сприяє високому рівню доставляємості листів до скриньок отримувачів.
- Масштабованість: Підходить для компаній будь-якого розміру, від стартапів до великих підприємств, завдяки можливості обробляти велику кількість листів без зниження продуктивності.

2. Простота інтеграції.

- SMTP: Можливість використання SMTP для відправки листів безпосередньо з існуючих додатків.

3. Функціональність та інструменти.

- Динамічний контент: Підтримка персоналізованого контенту для кожного отримувача.
- Шаблони електронних листів: Можливість створення та використання готових шаблонів листів.
- Аналітика та звітність: Детальна аналітика про доставку, відкриття, кліки, відписки тощо.

4. Безпека.

- Автентифікація та шифрування: Підтримка SPF, DKIM та DMARC для захисту домену від спаму та фішингу.

З огляду на переваги та недоліки розглянутих мов програмування, було

прийнято рішення використовувати Kotlin для розробки інтерактивного програмного застосунку, Firebase для зберігання даних та SendGrid для авторизації користувача. Kotlin поєднує сучасні можливості, високу продуктивність і зручність розробки, що робить його найкращим вибором для реалізації даного проекту.

3.2 Обґрунтування вибору середовища розробки програмного застосунку

Розробка інтерактивного програмного застосунку для онлайн-книгарні потребує обдуманого вибору середовища розробки, що відповідає специфіці проекту та задачам розробки. Під час аналізу можна розглянути такі аспекти для кожного із можливих середовищ:

1) IntelliJ IDEA – це потужне інтегроване середовище розробки, яке володіє широким набором інструментів для підтримки Kotlin [17]. IntelliJ IDEA забезпечує зручне автодоповнення коду, інтегровані інструменти аналізу та рефакторингу, що полегшує розробку високоякісного програмного забезпечення.

2) Eclipse є одним з найбільш відомих середовищ розробки з великою спільнотою розробників. Додавши плагін Kotlin до Eclipse, можна розробляти програми на Kotlin безпосередньо у цьому середовищі, використовуючи звичні інтерфейси та інструменти [18].

3) Visual Studio Code є легким і розширюваним середовищем розробки з великим вибором розширень для різних мов програмування. Встановлення розширення Kotlin Extension Pack додає підтримку Kotlin до VS Code, забезпечуючи зручне середовище для розробки програмного забезпечення.

Усі згадані середовища розробки були об'єктом аналізу, результати якого були внесені до таблиці 3.2. У цій таблиці проведено порівняння основних характеристик кожного середовища, таких як підтримка Kotlin, наявність спеціалізованих інструментів розробки для Android-додатків, рівень зручності та розширюваність, наявність документації та спільноти розробників.

Таблиця 3.2 – Порівняльна таблиця аналогів середовищ розробки

Характеристика	**IntelliJ IDEA**	**Eclipse з Kotlin Plugin**	**Visual Studio Code з Kotlin Extension Pack**	**Android Studio**
Підтримка Kotlin	+	+	+	+
Спеціалізовані інструменти розробки Android	–	–	–	+
Рівень зручності та розширюваність	+	+/–	+	+
Наявність документації та спільноти розробників	+	+/–	+	+

Після аналізу можна зробити висновок, що для розробки Android-додатків, таких як інтерактивна програма для онлайн-книгарні, оптимальним вибором є Android Studio. Android Studio має спеціалізовану підтримку для розробки Android-додатків, включаючи інтегровані інструменти для розробки, тестування та налагодження, що спрощує процес розробки та підвищує ефективність. Таким чином, обираючи Android Studio, розробник отримує ідеальне середовище для реалізації проекту з врахуванням його особливостей і потреб.

3.3 Розробка модулю для реєстрації або входу у профіль користувача

Модуль для реєстрації та входу у профіль користувача в інтерактивному програмному застосунку для онлайн-книгарні реалізований за допомогою Firebase Authentication та Firebase Storage. Він забезпечує функціонал реєстрації нового користувача, входу в систему, оновлення профілю, верифікації електронної пошти та виходу з профілю, використовуючи корутіни та потоки для асинхронних операцій.

Функція registerUser відповідає за реєстрацію нового користувача.

Спочатку вона перевіряє правильність введених електронної адреси та паролю за допомогою функції validateEmailAndPassword.

Після успішної валідації користувач реєструється за допомогою auth.createUserWithEmailAndPassword. Далі фотографія користувача завантажується до Firebase Storage за допомогою функції uploadImageToFirebaseStorage.

Потім оновлюються дані профілю (ім'я та фотографія) за допомогою UserProfileChangeRequest, і відправляється лист для верифікації електронної пошти за допомогою user.sendEmailVerification.

Після успішної реєстрації актуальні дані користувача отримуються за допомогою getUserDataFromFirebase, і результат операції повертається у вигляді Response.Success або Response.Failure.

Функція loginUser відповідає за вхід користувача в систему. Вона також починається з валідації введених даних за допомогою validateEmailAndPassword. Після успішної валідації виконується вхід за допомогою auth.signInWithEmailAndPassword.

Після успішного входу актуальні дані користувача отримуються за допомогою getUserDataFromFirebase, і результат операції повертається у вигляді Response.Success або Response.Failure.

Функція uploadImageToFirebaseStorage відповідає за завантаження фотографії користувача до Firebase Storage.

Вона створює посилання на файл у Firebase Storage за допомогою storage.reference, завантажує файл за допомогою imageRef.putFile і отримує URL завантаженого файлу за допомогою imageRef.downloadUrl.

Функція getUserDataFromFirebase отримує актуальні дані користувача з Firebase Authentication. Вона використовує auth.currentUser для отримання поточного користувача і отримує необхідні дані, такі як UID, електронна адреса, ім'я користувача, URL фотографії та статус верифікації електронної пошти. Отримані дані повертаються у вигляді об'єкта User.

Функція updateUserProfile дозволяє оновити ім'я користувача та

фотографію профілю. Вона перевіряє введене ім'я користувача, отримує поточні дані користувача за допомогою getUserDataFromFirebase, видаляє попередню фотографію з Firebase Storage за допомогою deleteImageFromFirebaseStorage, завантажує нову фотографію за допомогою uploadImageToFirebaseStorage і оновлює профіль користувача за допомогою UserProfileChangeRequest. Після успішного оновлення дані користувача повертаються у вигляді Response.Success.

Функція sendEmailVerification відповідає за відправку листа для верифікації електронної пошти. Вона перевіряє, чи електронна пошта вже верифікована, і якщо ні, відправляє лист для верифікації за допомогою user.sendEmailVerification. Функція markEmailAsVerified перевіряє та оновлює статус верифікації, перевіряючи статус верифікації та оновлюючи дані користувача після верифікації.

Функція logout відповідає за вихід користувача з системи. Вона виконує вихід за допомогою auth.signOut і повертає результат операції у вигляді Response.Success або Response.Failure.

Цей модуль забезпечує безпечну та ефективну роботу з обліковими записами користувачів, використовуючи переваги Firebase Authentication та Firebase Storage для зберігання і обробки даних.

3.4 Розробка модулю для додавання книги у список улюблених

Модуль для додавання книги у список улюблених в інтерактивному програмному застосунку для онлайн-книгарні реалізований у вигляді ViewModel, який забезпечує інтерактивну та асинхронну роботу з даними користувача. Цей модуль використовує MutableStateFlow та StateFlow для керування станами і відображенням даних у реальному часі.

Модуль починається з класу UserDataViewModel, який ініціалізується за допомогою ін'єкції залежностей. Він містить кілька MutableStateFlow та StateFlow для зберігання та спостереження за станами користувача, повідомленнями про помилки та успішність операцій. Наприклад, _user та user зберігають дані користувача, _profileError та profileError використовуються для

повідомлення про помилки, а _isLoading та isLoading сигналізують про стан завантаження.

Функція getUserData викликається для отримання даних користувача з authRepository. Вона запускає корутину у viewModelScope, яка викликає метод authRepository.getUserData. Якщо операція успішна, дані користувача оновлюються у _user і перевіряється статус верифікації електронної пошти.

Функція markEmailAsVerified відповідає за верифікацію електронної пошти користувача. Вона встановлює _isLoading у true, щоб відобразити стан завантаження, і запускає корутину для виклику методу authRepository.markEmailAsVerified.

У випадку успіху встановлюється відповідне повідомлення у _profileSuccess, а _isEmailVerified змінюється на true. У випадку невдачі _profileError зберігає повідомлення про помилку.

Функція setUser дозволяє встановити дані користувача у _user. Це може бути корисним для локального оновлення даних без взаємодії з віддаленим репозиторієм.

Функція updateUserProfile дозволяє оновити профіль користувача, включаючи його ім'я та фотографію. Вона встановлює _isLoading у true і запускає корутину, яка викликає метод authRepository.updateUserProfile.

Якщо операція успішна, локальні дані користувача оновлюються у _user, і встановлюється повідомлення про успіх у _editProfileSuccess. Після затримки у дві секунди для відображення повідомлення користувач перенаправляється назад до профілю за допомогою navController.popBackStack. У випадку невдачі _editProfileError зберігає повідомлення про помилку.

Функція clearMessages очищує всі повідомлення про помилки та успіхи, встановлюючи відповідні поля у null.

Функція logout відповідає за вихід користувача з системи. Вона запускає корутину, яка викликає метод authRepository.logout, а потім очищує дані користувача у _user.

Цей модуль забезпечує інтерактивну та асинхронну роботу з даними

користувача, використовуючи MutableStateFlow та StateFlow для динамічного оновлення інтерфейсу користувача у реальному часі. Він також забезпечує обробку помилок та відображення повідомлень про успішність операцій, що робить його ефективним інструментом для управління профілем користувача у програмному застосунку для онлайн-книгарні.

3.5 Висновок

У третьому розділі було детально розглянуто технічні аспекти розробки інтерактивного програмного застосунку для онлайн-книгарні. Обґрунтування вибору технологій і засобів розробки є критичним етапом, що визначає ефективність і надійність кінцевого продукту.

Було розглянуто вибір мови програмування Kotlin. Kotlin є сучасною мовою, що забезпечує високий рівень безпеки, продуктивності та зручності для розробників. Його сумісність із Java і підтримка Android робить його ідеальним вибором для розробки мобільних додатків.

Обґрунтовано вибір середовища розробки Android Studio. Це середовище пропонує потужні інструменти для розробки, налагодження та тестування додатків на платформі Android. Його інтеграція з інструментами Google і підтримка сучасних технологій робить процес розробки більш ефективним та інтуїтивно зрозумілим.

Було розроблено модуль для реєстрації та входу у профіль користувача. Використання Firebase Authentication забезпечує надійне та безпечне управління користувацькими даними, включаючи реєстрацію, вхід, верифікацію електронної пошти та оновлення профілю. Модуль реалізовано з використанням корутин і потоків для асинхронної обробки, що покращує продуктивність і користувацький досвід.

Розроблено модуль для додавання книг у список улюблених. Цей модуль інтегрує функціонал керування улюбленими книгами в користувацькому профілі, забезпечуючи зручність і персоналізацію для користувачів. Модуль використовує ті ж технології, що і модуль для реєстрації та входу, що забезпечує

консистентність і надійність роботи застосунку.

Таким чином, у третьому розділі було здійснено важливі кроки для створення надійного та зручного програмного застосунку для онлайн-книгарні, використовуючи сучасні технології та підходи до розробки програмного забезпечення. Це дозволяє забезпечити високий рівень функціональності, безпеки та користувацького досвіду, що є ключовими аспектами для успішного впровадження і використання застосунку.

4 ТЕСТУВАННЯ ІНТЕРАКРИВНИГО ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ ОНЛАЙН-КНИГАРНІ

4.1 Тестування програмного застосунку

Тестування програмного застосунку є невід'ємною частиною процесу розробки, спрямованою на перевірку його функціональності, надійності та відповідності вимогам [19]. У цьому підрозділі буде детально розглянуто план тестування, методи тестування та використані інструменти для забезпечення якості програмного забезпечення розробленого інтерактивного програмного застосунку для онлайн-книгарні.

Під час опису тестування буде звернута увага на ключові функціональність програмного застосунку, такі як реєстрація та вхід користувача, додавання книг у список улюблених, а також взаємодія з базою даних та іншими сервісами. Тестові випадки будуть розглянуті з точки зору їх покриття та ефективності виявлення помилок.

Початок тестування програмного застосунку передбачає аналіз етапу входу у систему. На початковому екрані, який представлений на рисунку 4.1, користувачеві відображається форма реєстрації. На цьому екрані присутнє вітальне повідомлення, яке запрошує користувача ввести свої дані для реєстрації в системі.

Тестування цього етапу включає перевірку кількох ключових аспектів. По-перше, перевіряється коректність відображення елементів інтерфейсу, таких як поля для введення даних та кнопка реєстрації. Далі, проводиться аналіз правильності текстових повідомлень, що включають в себе вітальне повідомлення та вказівки до полів вводу.

Після цього тестується можливість користувача вводити свої дані у відповідні поля. Окрема увага приділяється валідації введених даних, так як вона повинна гарантувати правильність формату електронної пошти та довжини паролю.

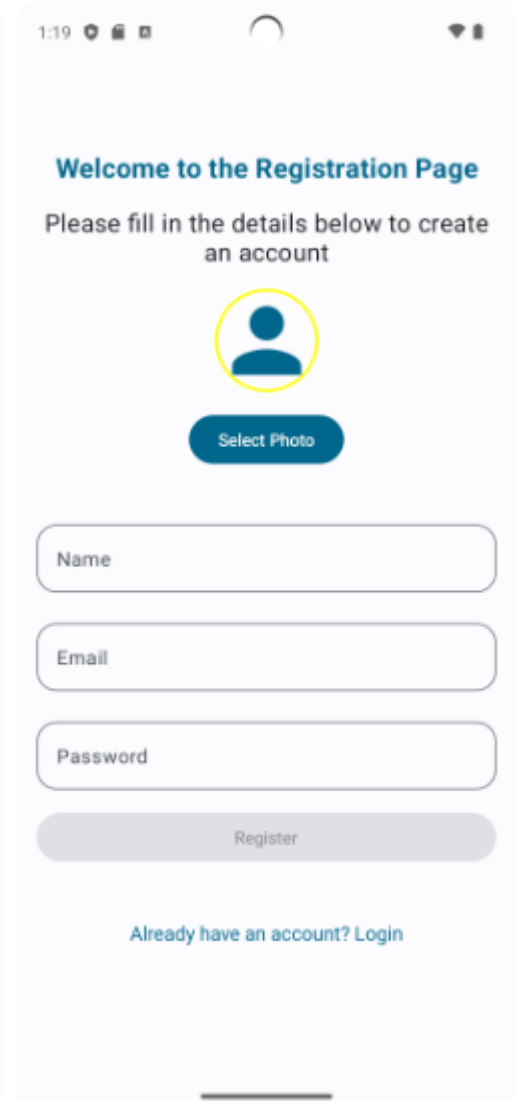

Рисунок 4.1 – Вікно реєстрації нового користувача

Останнім етапом цього тестування є перевірка взаємодії з кнопкою реєстрації. Важливо, щоб програмний застосунок реагував на натискання кнопки коректно, провіряючи введені дані та передаючи їх на сервер для подальшої обробки.

Цей етап тестування є важливим для забезпечення коректної роботи форми реєстрації та відображення інформації для користувача, що створює позитивний користувацький досвід під час взаємодії з програмним застосунком.

Далі у послідовності тестування перевіряється вікно авторизації, зображене на рисунку 4.2. Це вікно містить два поля для введення імені та паролю, а також кнопку для входу у акаунт.

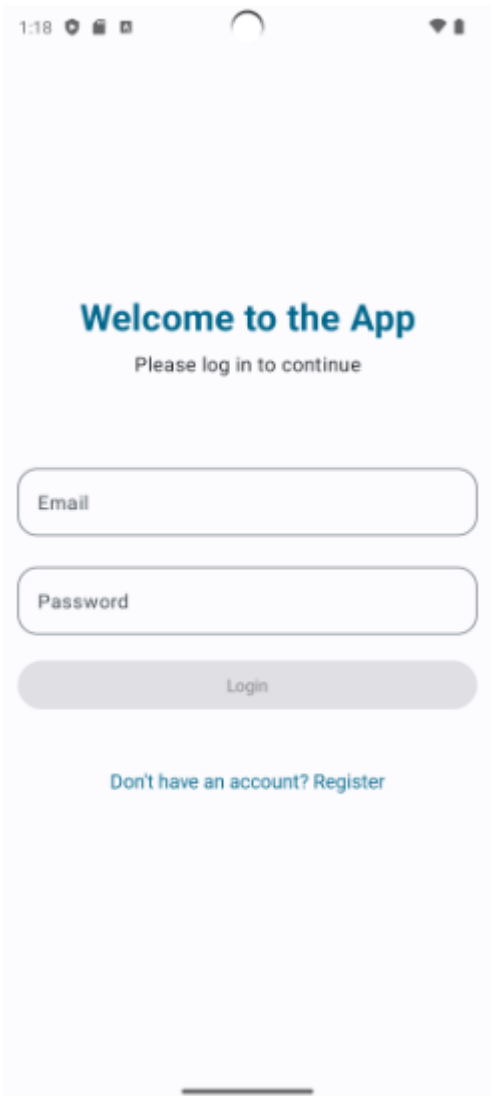

Рисунок 4.2 – Вікно авторизації у розробленому програмному застосунку

Тестування цього етапу включає перевірку:

1. Відображення всіх необхідних елементів інтерфейсу, зокрема полів введення та кнопки входу.
2. Можливість користувача вводити свої дані в поля для імені та паролю.
3. Коректність реакції на натискання кнопки входу, включаючи валідацію введених даних та передачу їх на сервер для авторизації.
4. Забезпечення безпеки та конфіденційності під час введення пароля.

Перевірка правильності роботи вікна авторизації важлива для забезпечення доступу користувача до його особистого облікового запису у системі.

Після успішної авторизації користувача перевіряється відображення

інтерфейсу головної сторінки, яка представлена на рисунку 4.3. Інтерфейс головної сторінки містить такі елементи як список книг, рекомендації, кнопки навігації і т.д..

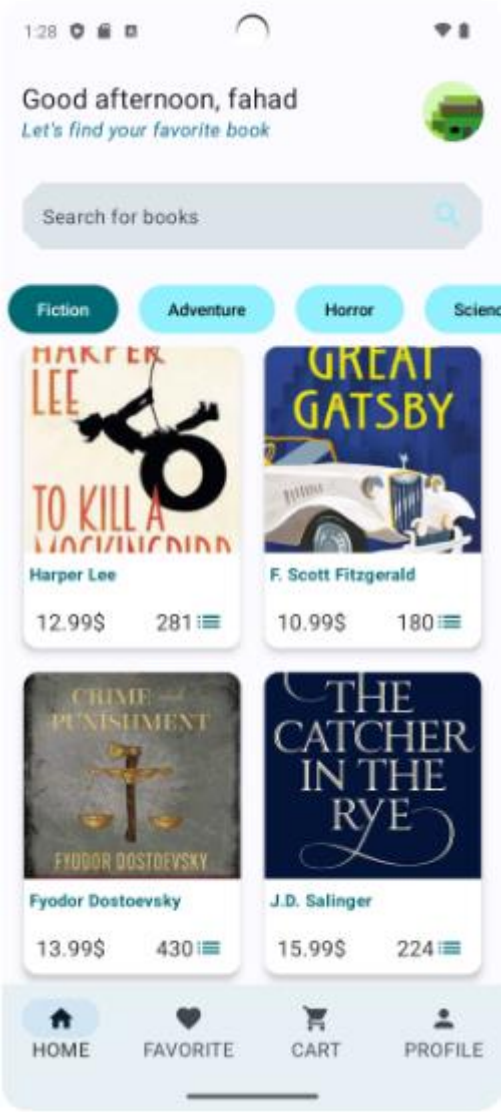

Рисунок 4.3 – Відображення головної сторінки додатку

Особлива увага приділяється забезпеченню зручного та інтуїтивно зрозумілого користувацького досвіду на головній сторінці програмного застосунку.

Після успішного входу в систему та навігації до головної сторінки, користувач може обрати конкретну книгу для перегляду більш детальної інформації про неї, як зображено на рисунку 4.4. Цей етап тестування

спрямований на перевірку коректності відображення та функціональності інтерфейсу детальної інформації про книгу.

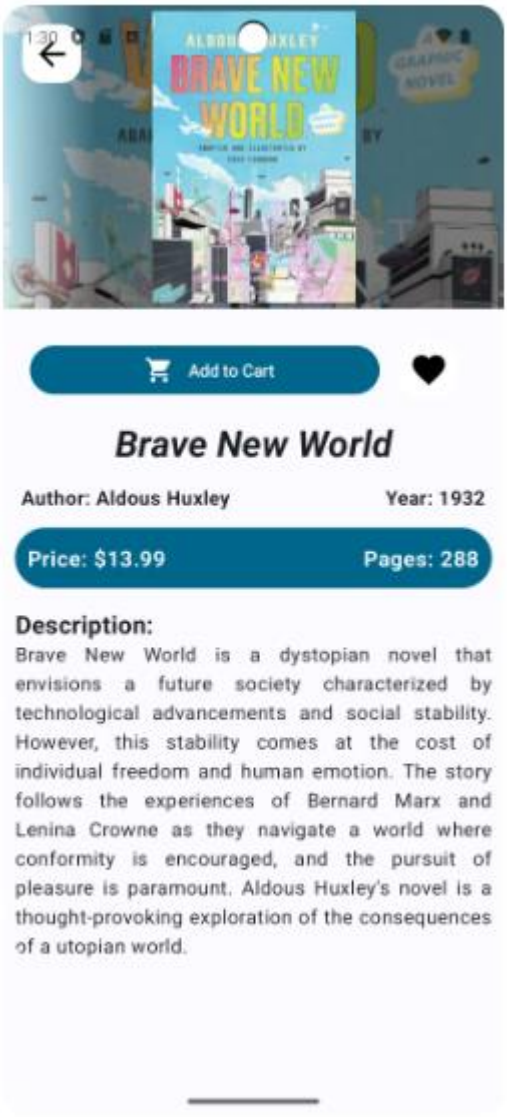

Рисунок 4.4 – Вигляд інтерфейсу під час вибору конкретної книги

Цей етап тестування важливий для забезпечення зручного та інформативного інтерфейсу для користувачів, які шукають детальну інформацію про книги та мають намір часто користуватися цим функціоналом.

Після ознайомлення з детальною інформацією про книгу, користувач може вирішити придбати її. Для цього в системі передбачено корзину, яка допомагає користувачеві здійснити покупку, як показано на рисунку 4.5. Важливим етапом тестування є перевірка коректності відображення цін для уникнення можливих

проблем під час покупки.

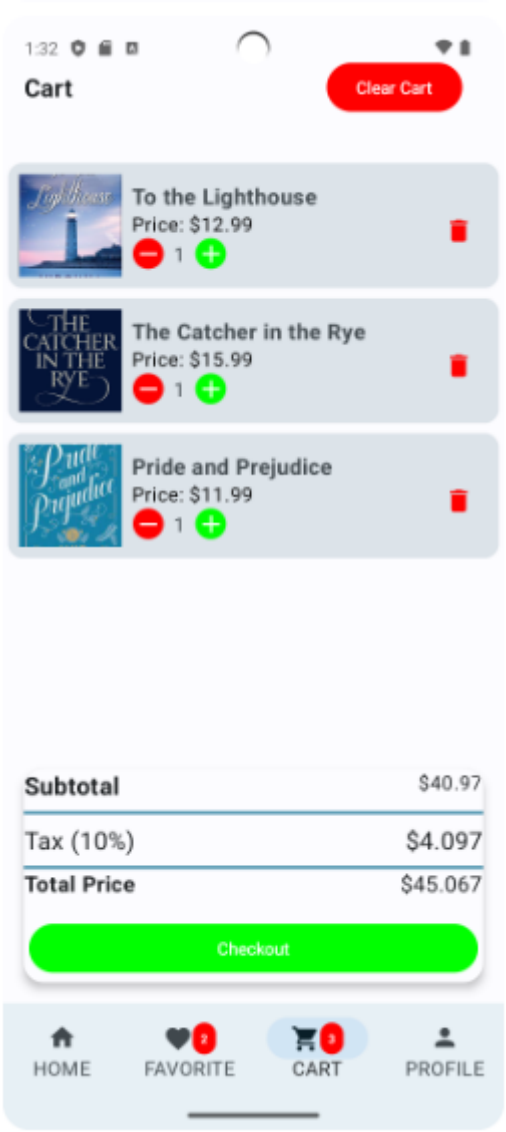

Рисунок 4.5 – Відображення вкладки «Корзини»

Цей етап тестування має велике значення для забезпечення зручності та безпеки покупки для користувача, а також для запобігання можливих непорозумінь чи проблем у процесі здійснення покупки.

Після огляду інформації про книгу користувач може вирішити додати її у свій список улюблених. Для цього в системі передбачено відповідну вкладку, яка дозволяє користувачеві зберігати певні книги для майбутнього перегляду або покупки, як показано на рисунку 4.6.

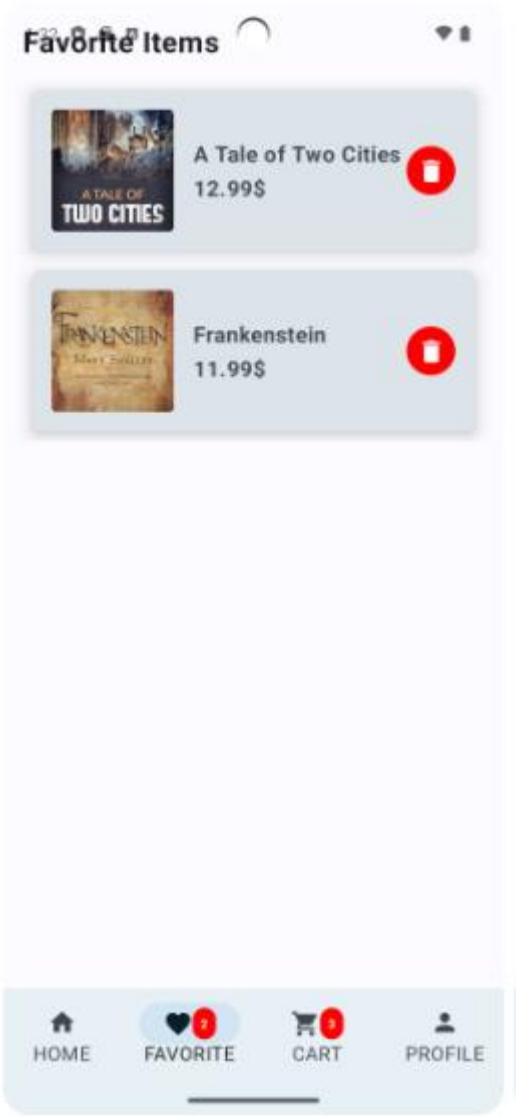

Рисунок 4.6 – Вигляд списку улюблених книг

Функціонал додавання книги у список улюблених дуже важливий для зручності користувача і забезпечення його задоволення від використання програмного застосунку. Цей етап тестування має велике значення для забезпечення зручності використання програмного застосунку користувачем та підтримки його інтересів у виборі та зберіганні книг для подальшого перегляду або придбання.

Після виконання різноманітних дій у програмному застосунку, таких як реєстрація, авторизація, додавання книг у список улюблених та здійснення покупок, користувач може бажати переглянути свій власний профіль для отримання загальної інформації про свої дії та налаштування. В профілі

відображено інформація про користувача, така як ім'я, адреса електронної пошти, статус підтвердження електронної пошти, список улюблених книг і т.д.. На рисунку 4.7 наведено приклад того, як виглядає ця частина інтерфейсу в остаточному програмному застосунку.

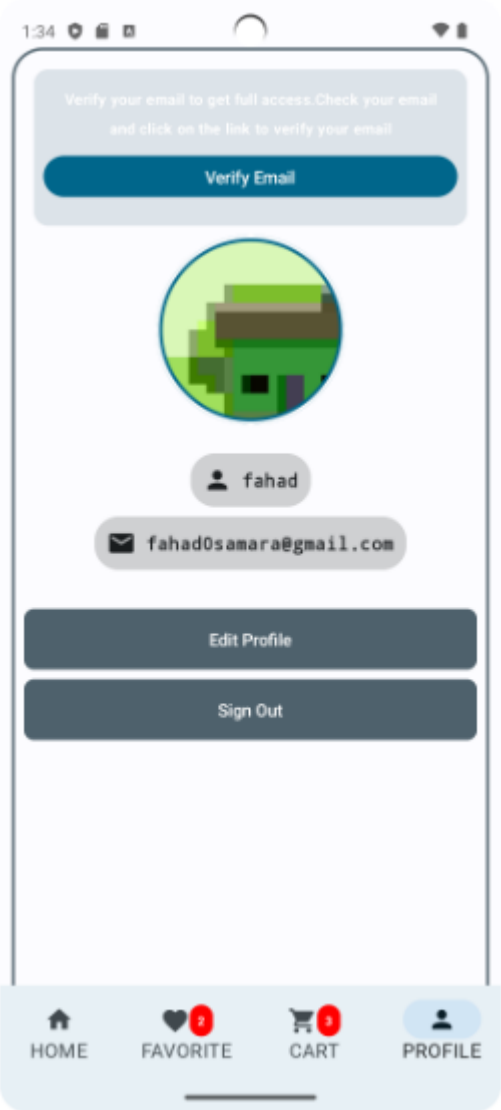

Рисунок 4.7 – Приклад відображення профілю користувача

Важливим етапом тестування є перевірка коректності відображення та функціональності профілю користувача. Останнім функціоналом, що буде протестовано буде відображення інтерфейсу у нічному форматі. На рисунку 4.8 відображено головну сторінку у нічному форматі інтерфейсу.

Рисунок 4.8 – Відображення інтерфейсу головного меню в нічному режимі

Підтримка нічного режиму є важливою функцією для забезпечення комфортного використання програмного застосунку в будь-який час доби, особливо в умовах недостатнього освітлення або для користувачів, які віддають перевагу темному дизайну інтерфейсу. На рисунку 4.9 показано вигляд вкладки "Корзина" у нічному режимі.

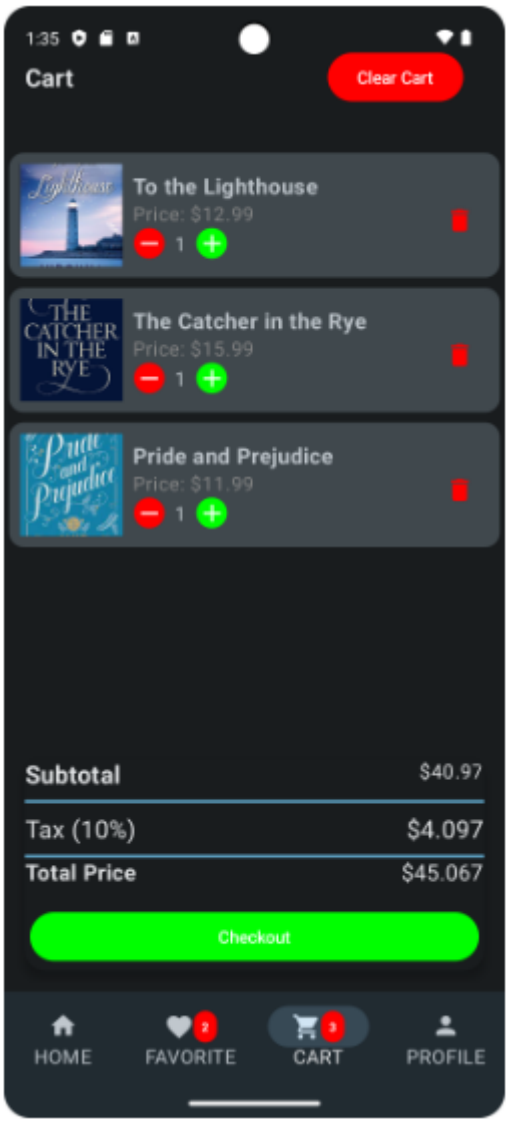

Рисунок 4.9 – Нічний режим у вкладці «Cart»

У цьому підрозділі було описано процес тестування різних функціональних можливостей програмного застосунку. Тестування включало в себе перевірку правильності відображення інтерфейсу, функціональності різних елементів та їх реакцію на дії користувача. Важливим аспектом було також перевірка роботи в нічному режимі, що є важливою функцією для забезпечення комфортного користування в будь-який час доби.

В результаті успішного тестування було підтверджено правильну роботу функціоналу та забезпечено якість та зручність користування програмою для кінцевих користувачів.

4.2 Створення інструкції для користування

Мобільний додаток для онлайн-книгарні розроблений з урахуванням потреб користувачів, пропонуючи зручний та інтуїтивний інтерфейс. Починаючи роботу з додатком, користувач повинен зареєструватися або увійти в свій профіль. Це дозволяє отримати доступ до персоналізованих рекомендацій та зберігати налаштування.

Після успішного входу, користувач може переглядати рекомендовані книги, які формуються на основі його попередніх переглядів та купівель. Для зручності пошуку потрібної книги передбачено функцію пошуку, яка дозволяє швидко знаходити необхідні видання за ключовими словами, авторами або жанрами.

При виборі книги користувач може ознайомитися з детальною інформацією про неї, включаючи опис, відгуки інших читачів, рейтинги та інформацію про автора. Це допомагає прийняти зважене рішення перед покупкою або додаванням книги до улюблених.

Якщо книга сподобалася, користувач може додати її до списку улюблених, що дозволяє зберегти її для майбутнього перегляду або придбання. Для безпосередньої покупки передбачена функція додавання книги до корзини. Після завершення вибору книг користувач може перейти до оформлення замовлення.

Додаток також пропонує можливість змінювати вигляд інтерфейсу на нічний режим. Це забезпечує комфортне читання та перегляд книг у темний час доби, зменшуючи навантаження на очі.

Перегляд профілю дозволяє користувачу контролювати свої особисті дані, переглядати історію покупок, налаштовувати параметри додатка та отримувати інформацію про накопичені бонуси або знижки. Такий функціонал робить користування додатком максимально зручним та приємним, забезпечуючи користувачів всім необхідним для комфортного шопінгу та читання.

4.3 Визначення технічних вимог

Визначення технічних вимог є критично важливим етапом у процесі розробки програмного продукту [20]. Це дозволяє забезпечити стабільну та ефективну роботу додатку на різних пристроях, задовольняючи потреби користувачів. Формування вимог гарантує, що всі компоненти системи функціонуватимуть гармонійно, мінімізуючи ризики збоїв та проблем під час експлуатації. Завдяки чіткому визначенню технічних параметрів, розробники можуть оптимізувати ресурсоспоживання додатку, а користувачі – насолоджуватися плавною роботою без зайвих затримок чи помилок. Це підвищує загальну якість програмного продукту, роблячи його конкурентоспроможним на ринку.

Нижче у таблицях 4.1 та 4.2 наведено мінімальні та рекомендовані технічні вимоги для роботи додатку онлайн-книгарні.

Таблиця 4.1 – Мінімальні технічні вимоги

Параметр	**Вимога**
Тип процесора	Двоядерний процесор 1.5 ГГц
Об’єм оперативної пам’яті	2 ГБ
Місце на жорсткому диску	100 МБ
Операційна система	Android 5.0 / iOS 10.0
Швидкість інтернету	1 Мбіт/с

Таблиця 4.2 – Рекомендовані технічні вимоги

Параметр	**Вимога**
Тип процесора	Чотириядерний процесор 2.0 ГГц
Об’єм оперативної пам’яті	4 ГБ
Місце на жорсткому диску	200 МБ
Операційна система	Android 9.0 / iOS 13.0
Швидкість інтернету	5 Мбіт/с

Вибір процесора є критичним для забезпечення швидкої та стабільної роботи додатку. Мінімальні вимоги включають двоядерний процесор 1.5 ГГц, що достатньо для базових операцій, таких як перегляд книг та пошук. Рекомендовані вимоги передбачають використання чотириядерного процесора 2.0 ГГц, який забезпечить більш плавну роботу при взаємодії з великим обсягом даних та під час багатозадачності.

Оперативна пам’ять впливає на здатність додатку працювати без затримок. Мінімальний обсяг у 2 ГБ дозволяє додатку функціонувати на базовому рівні, тоді як 4 ГБ оперативної пам’яті в рекомендованих вимогах забезпечують кращу продуктивність та швидкість реакції на дії користувача.

Розмір додатку є важливим для зберігання даних і швидкого доступу до них. Мінімальні вимоги вказують на 100 МБ, що достатньо для основних функцій додатку, включаючи зберігання інформації про книги та користувачів. Рекомендовані 200 МБ забезпечують додатковий простір для зберігання кешу та інших тимчасових даних, що покращує загальний досвід користування.

4.4 Висновок

У четвертому розділі детально розглянули процес розробки інтерактивного програмного застосунку для онлайн-книгарні, приділивши увагу створенню інструкції для користувачів, формуванню технічних вимог. Ці кроки є невід'ємними для забезпечення високої якості та надійності кінцевого продукту.

Окрім цього, у розділі розглянуто процес тестування програмного застосунку, що є важливим етапом для виявлення та усунення можливих помилок. Тестування гарантує, що додаток відповідає всім встановленим технічним вимогам та забезпечує високу якість користувацького досвіду.

ВИСНОВКИ

У бакалаврській дипломній роботі було розглянуто розробку інтерактивного програмного застосунку для онлайн-книгарні, починаючи від формулювання завдань і аналізу актуальності до тестування та визначення технічних вимог. Комплексний підхід до кожного етапу дозволив створити функціональний та зручний у використанні продукт, який відповідає сучасним вимогам користувачів і ринку.

У першому розділі здійснено аналіз актуальності онлайн-книгарень, що підтвердило зростаючий попит на цифрові ресурси для покупки книг. Порівняльний аналіз існуючих аналогів виявив недоліки та переваги, що дало змогу точно визначити завдання для розробки унікального і конкурентоспроможного продукту.

Другий розділ присвячений аналізу структури алгоритмів та формуванню проблематики програмного забезпечення. Дослідження функціональних можливостей та розробка загальної моделі застосунку допомогли створити міцний фундамент для подальшої розробки. Вивчення алгоритмів роботи системи дозволило забезпечити її ефективність і стабільність.

У третьому розділі було описано розробку програмних модулів інтерактивного застосунку. Обґрунтування вибору засобів та середовища для реалізації програмного продукту забезпечили оптимальні умови для його створення. Реалізація ключових модулів, таких як модуль для реєстрації або входу у профіль користувача та модуль для додавання книги у список улюблених, забезпечила основну функціональність додатку.

Четвертий розділ описує тестування інтерактивного програмного застосунку, що дозволило виявити та усунути можливі помилки. Створення інструкції для користування допомогло зробити додаток зручним та інтуїтивно зрозумілим для користувачів. Визначення технічних вимог забезпечило сумісність додатку з різними пристроями та операційними системами, що сприяє його широкому використанню.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Розвиток інформаційних технологій в Україні [Електронний ресурс] – Режим доступу до ресурсу: https://sendpulse.ua/knowledge-base/chatbot/telegram/create-telegram-chatbot.

2. Платформи для онлайн-продажу книг [Електронний ресурс] – Режим доступу до ресурсу: https://fliphtml5.com/learning-center/uk/top-8-best-websites-to-sell-ebooks-online-for-digital-content-creators/.

3. Розвиток інформаційних технологій та їх перспективи [Електронний ресурс] – Режим доступу до ресурсу: https://posibniki.com.ua/post-elektronnii-biznes-yak-napryam-rozvitku.

4. Бірюк Н.Ф. Простір онлайн-бібліотеки, як середовище соціокультурної взаємодії, Вінниця, 2024. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/21193.

5. Онлайн-книгарні та їх актуальність [Електронний ресурс] – Режим доступу до ресурсу: https://iprosvita.com/chomu-v-eru-hadzhetiv-neobkhidno-chytaty-knyhy/.

6. Amazon – ринок майбутнього [Електронний ресурс] – Режим доступу до ресурсу: https://business.diia.gov.ua/consult/market-analysis.

7. Найбільша книгарня світу Barnes & Noble [Електронний ресурс] – Режим доступу до ресурсу: https://dostavkain.com/ua/barnesandnoble.com.

8. Kobo – це? [Електронний ресурс] – Режим доступу до ресурсу: https://portal-express.com.ua/shop/kobo-books.

9. Shopify, як платформа для створення власного інтернет магазину. [Електронний ресурс] – Режим доступу до ресурсу: https://xn-----6kcbb4cegbzednvr1ak3exe8ipar.in.ua/blogs/online-store-success-business/what-is-shopify.

10. WooCommerce, як плагін для електронної комерції [Електронний ресурс] – Режим доступу до ресурсу: https://woocommerce.com/.

11. Magento платформа з відкритим кодом [Електронний ресурс] – Режим доступу до ресурсу: https://business.adobe.com/products/magento/magento-commerce.html.

12. React Native Navigation [Електронний ресурс] – Режим доступу до ресурсу: https://reactnavigation.org/.

13. Flutter – Build apps for any screen [Електронний ресурс] – Режим доступу до ресурсу: https://flutter.dev/.

14. Xamarin – американська компанія в галузі розробки ПЗ [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Xamarin.

15. Kotlin, як мова програмування [Електронний ресурс] – Режим доступу до ресурсу: https://kotlinlang.org/.

16. Java [Електронний ресурс] – Режим доступу до ресурсу: https://www.java.com/.

17. IntelliJ IDEA –інтегроване середовище розробки [Електронний ресурс] – Режим доступу до : https://w3schoolsua.github.io/hyperskill/ide.html#gsc.tab=0.

18. Eclipse – середовище розробки [Електронний ресурс] – Режим доступу до ресурсу: https://javarush.com/ua/groups/posts/uk.2359.eclipse-java-seredovijshe-rozrobki-pd-sebe.

19. Kotlin [Електронний ресурс] – Режим доступу до ресурсу: https://lemon.school/blog/mova-programuvannya-kotlin

20. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 121 «Інженерія програмного забезпечення» / уклад. : О. Н. Романюк, Г. О. Черноволик. – Вінниця : ВНТУ, 2022. – 51 с.

ДОДАТКИ

Додаток А. Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор
_____ О. Н. Романюк
«12» березня 2024 р.

Технічне завдання
на бакалаврську дипломну роботу «Розробка інтерактивного програмного застосунку для онлайн-книгарні» за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
_____ д.т.н., доцент Н.П. Бабюк
«12» березня 2024 р.
Виконав:
_____ студент гр. ЗП-20б Н.Ф. Бірюк
«12» березня 2024 р.

Вінниця – 2024 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка інтерактивного програмного застосунку для онлайн-книгарні».

Галузь застосування – мобільний застосунок.

2. Підстава для розробки.

Підставою для виконання бакалаврської дипломної роботи (БДР) є індивідуальне завдання на БДР та наказ №80 від «11» березня 2024 р. ректора ВНТУ про закріплення тем БДР.

3. Мета та призначення розробки.

Метою бакалаврської дипломної роботи є розробка інтерактивного програмного застосунку для онлайн-книгарні, який забезпечить зручне користування, ефективну організацію продажів та підвищення рівня задоволеності клієнтів.

Призначення роботи – розробка та реалізація інтерактивного програмного застосунку для онлайн-книгарні.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БДР:

1. Бірюк Н.Ф. Простір онлайн-бібліотеки, як середовище соціокультурної взаємодії, Вінниця, 2024. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/21193.

2. Онлайн-книгарні та їх актуальність [Електронний ресурс] – Режим доступу до ресурсу: https://iprosvita.com/chomu-v-eru-hadzhetiv-neobkhidno-chytaty-knyhy/.

3. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 121 «Інженерія програмного забезпечення» / уклад. : О. Н. Романюк, Г. О. Черноволик. – Вінниця : ВНТУ, 2022. – 51 с.

5. Технічні вимоги

Тип програми – мобільний застосунок; модель розробки – ітеративна; засоби зберігання даних – Firebase; вхідні дані: персональні дані реєстрації користувача, вподобоння користувача; вихідні дані: замовлення та оформлення користувачем покупку; середовище розробки – Android Studio; мова програмування – Kotlin; програмне забезпечення для симуляції мобільних пристроїв – Android Emulator.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

1. Пояснювальна записка до БДР.
2. Технічне завдання.
3. Лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Аналіз стану онлайн-книгарень у сучасному світу	14.03.24 –30.03.24
2	Аналіз аналогів інтерактивного програмного застосунку онлайн-книгарні	01.04.24 – 15.04.24
3	Розробка алгоритмів застосунку	16.04.24 – 20.04.24
4	Розробка модулів програмного застосунку	21.04.24 – 15.05.24
5	Тестування програмного застосунку	16.05.24 – 22.05.24
6	Оформлення матеріалів до захисту БДР	22.05.24 – 03.06.24

10. Порядок контролю та прийняття

Виконання етапів бакалаврської дипломної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття бакалаврської дипломної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

Додаток Б – Протокол перевірки на плагіат

ПРОТОКОЛ
ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка інтерактивного програмного застосунку для онлайн-книгарні

Тип роботи: БДР

Підрозділ: кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Романюк О.Н.

Оригінальність	**97,8**
Схожість	2,2

Аналіз звіту подібності

▪ **Запозичення, виявлені у роботі, оформленні коректно і не містять ознак плагіату.**

☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цілісності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховання недобросовісних запозичень.

Особа, відповідальна за перевірку — Черноволик Г.О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи — Бірюк Н.Ф.

Керівник роботи — Бабюк Н.П.

Додаток В. Лістинг програми

AuthRepositoryImpl.kt

```
package com.fahad.list_food.data.repository

import android.net.Uri
import android.util.Log
import com.fahad.auth_firebase.domain.model.Response
import com.fahad.auth_firebase.domain.model.User
import com.fahad.auth_firebase.domain.repository.AuthRepository
import com.fahad.auth_firebase.util.valid.validateEmailAndPassword
import com.google.firebase.Firebase
import com.google.firebase.auth.FirebaseAuthUserCollisionException
import com.google.firebase.auth.UserProfileChangeRequest
import com.google.firebase.auth.auth
import com.google.firebase.storage.FirebaseStorage
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.delay
import kotlinx.coroutines.flow.Flow
import kotlinx.coroutines.flow.flow
import kotlinx.coroutines.flow.flowOn

import kotlinx.coroutines.flow.single
import kotlinx.coroutines.tasks.await
import javax.inject.Inject
import javax.inject.Singleton

@Singleton
class AuthRepositoryImpl @Inject constructor() : AuthRepository {
    private val auth = Firebase.auth
```

```
    private val storage = FirebaseStorage.getInstance()

    override suspend fun registerUser(
        email: String, password: String, displayName: String, photoUri: String
    ): Flow<Response<User>> = flow {
        val validationResponse = validateEmailAndPassword(email, password)
        if (validationResponse is Response.Failure) {
            emit(validationResponse)
            return@flow
        }
        try {
            // Register the user with email and password
            auth.createUserWithEmailAndPassword(email, password).await()

            // Upload the image to Firebase Storage
            val uploadedPhotoUrl = uploadImageToFirebaseStorage(Uri.parse(photoUri))
            // Send email verification


            // After successful registration, set the display name and photo URL
            val user = auth.currentUser
            val profileUpdates = UserProfileChangeRequest.Builder().setDisplayName(displayName)
                .setPhotoUri(Uri.parse(uploadedPhotoUrl)).build()
            user?.updateProfile(profileUpdates)?.await()
            user?.sendEmailVerification()?.await()
            // Fetch the latest user data from Firebase
            val updatedUser = getUserDataFromFirebase()
            emit(Response.Success(updatedUser))
```

```
            } catch (e: FirebaseAuthUserCollisionException) {
                emit(Response.Failure(Exception("The email address is already in use by another account.")))
            } catch (e: Exception) {
                emit(Response.Failure(e))
            }
        }.flowOn(Dispatchers.IO)


        override suspend fun loginUser(
            email: String, password: String,
        ): Flow<Response<User>> = flow {
            val validationResponse = validateEmailAndPassword(email, password)
            if (validationResponse is Response.Failure) {
                emit(validationResponse)
                return@flow
            }

            try {
                auth.signInWithEmailAndPassword(email, password).await()

                // Fetch the latest user data from Firebase
                val updatedUser = getUserDataFromFirebase()



                emit(Response.Success(updatedUser))
            } catch (e: Exception) {
                emit(Response.Failure(e))
            }
```

```
        }.flowOn(Dispatchers.IO)


        private suspend fun uploadImageToFirebaseStorage(photoUri: Uri): String {
            return try {
                // Create a reference to the image file in Firebase Storage
                val storageRef = storage.reference
                val imageRef = storageRef.child("profile_images/${photoUri.lastPathSegment}")

                // Upload the image to Firebase Storage
                imageRef.putFile(photoUri).await()

                // Get the download URL
                val downloadUrl = imageRef.downloadUrl.await()

                downloadUrl.toString()
            } catch (e: Exception) {
                // Handle the exception, e.g., log it or throw a custom exception
                Log.e("AuthRepositoryImpl", "Error uploading image to Firebase Storage: ${e.message}")
                ""
            }
        }

        private fun getUserDataFromFirebase(): User {
            val firebaseUser = auth.currentUser
            val uid = firebaseUser?.uid ?: throw Exception("User is not logged in")

            val email = firebaseUser.email ?: ""
```

```
            val displayName = firebaseUser.displayName ?: ""
            val photoUrl = downloadImageFromFirebaseStorage(firebaseUser.photoUrl?.toString() ?: "")
            val isEmailVerified = firebaseUser.isEmailVerified

            return User(uid, email, displayName, photoUrl, isEmailVerified)
        }


        private fun downloadImageFromFirebaseStorage(photoUrl: String?): String {
            return try {
                // Ensure that the photoUrl is a full download URL
                if (photoUrl != null && photoUrl.startsWith("https://")) {
                    // If it's a Firebase Storage URL, use it directly
                    photoUrl

                } else {
                    // Handle other cases or log an error
                    Log.e("AuthRepositoryImpl", "Invalid photoUrl format: $photoUrl")
                    ""
                }
            } catch (e: Exception) {
                // Handle the exception, e.g., log it or throw a custom exception
                Log.e("AuthRepositoryImpl", "Error processing photoUrl: $photoUrl. Error: ${e.message}")
                ""
            }
        }
```

```
override suspend fun getUserData(): Response<User> = flow {
    try {
        val user = getUserDataFromFirebase()
        emit(Response.Success(user))
        Log.d("response", "getUserData: $user")
    } catch (e: Exception) {
        emit(Response.Failure(e))
    }
}.flowOn(Dispatchers.IO).single()

// In AuthRepository

override suspend fun updateUserProfile(
    uid: String, displayName: String, photoUri: String
): Response<User> = flow {
    try {
        if (displayName.isBlank()) {
            emit(Response.Failure(Exception("Display name cannot be empty")))
            return@flow
        }

        // Fetch the current user's data
        val currentUser = getUserDataFromFirebase()

        // Delete the previous image from Firebase Storage
        currentUser.photoUrl?.let { deleteImageFromFirebaseStorage(it) }

        // Upload the new image to Firebase Storage
        val uploadedPhotoUrl =
uploadImageToFirebaseStorage(Uri.parse(photoUri))
```

```
                if (uploadedPhotoUrl.isEmpty()) {
                    emit(Response.Failure(Exception("Failed to upload image to Firebase Storage")))
                    return@flow
                }

                // Update the user's profile with the new image URL
                val user = auth.currentUser
                val profileUpdates = UserProfileChangeRequest.Builder().setDisplayName(displayName)
                    .setPhotoUri(Uri.parse(uploadedPhotoUrl)).build()

                user?.updateProfile(profileUpdates)?.await()

                // Fetch the updated user data from Firebase
                val updatedUser = getUserDataFromFirebase()
                emit(Response.Success(updatedUser))
            } catch (e: Exception) {
                emit(Response.Failure(e))
            }
        }.flowOn(Dispatchers.IO).single()


    private suspend fun deleteImageFromFirebaseStorage(photoUrl: String): Response<Unit> {
        return try {
            if (photoUrl.isNotEmpty() && photoUrl.startsWith("https://")) {
                val storageRef = storage.getReferenceFromUrl(photoUrl)
                storageRef.delete().await()
```

```
            Response.Success(Unit)
        } else {
            Response.Failure(Exception("Invalid photoUrl format"))
        }
    } catch (e: Exception) {
        // Handle the exception, e.g., log it or throw a custom exception
        Response.Failure(e)
    }
}

override suspend fun sendEmailVerification(): Response<Unit> = flow {
    try {
        val user = auth.currentUser
        // If the user is already verified, don't send email again
        if (user?.isEmailVerified == true) {
            emit(Response.Failure(Exception("Your email is already verified")))
        } else {
            user?.sendEmailVerification()?.await()
            emit(Response.Success(Unit))
        }
    } catch (e: Exception) {
        emit(Response.Failure(e))
    }
}.flowOn(Dispatchers.IO).single()



override suspend fun markEmailAsVerified(): Response<Unit> = flow {
    try {
        val user = auth.currentUser
```

```
            user?.reload()?.await() // Reload the user data to get the latest email verification status
            val updatedUser = auth.currentUser // Retrieve the user again to get the updated status

            if (updatedUser?.isEmailVerified == true) {
                emit(Response.Success(Unit))
            } else {
                // If the user is not verified, send email again and show error message
                user?.sendEmailVerification()?.await()
                emit(Response.Failure(Exception("Your email is not verified. We have sent you another email. Please verify your email")))



            }
        } catch (e: Exception) {
            emit(Response.Failure(e))

        }
    }.flowOn(Dispatchers.IO).single()
    override suspend fun logout(): Response<Unit> = flow {
        try {
            auth.signOut()
            emit(Response.Success(Unit))
        } catch (e: Exception) {
            emit(Response.Failure(e))
        }
    }.flowOn(Dispatchers.IO).single()
}
```

UserDataViewModel.kt

```
package com.fahad.list_food.ui.screen

import android.net.Uri
import android.util.Log
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import androidx.navigation.NavController
import com.fahad.auth_firebase.domain.model.Response
import com.fahad.auth_firebase.domain.model.User
import com.fahad.auth_firebase.domain.repository.AuthRepository
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.NonCancellable
import kotlinx.coroutines.delay
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.launch
import kotlinx.coroutines.withContext
import javax.inject.Inject

@HiltViewModel
class UserDataViewModel @Inject constructor(private val authRepository: AuthRepository) :
    ViewModel() {
    private val _user = MutableStateFlow<User?>(null)
    val user: StateFlow<User?> = _user

    private val _profileError = MutableStateFlow<String?>(null)
    val profileError: StateFlow<String?> = _profileError
```

```
private val _editProfileError = MutableStateFlow<String?>(null)
val editProfileError: StateFlow<String?> = _editProfileError

private val _isLoading = MutableStateFlow(false)
val isLoading: StateFlow<Boolean> = _isLoading



// Mutable state flows for internal use
private val _profileSuccess = MutableStateFlow<String?>(null)
private val _editProfileSuccess = MutableStateFlow<String?>(null)

// Exposed state flows for external observation
val profileSuccess: StateFlow<String?> = _profileSuccess
val editProfileSuccess: StateFlow<String?> = _editProfileSuccess

private val _isEmailVerified = MutableStateFlow(false)
val isEmailVerified: StateFlow<Boolean> = _isEmailVerified



fun getUserData() {



    viewModelScope.launch {
        val response = authRepository.getUserData()
        if (response is Response.Success) {
            _user.value = response.data
            // Check if the user is already verified
            _isEmailVerified.value = response.data.isEmailVerified
          Log.d("TAG", "getUserData: ${_user.value?.displayName}")
          Log.d("TAG", "getUserData: ${response.data.displayName}")
```

```
            }
        }

    }


    fun markEmailAsVerified() {
        _isLoading.value = true
        _profileSuccess.value = null
        _isEmailVerified.value = false
        _profileError.value = null

        viewModelScope.launch {
            try {

                val response = authRepository.markEmailAsVerified()
                if (response is Response.Success) {
                    _profileSuccess.value = "Email marked as verified"
                    _isEmailVerified.value = true
                } else if (response is Response.Failure) {
                    _profileError.value =
                        "Failed to mark email as verified: ${response.exception.message}"
                }



            } catch (e: Exception) {
                _profileError.value = "Failed to mark email as verified: ${e.message}"
```

```
            } finally {
                _isLoading.value = false


            }
        }
    }


    fun setUser(userData: User) {
        _user.value = userData
    }

    // In UserDataViewModel
    fun updateUserProfile(displayName: String, photoUri: Uri, navController: NavController) {
        _isLoading.value = true

        _editProfileError.value = null
        _editProfileSuccess.value = null


        // Update data in Firebase
        val currentUser = user.value
        val uid = currentUser?.uid

        if (uid != null) {
            viewModelScope.launch {
```

```
                try {
                    val response =
                        authRepository.updateUserProfile(uid, displayName,
photoUri.toString())
                    if (response is Response.Success) {
                        // Update local data only if the remote update is successful
                        _user.value = currentUser.copy(
                            displayName = displayName, photoUrl = photoUri.toString()
                        )

                        _editProfileSuccess.value = "Profile updated successfully"
                        // Navigate to the profile screen after the snackbar disappears
                        delay(2000)
                        navController.popBackStack()



                    } else if (response is Response.Failure) {
                        _editProfileError.value =
                            "Failed to update profile: ${response.exception.message}"
                    }
                } catch (e: Exception) {
                    _editProfileError.value = "Failed to update profile: ${e.message}"
                } finally {
                    _isLoading.value = false
                }
            }
        }
    }
```

```
    fun clearMessages() {

        _profileError.value = null
        _profileSuccess.value = null
        _editProfileError.value = null
        _editProfileSuccess.value = null
    }



    fun logout() {
        viewModelScope.launch {
            authRepository.logout()
            _user.value = null
        }
    }
}
```

CartViewModel.kt

```
package com.fahad.list_food.ui.screen.cart

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.fahad.list_food.data.local.BookItem


import com.fahad.list_food.data.local.availableBooks


import com.fahad.list_food.data.local.entities.Item
import com.fahad.list_food.data.local.repository.ItemRepository

import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.flow.Flow
import kotlinx.coroutines.launch
import javax.inject.Inject


@HiltViewModel
class CartViewModel @Inject constructor(private val itemRepository: ItemRepository):ViewModel() {
    val cart: Flow<List<Item>> = itemRepository.getAllItems()


    val groupedItems = availableBooks.groupBy { it.bookType }

    fun addToCart(item: BookItem) {
        viewModelScope.launch {
```

```
            val newItem = Item(
                title = item.title,
                description = item.description,
                imageResId = item.imageResId,
                price = item.price
            )
            itemRepository.insertItem(newItem)
        }
    }

    fun deleteItem(item: Item) {
        viewModelScope.launch {
            itemRepository.deleteItem(item)
        }
    }

    fun clearCart() {
        viewModelScope.launch {
            itemRepository.deleteAllItems()
        }
    }

    fun incrementItem(item: Item) {
        viewModelScope.launch {
            itemRepository.incrementItemQuantity(item.id)
        }
    }

    fun decrementItem(item: Item) {
         viewModelScope.launch {
```

```
                itemRepository.decrementItemQuantity(item.id)
            }
        }
    }
```

RegisterViewModel.kt

```
package com.fahad.list_food.ui.screen.auth.register

import android.util.Log
import androidx.compose.runtime.mutableStateOf
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import androidx.navigation.NavController
import
com.fahad.auth_fierbase_bottomnavigation.ui.screen.navigation.auth.AuthScreen
import com.fahad.auth_firebase.domain.model.Response
import com.fahad.auth_firebase.domain.model.User
import com.fahad.auth_firebase.domain.repository.AuthRepository
import com.fahad.list_food.ui.navigation.Graph
import com.fahad.list_food.ui.screen.UserDataViewModel


import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.flow.first
import kotlinx.coroutines.launch
import javax.inject.Inject

@HiltViewModel
```

```
class RegisterViewModel @Inject constructor(
    private val repository: AuthRepository, private val userDataViewModel: UserDataViewModel
) : ViewModel() {
    private val _registrationState = MutableStateFlow<Response<User>>(Response.Loading)
    val registrationState: StateFlow<Response<User>> = _registrationState

    private val _isLoading = mutableStateOf(false)
    val isLoading: Boolean
        get() = _isLoading.value


    fun registerUser(
        email: String, password: String, displayName: String, photoUri: String, // Make it nullable
        navController: NavController
    ) {
        _isLoading.value = true
        viewModelScope.launch {
            _registrationState.value = Response.Loading
            try {
                val registrationResult = repository.registerUser(
                    email, password, displayName, photoUri
                ).first()
                _isLoading.value = false

                if (registrationResult is Response.Success) {
                    val user = registrationResult.data
                    userDataViewModel.setUser(user)
```

```
                    navController.navigate( Graph.HOME)
                } else if (registrationResult is Response.Failure) {
                    _registrationState.value = registrationResult

                }
            } catch (e: Exception) {
                Log.e("ViewModel", "registerUser: ${e.message}")
                _registrationState.value = Response.Failure(Exception("Something went wrong"))

            }
        }
    }
}
```

Додаток Г. Ілюстративна частина

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ІНТЕРАКТИВНОГО ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ ОНЛАЙН-КНИГАРНІ

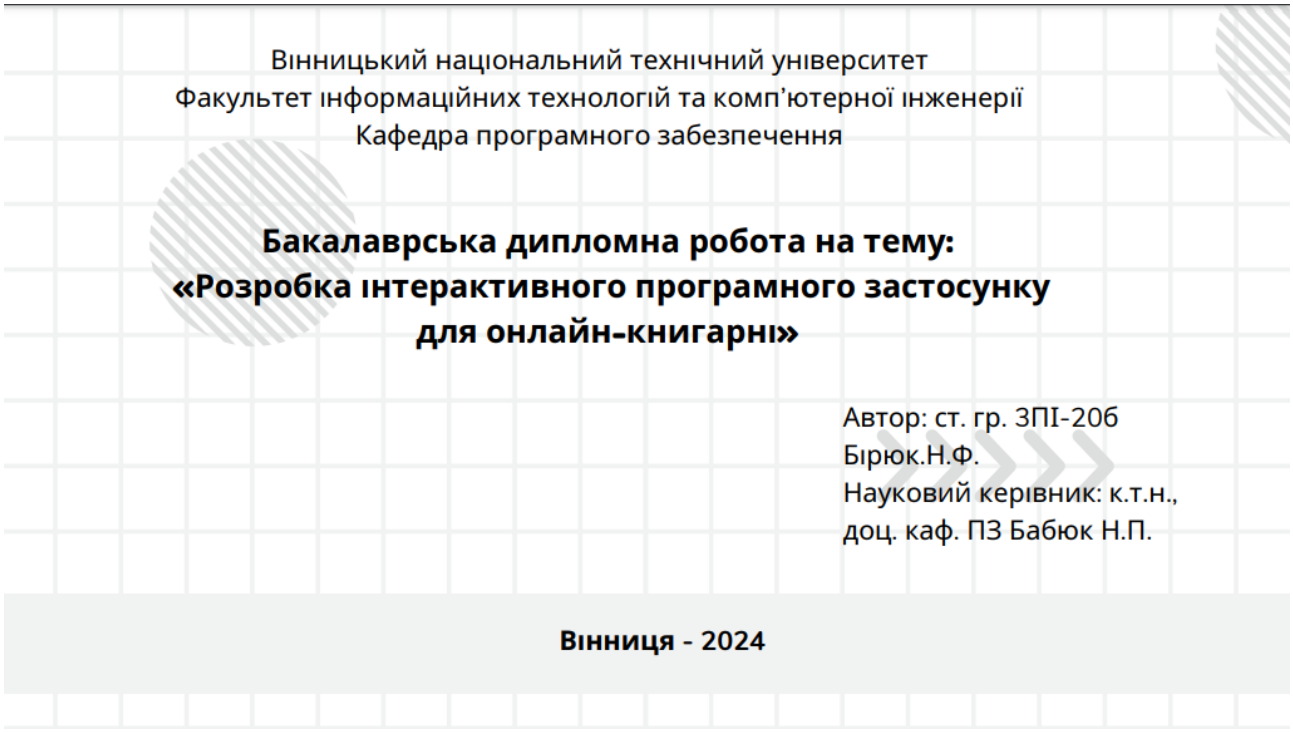

Рисунок Г.1 – Титульний слайд

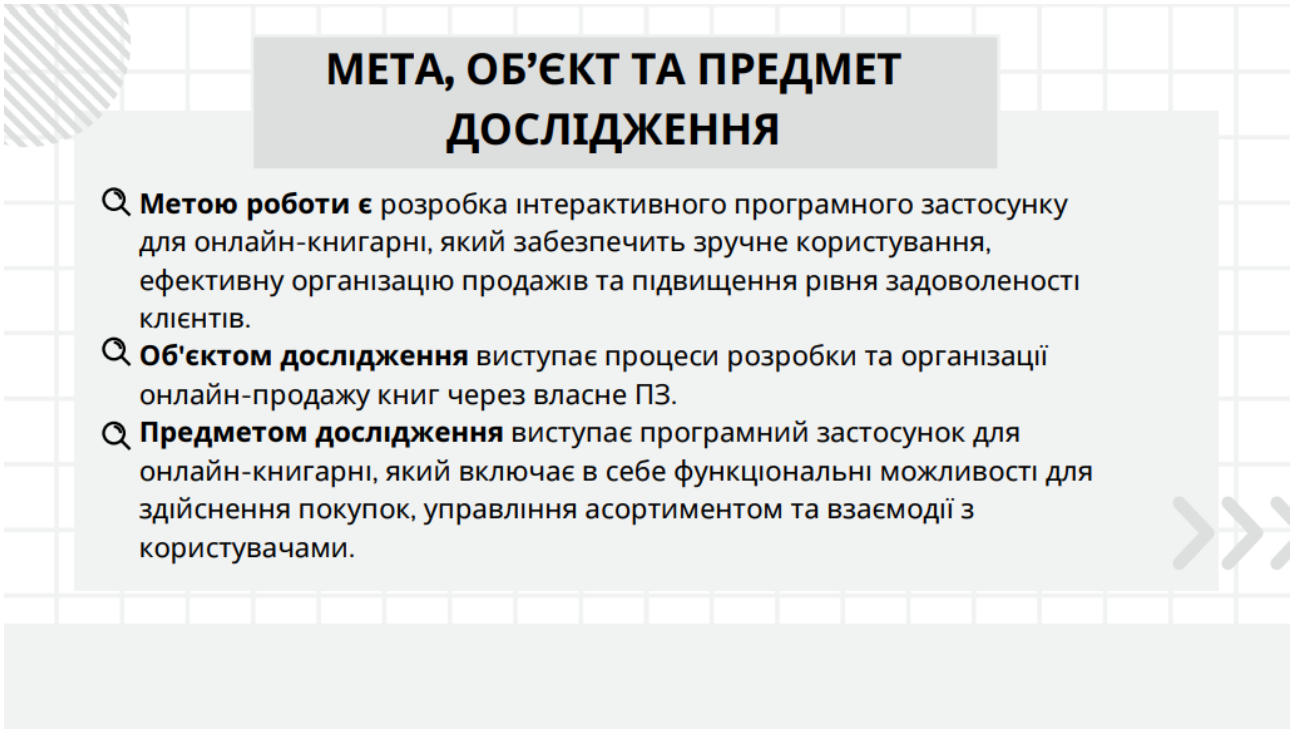

Рисунок Г.2 – Мета, об'єкт та предмет дослідження

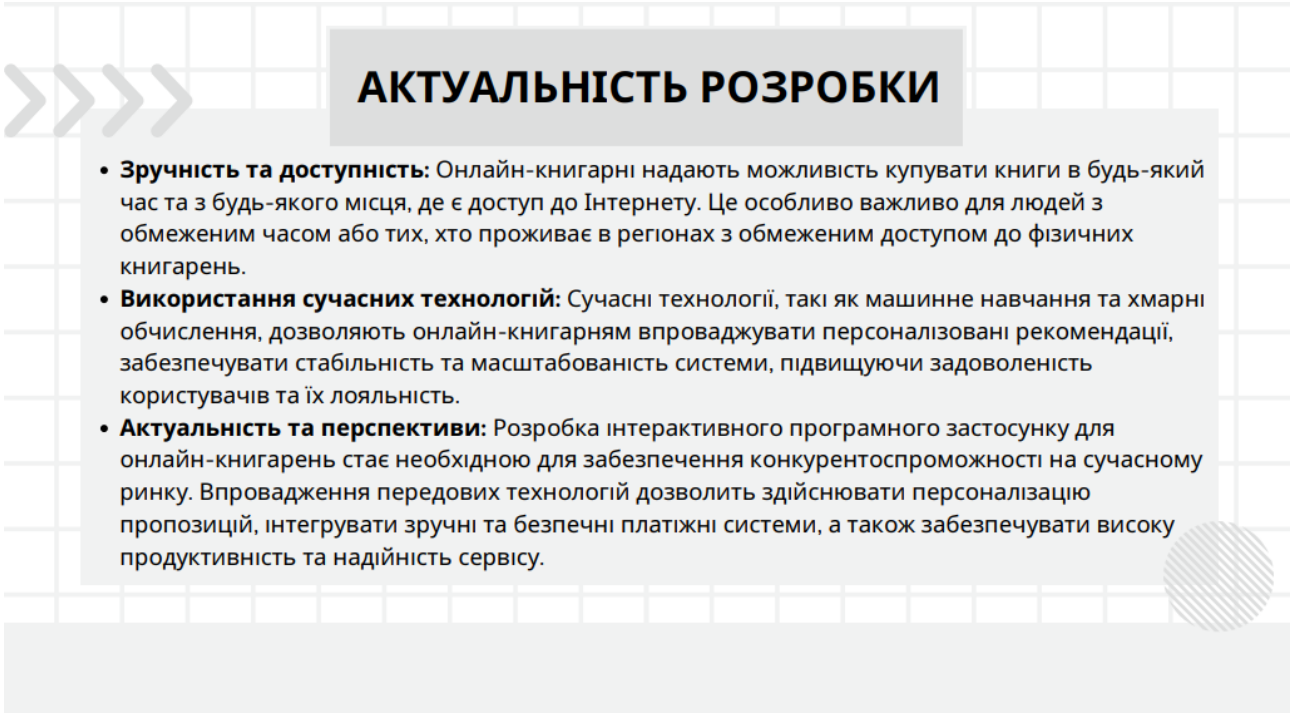

Рисунок Г.3 – Актуальність розробки

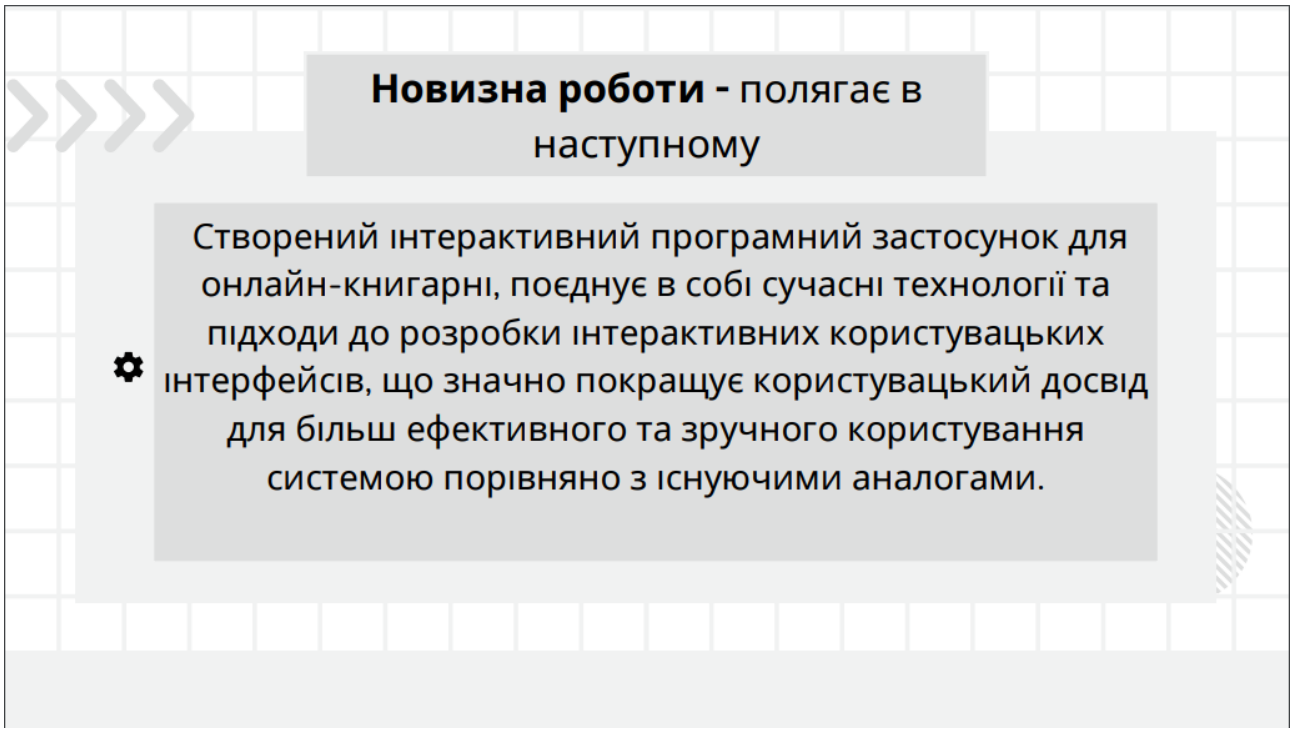

Рисунок Г.4 – Новизна роботи

ПОРІВНЯЛЬНИЙ АНАЛІЗ АНАЛОГІВ

Характеристика	Amazon	Barnes & Noble	Kobo	БДР
Широкий асортимент	+	–	–	+
Зручність користування	+	+/–	+	+
Персоналізація рекомендацій	+	+/–	+/–	+
Інтерактивні функції	–	–	+	+
Синхронізація між пристроями	+	–	+	+
Локалізація	+	+	–	+
Надійність бренду	+	+	+/–	+
Ціни та акції	+/–	+	+/–	+
Екологічність	+/–	–	+	+

Рисунок Г.5 – Порівняльний аналіз аналогів

ВИБІР ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ ЗАСТОСУНКУ

Характеристики	Java	JavaScript	Swift	Kotlin
Сучасні можливості	+/–	+/–	+	+
Взаємодія з Java	+	–	–	+
Обсяг коду	–	+	+	+
Офіційна підтримка Android	+	-	–	+
Універсальність	+/–	+	–	+
Продуктивність	+	–	+	+
Стабільність	+	–	+	+
Гнучкість розробки інтерфейсів	–	+	+	+

Рисунок Г.6 – Вибір засобів для реалізації застосунку

Рисунок Г.7 – Загальний алгоритм системи

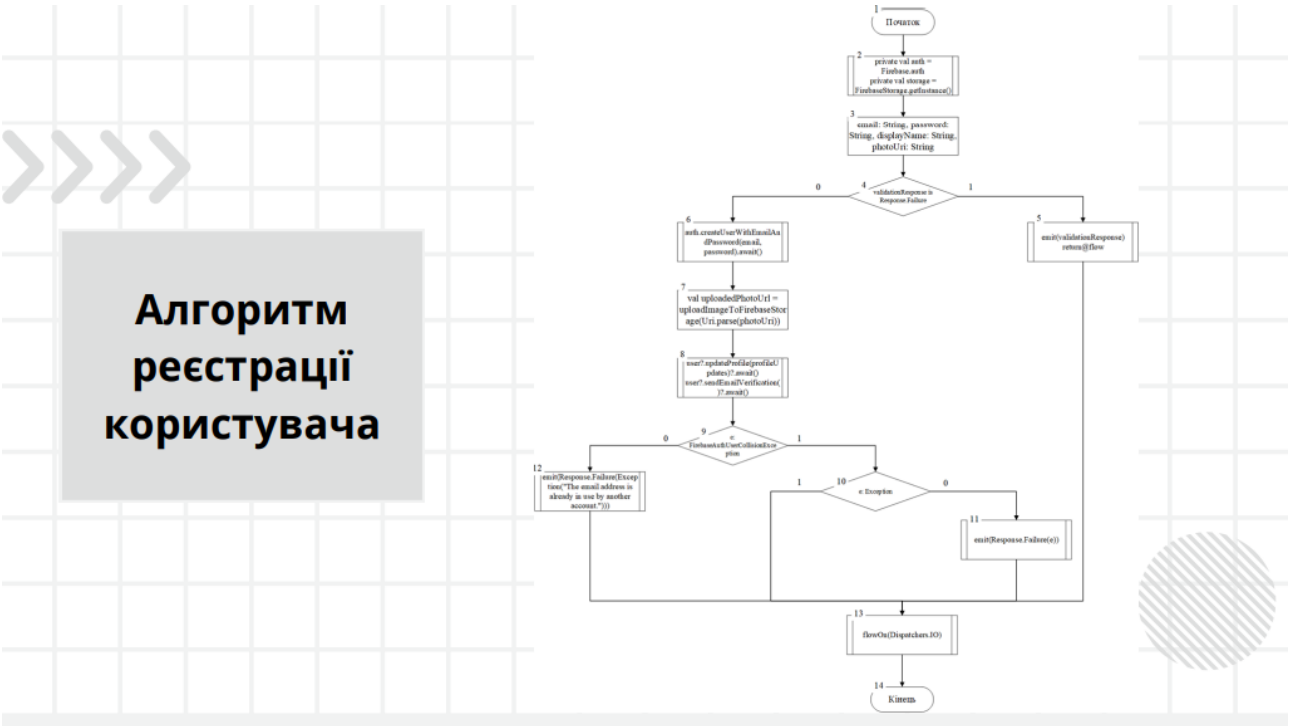

Рисунок Г.8 – Алгоритм реєстрації користувача

Рисунок Г.9 – Використані технології

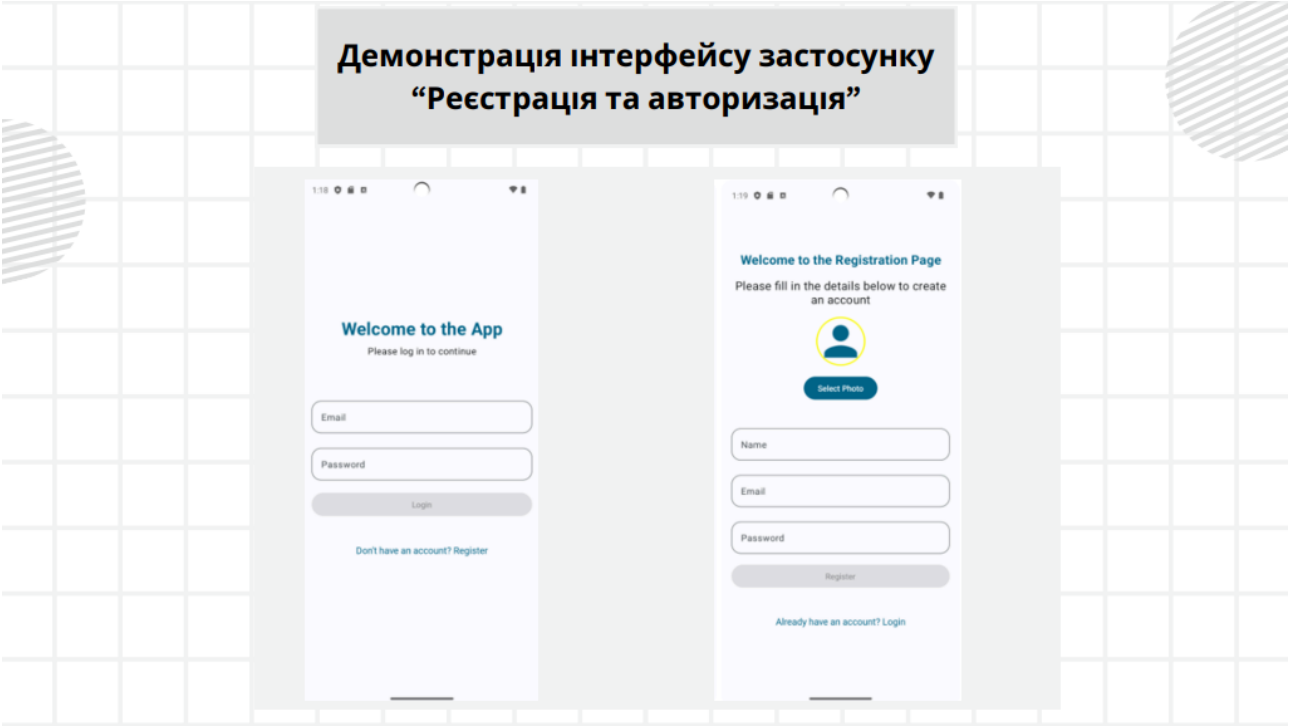

Рисунок Г.10 Демонстрації інтерфейсу застосунку “Реєстрація та авторизація”

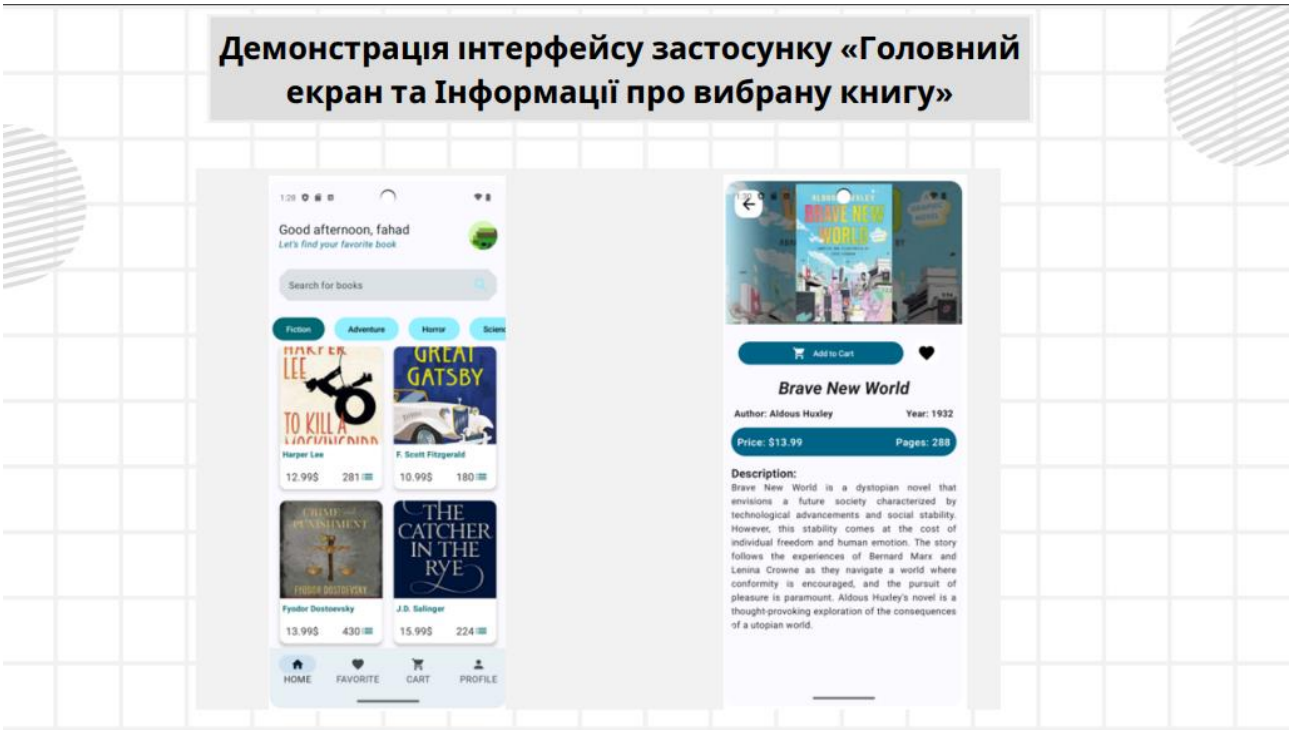

Рисунок Г.11 – Демонстрації інтерфейсу застосунку “Головний екран та Інформації про вибрану книгу”

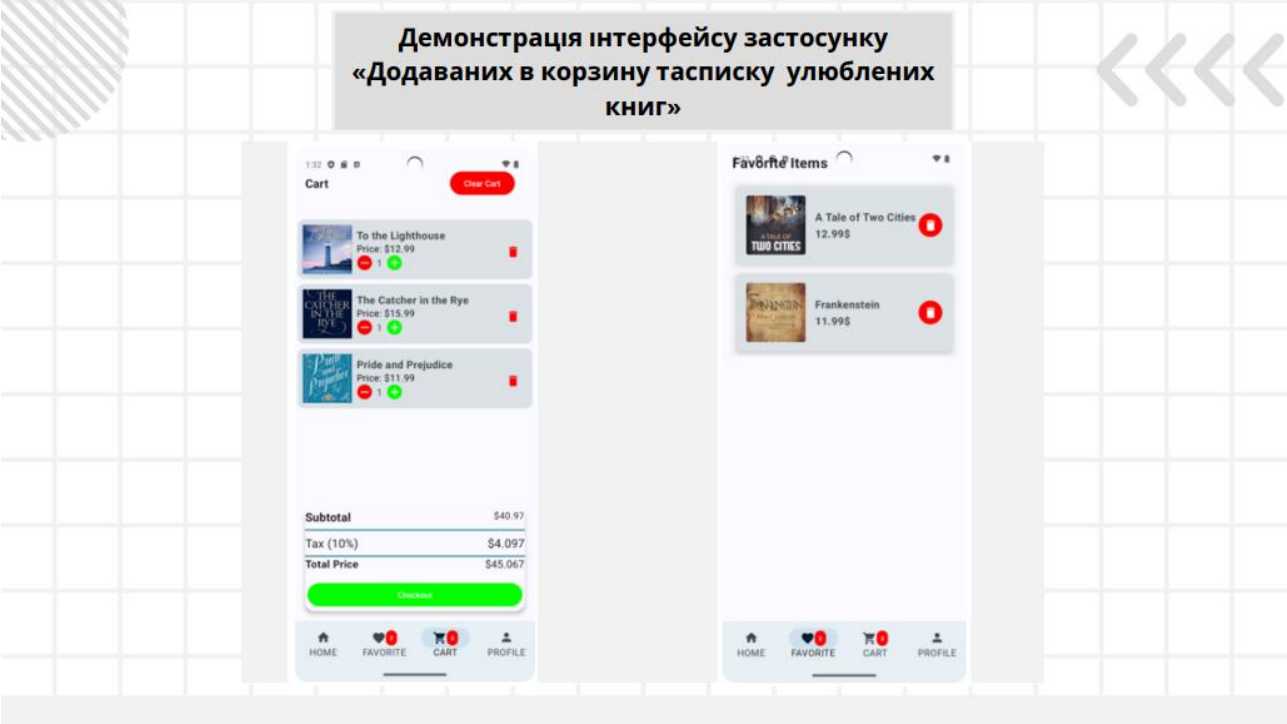

Рисунок Г.12 – Демонстрації інтерфейсу застосунку “Додаваних в корзину та списку улюблених”

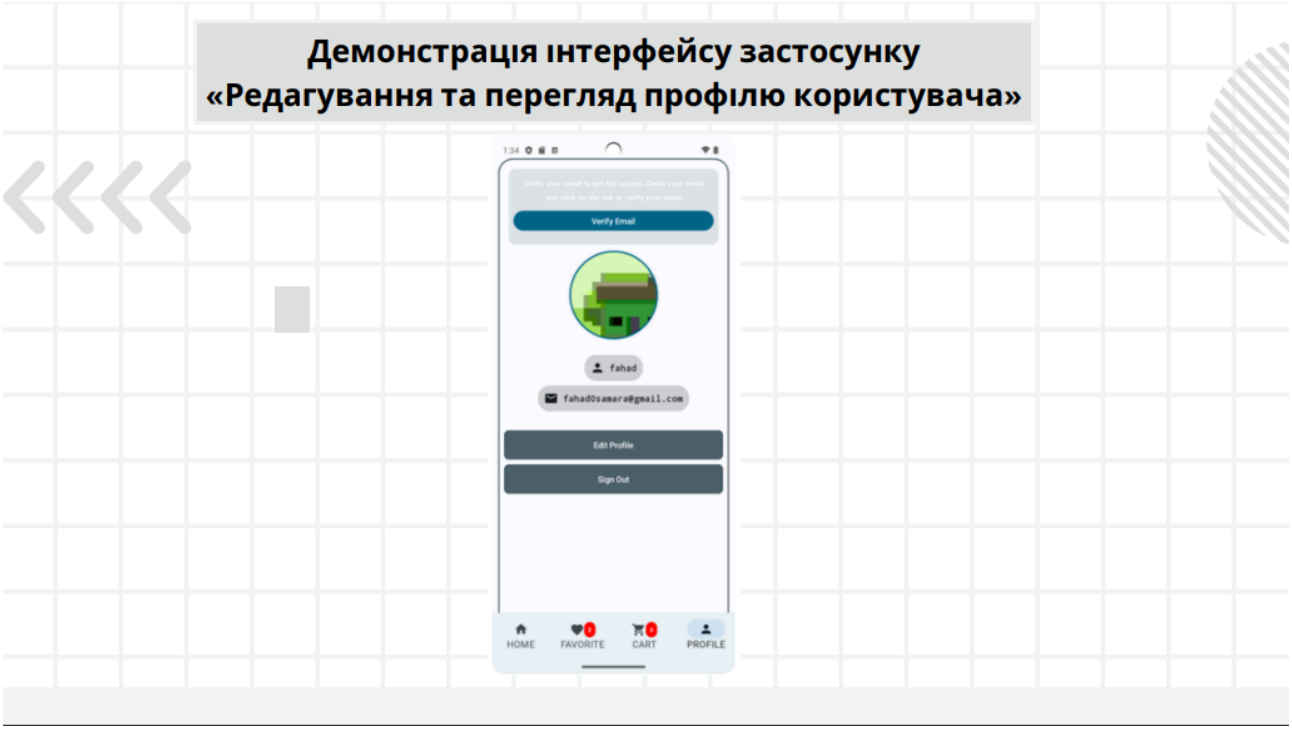

Рисунок Г.13 – Демонстрації інтерфейсу застосунку “Редагування та перегляд профілю користувача”

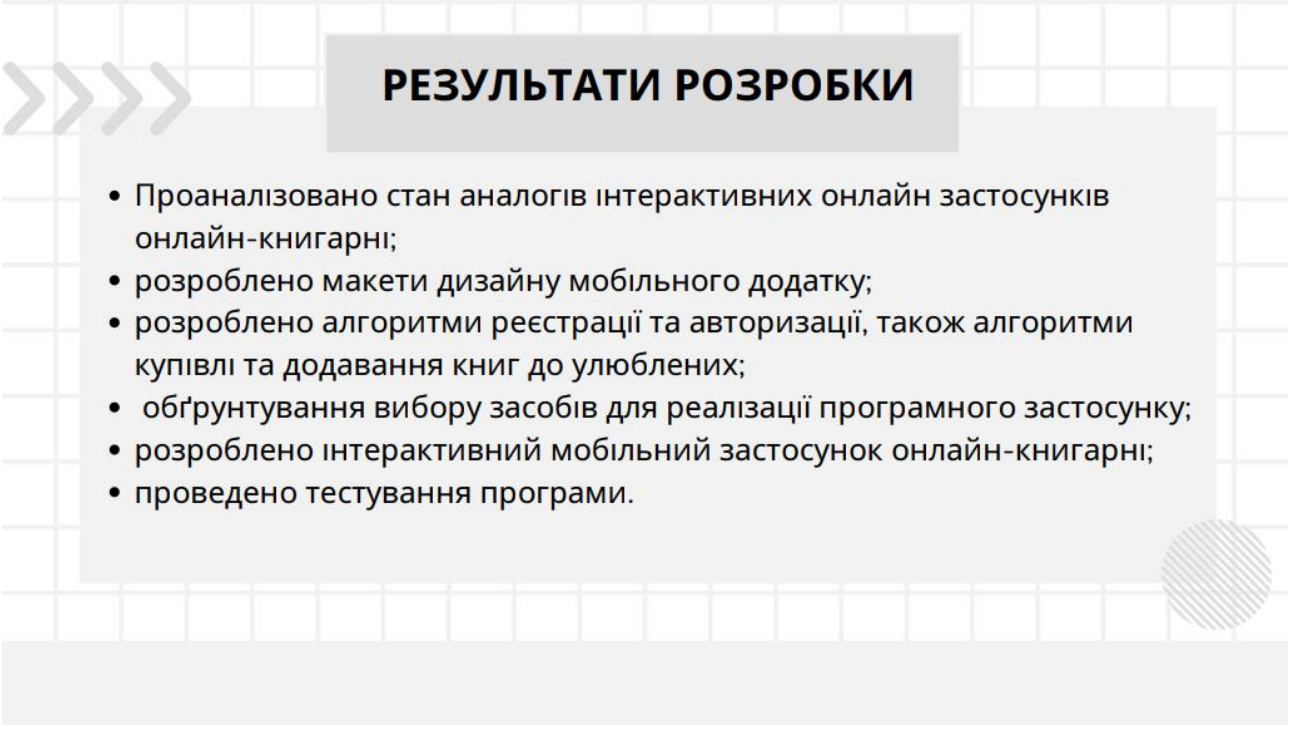

Рисунок Г.14 – Результати розробки

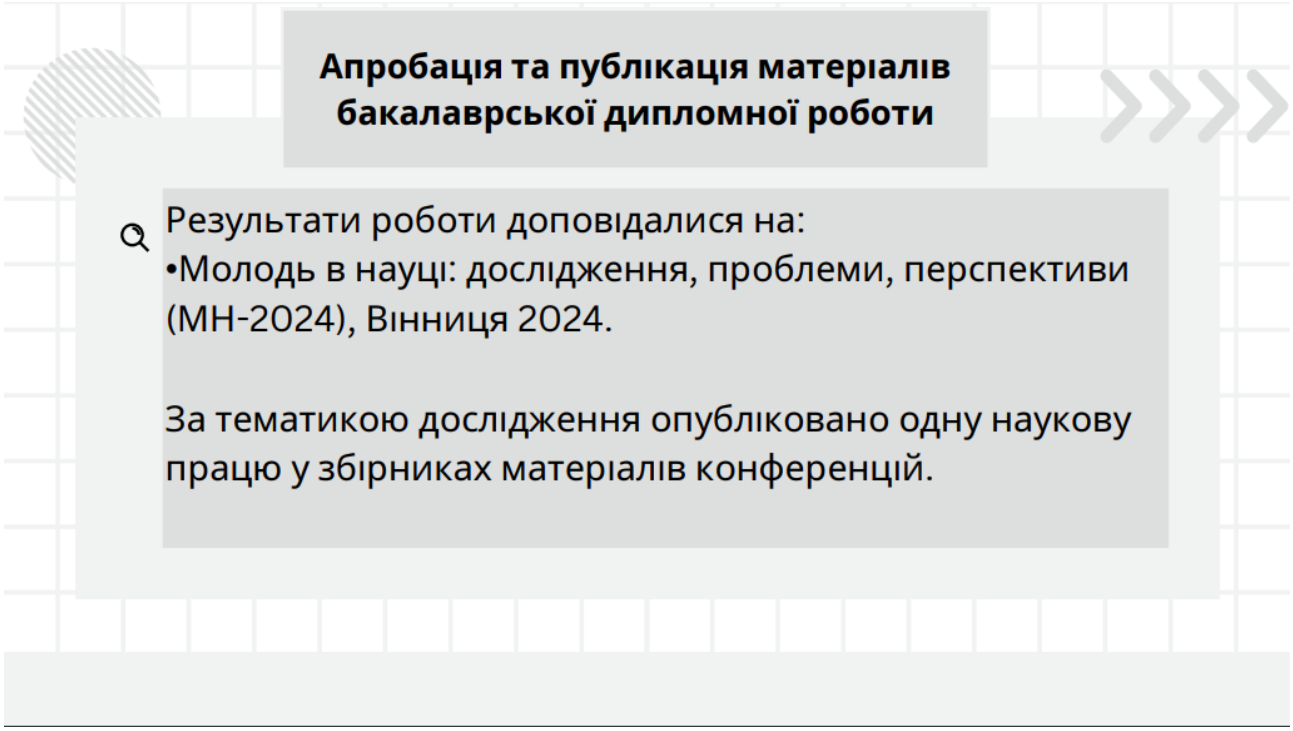

Рисунок Г.15 – Демонстрації інтерфейсу застосунку “ Апробація та публікація матеріалів бакалаврської дипломної роботи ”

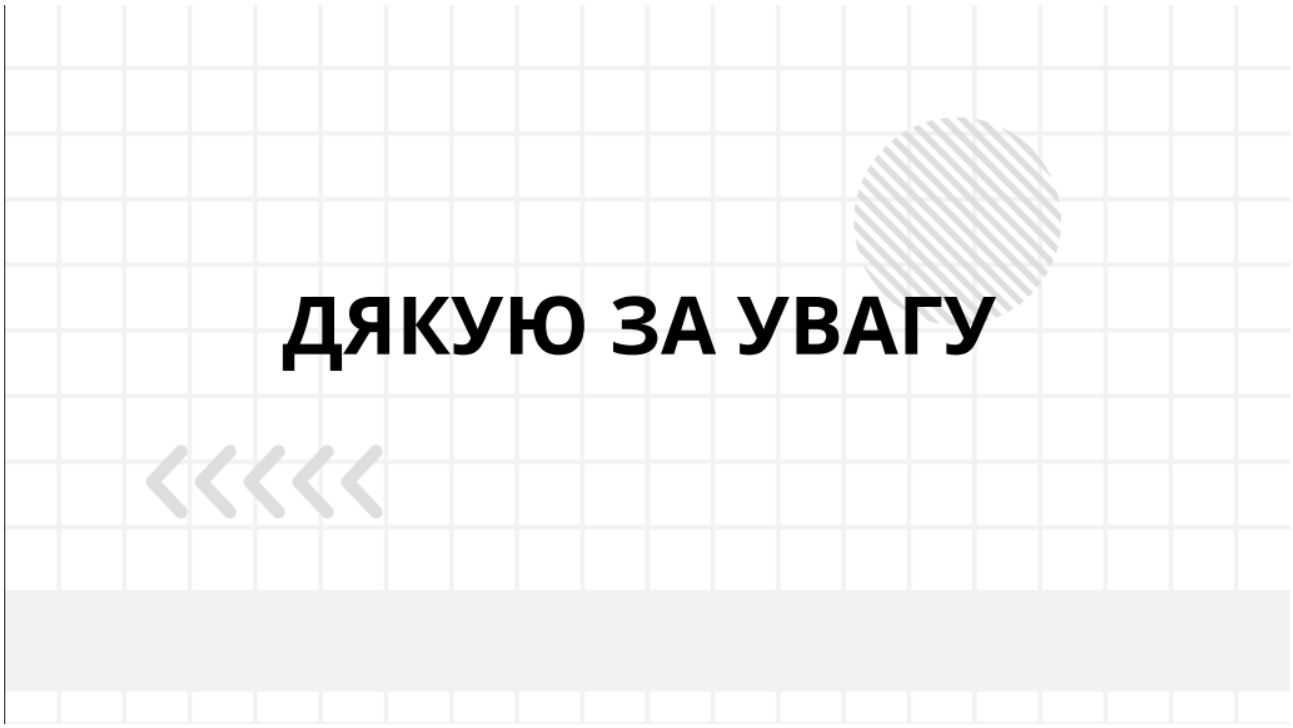

Рисунок Г.16 – Фінальний слайд