Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Мобільний додаток для читання та прослуховування електронних книг

(тема)

Виконав:
здобувач _4_ року навчання
групи ПЗПІ-21-7

Ілля ВИСОЧИН
(Власне ім'я, ПРІЗВИЩЕ)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. кафедри ПІ Наталя КРАВЕЦЬ
(посада, Власне ім'я, ПРІЗВИЩЕ)

Допускається до захисту
Зав. кафедри _____ Кирило СМЕЛЯКОВ
(підпис) (Власне ім'я, ПРІЗВИЩЕ)

2025 р.

Харківський національний університет радіоелектроніки

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

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Височину Іллі Максимовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Мобільний додаток для читання та прослуховування електронних книг

Затверджена наказом по університету від 19.05.2025р. № 397 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 16.06.2025

3. Вихідні дані до роботи: розробити мобільний додаток для читання книг с функціоналом озвучення та перекладу в режимі офлайн та онлайн.

4. Перелік питань, що потрібно опрацювати в роботі

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	20.04.2025	виконано
2	Створення специфікації ПЗ	29.04.2025	виконано
3	Проектування ПЗ	07.05.2025	виконано
4	Розробка ПЗ	12.05.2025	виконано
5	Тестування ПЗ	20.05.2025	виконано
6	Оформлення пояснювальної записки	03.06.2025	виконано
7	Підготовка презентації та доповіді	06.06.2025	виконано
8	Попередній захист	13.06.2025	виконано
9	Нормоконтроль, рецензування	13.06.2025	виконано
10	Здача роботи у електронний архів	11.06.2025	виконано
11	Допуск до захисту у зав. кафедри	13.06.2025	виконано

Дата видачі завдання «20» «квітня» 2025р.

Здобувач _____ Ілля ВИСОЧИН
(підпис)

Керівник роботи _____ доц. кафедри ПІ Наталя КРАВЕЦЬ
(підпис) (посада, Власне ім’я, ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра, 113 стор., 18 рис., 17 джерел.

ЕЛЕКТРОННА КНИГА, МОБІЛЬНИЙ ДОДАТОК, ОЗВУЧУВАННЯ ТЕКСТУ, ПЕРЕКЛАД СЛІВ, REACT NATIVE, TYPESCRIPT

Об’єкт розробки – мобільний програмний додаток для читання та прослуховування електронних книг.

Мета розробки – створення кросплатформенного мобільного додатка для читання електронних книг з підтримкою автоматичного озвучування тексту та перекладу окремих слів у режимі реального часу.

Метод рішення – використання фреймворку React Native, мови програмування TypeScript, а також технологій машинного навчання для озвучування тексту та перекладу.

У результаті розробки буде створено мобільний програмний додаток, що дозволяє користувачам читати електронні книги різних форматів, озвучувати їх вголос, перекладати незнайомі слова та персоналізувати інтерфейс читання для зручності всіх груп користувачів.

ABSTRACT

E-BOOK, MOBILE APPLICATION, TEXT-TO-SPEECH, WORD TRANSLATION, REACT NATIVE, TYPESCRIPT

The object of development is a mobile software application for reading and listening to e-books.

The purpose of the development is creation of a cross-platform mobile application for reading e-books with the support of automatic reading of text and translation of words in real time.

The solution method involves using the React Native framework, the TypeScript programming language, as well as machine learning technologies for text-to-speech and word translation.

As a result of the development, a mobile application will be created that allows users to read e-books in various formats, listen to them aloud, translate unfamiliar words, and personalize the reading interface for the convenience of all user groups.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

TTS – Text to Speech

ML – Machine Learning

SRS – Software Requirements Specifications

API – Application Programming Interface

HTTPS – HyperText Transfer Protocol Secure

WCAG - Web Content Accessibility Guidelines

FSD – Feature-Sliced Design

ВСТУП

У сучасному світі інформаційні технології стають невід'ємною частиною повсякденного життя. Важливим напрямком їх застосування є оптимізація процесу доступу до знань та інформації. Зі зростанням обсягів споживання цифрового контенту, особливо літератури, виникає потреба у зручних засобах читання електронних книг, які не лише надають базовий функціонал, а й враховують потреби широкого кола користувачів: від людей із вадами зору до тих, хто вивчає іноземні мови. Такі вимоги обумовлюють актуальність створення мобільного додатка, який поєднує у собі функції читання, озвучування та перекладу текстів.

Існуючі мобільні додатки, що забезпечують подібний функціонал, здебільшого мають обмеження у гнучкості використання, мовній підтримці або вимагають складного налаштування. Використання сучасних інструментів розробки, таких як React Native, TypeScript, а також інтеграція засобів машинного навчання для озвучування та перекладу, дозволяють створити зручний, інтуїтивно зрозумілий і технологічно передовий програмний продукт. Таким чином, розробка такого додатка є своєчасною і необхідною, з огляду на постійне зростання попиту на цифрові книжкові сервіси.

Метою проєкту є створення кросплатформенного мобільного додатка для читання електронних книг з підтримкою автоматичного озвучування тексту та перекладу окремих слів у режимі реального часу.

Задля досягнення поставленої мети буде виконано такі завдання:

- проаналізовано вимоги цільової аудиторії до програмного забезпечення для читання електронних книг;
- розроблено функціональні вимоги до додатка ;
- спроєктовано архітектуру системи із використанням сучасних інструментів та підходів;
- реалізовано модулі для озвучування тексту та перекладу слів за допомогою інструментів машинного навчання;

– створено зручний користувацький інтерфейс із можливістю персоналізації;

– протестовано додаток для перевірки його працездатності та відповідності очікуванням користувачів.

Об’єктом дослідження є мобільний додаток для роботи з електронними книгами.

Предметом дослідження є функціональні та технологічні рішення, що забезпечують зручне читання, озвучування та переклад текстового контенту.

У ході роботи були застосовані методи функціонального аналізу, огляду сучасних технологій, а також порівняльний аналіз наявних рішень у цій галузі. Особливу увагу було приділено технічним аспектам інтеграції озвучування тексту та перекладача в архітектуру мобільного додатка .

Реалізація мобільного додатка дає змогу підвищити рівень доступності книжкового контенту, сприяє популяризації читання серед користувачів, зокрема серед людей з особливими потребами, а також допомагає ефективніше вивчати іноземні мови. Використання сучасних інструментів і технологій гарантує масштабованість та зручність подальшого розвитку системи.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

У XXI столітті цифрові технології проникли практично в усі сфери людської діяльності, і однією з найпомітніших тенденцій є цифровізація процесів споживання інформації. Зокрема, стрімке поширення мобільних пристроїв та зростання популярності електронного контенту призвело до активного розвитку ринку електронних книг та мобільних додатків для їх читання. Читання книг з екранів смартфонів або планшетів стало звичною справою для мільйонів користувачів у всьому світі, оскільки це дозволяє мати доступ до літератури у будь-якому місці та в будь-який час.

Сучасна індустрія e-reading — це сукупність програмних рішень, що забезпечують доступ, зберігання, читання, керування та взаємодію з текстовим контентом в електронному форматі. До найбільш поширених форматів електронних книг належать EPUB, MOBI, PDF, TXT, DJVU та інші. Більшість популярних читалок, таких як Google Play Books, Audible, Speechify, Amazon Kindle, Apple Books або ReadEra, забезпечують базовий функціонал: перегортання сторінок, зміна розміру шрифту, вибір теми інтерфейсу, закладки та ін. Деякі додатки дозволяють озвучувати текст, але ця функціональність реалізована далеко не у всіх продуктах або має обмежений набір мов та голосів [1].

Крім того, одним із важливих напрямків є доступність цифрового контенту для людей із порушеннями зору. Згідно з даними Всесвітньої організації охорони здоров’я, близько 2,2 мільярдів людей у світі мають порушення зору, і значна частина з них використовує програми озвучування тексту (text-to-speech, TTS) як основний засіб взаємодії з цифровим контентом. Таким чином, впровадження технологій синтезу мовлення є критично важливим для забезпечення інклюзивності.

Ще однією актуальною тенденцією є вивчення іноземних мов через занурення у мовне середовище. Електронні книги — зручне джерело автентичного тексту, однак читачам, які тільки починають вивчення, часто складно розуміти

контекст або значення окремих слів. Тому потреба у швидкому перекладі окремих слів або фраз без переривання читання є ключовим фактором підвищення ефективності навчання.

На ринку існують розрізнені рішення, що вирішують окремі проблеми — наприклад, програми TTS, окремі перекладачі, або читалки з базовим функціоналом. Проте інтегровані рішення, що об’єднують всі ці функції у одному додатка , зустрічаються рідко, або ж мають обмеження за платформами, локалізацією чи зручністю використання.

Ще одним аспектом, який варто враховувати під час аналізу предметної галузі, є персоналізація інтерфейсу та зручність користування. Сучасні користувачі очікують адаптивного дизайну, можливості налаштування зовнішнього вигляду тексту, вибору мови озвучування, налаштування швидкості читання, а також синхронізації з хмарними сервісами для зберігання книг.

З розвитком мобільної розробки та фреймворків кросплатформеного програмування, таких як React Native, з’явилася можливість створювати функціональні та зручні мобільні додатки, які працюють як на Android, так і на iOS. Використання TypeScript у поєднанні з цим фреймворком дозволяє покращити якість коду, прискорити розробку та спростити підтримку продукту.

Загалом, предметна галузь, що пов’язана з мобільними засобами доступу до електронних книг, має великий потенціал для розвитку. Ключовими напрямками інновацій є:

- інтеграція засобів озвучування тексту;
- можливість перекладу окремих слів або фраз у реальному часі;
- персоналізація користувацького досвіду;
- забезпечення доступності для людей з особливими потребами;
- підтримка різноманітних форматів книжок;
- синхронізація з хмарними сховищами та обліковими записами.

Таким чином, створення додатка , що поєднує можливості для читання, озвучування і перекладу, є актуальним і перспективним напрямом, який відповідає вимогам часу та запитам широкої аудиторії користувачів.

2.2 Виявлення та вирішення проблем

Було проведено дослідження трьох популярних мобільних додатків, які мають схожий функціонал: Google Play Books, Audible, та Speechify.

2.2.1 Google Play Books: Переклад тексту у багатомовному середовищі [2].

Google Play Books надає можливість перекладати текст безпосередньо під час читання завдяки інтеграції з Google Translate (рис. 1). Цей функціонал корисний для користувачів, що читають іноземною мовою, однак має певні обмеження. Зокрема, під час обробки художніх текстів — особливо поетичних або з використанням діалектів — автоматичний переклад часто втрачає стилістичну й естетичну точність. Крім того, великі обсяги тексту, які перекладаються у режимі реального часу, можуть спричиняти затримки в роботі. Хоча Google Translate охоплює понад сотню мов, система не завжди враховує культурні нюанси. Наприклад, переклад японських текстів нерідко не передає граматичну ввічливість, що важливо для розуміння контексту.

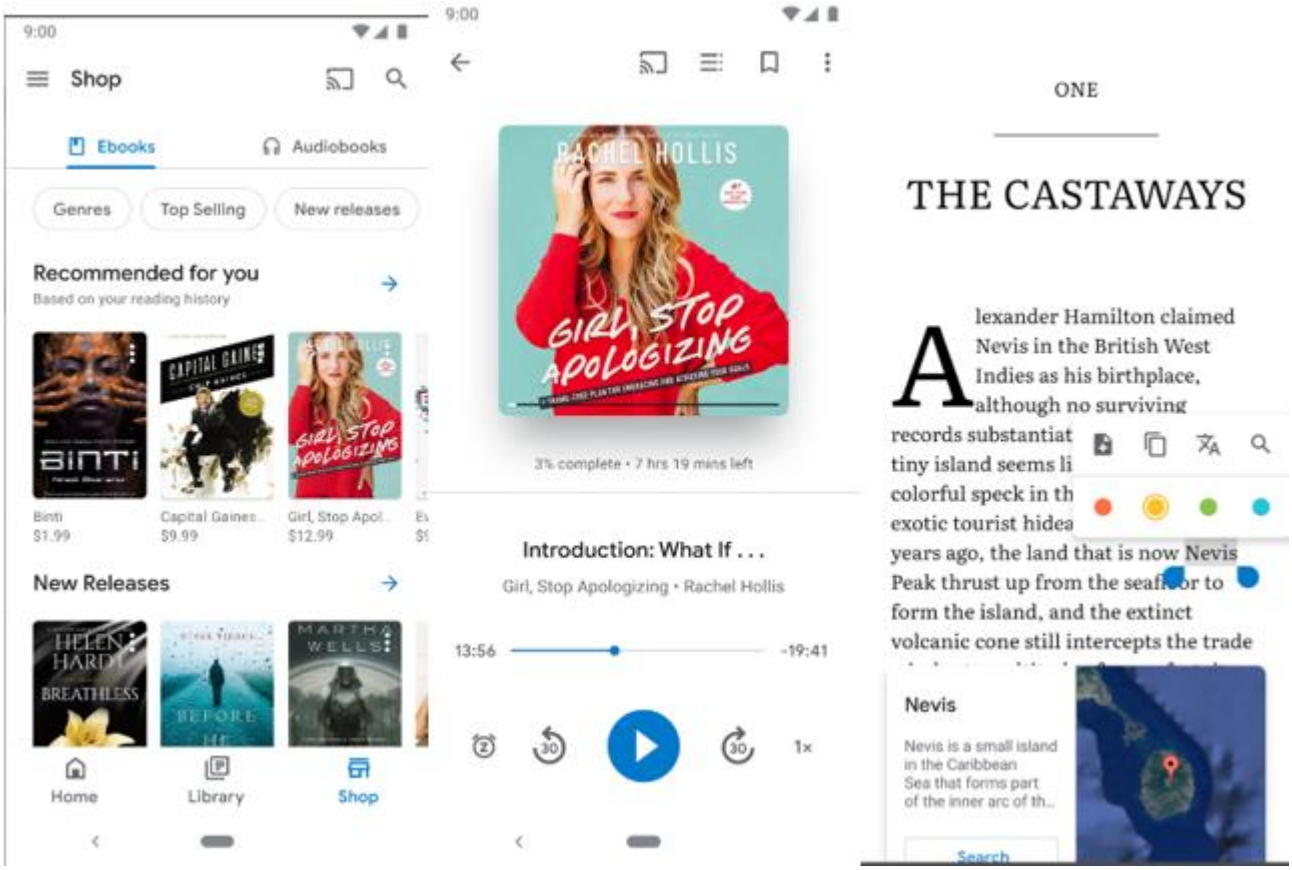

Рисунок 1 – Інтерфейс Google Play Books (за даними [2])

Окрім перекладу, Google Play Books пропонує й інші мовні функції, наприклад — озвучення фрагментів тексту (Text-to-Speech), що може бути особливо корисним для користувачів з вадами зору або тих, хто бажає поєднувати читання зі слуханням. Проте цей функціонал має обмеження щодо якості голосів, особливо для мов із меншою підтримкою. Наприклад, озвучення українською доступне лише на деяких пристроях та має помітно нижчу якість порівняно з англійською чи німецькою.

Ще однією проблемою є відсутність можливості збереження перекладів у персональний словник для подальшого перегляду або навчання. У той час як інтеграція з Google Translate дозволяє переглянути значення виділеного слова чи фрази, користувач не має змоги створити базу з перекладеними словами або позначити переклади як "вивчені". Це знижує цінність функції для тих, хто використовує додаток як інструмент для вивчення мови.

Крім того, Google Play Books не надає офлайн-доступу до перекладу, що обмежує функціональність у ситуаціях, коли користувач не має стабільного інтернет-з’єднання — наприклад, під час подорожей або в зоні з низьким покриттям. У таких випадках відсутність офлайн-режиму створює суттєві незручності, особливо в умовах, коли переклад є критично важливим для розуміння змісту.

Таким чином, попри розвинену інтеграцію з Google Translate, функціональність перекладу в Google Play Books має низку обмежень, які впливають на гнучкість, зручність і ефективність взаємодії користувача з текстом у багатомовному середовищі. Саме ці обмеження стали основою для проєктування альтернативного підходу в розроблювальному мобільному додатка , де передбачено підтримку офлайн-перекладу, збереження історії, персоналізацію перекладу та точне озвучення синтезованими голосами.

2.2.2 Audible: Генерація мовлення у аудіо книгах [3].

Audible — один із провідних сервісів у сфері аудіокниг, де переважають записи, створені професійними дикторами (рис. 2). Хоча компанія експериментує з технологіями машинного навчання для створення голосових версій книжок, їх

використання обмежене кількома тестовими релізами. У таких версіях відчувається штучність інтонацій, що знижує емоційне сприйняття творів, особливо у випадку літератури з драматичними сценами. Крім того, синтетичні голоси поки що недостатньо ефективно справляються з інтонаційною передачею та мають труднощі з правильною вимовою при озвученні текстів іншими мовами, крім англійської.

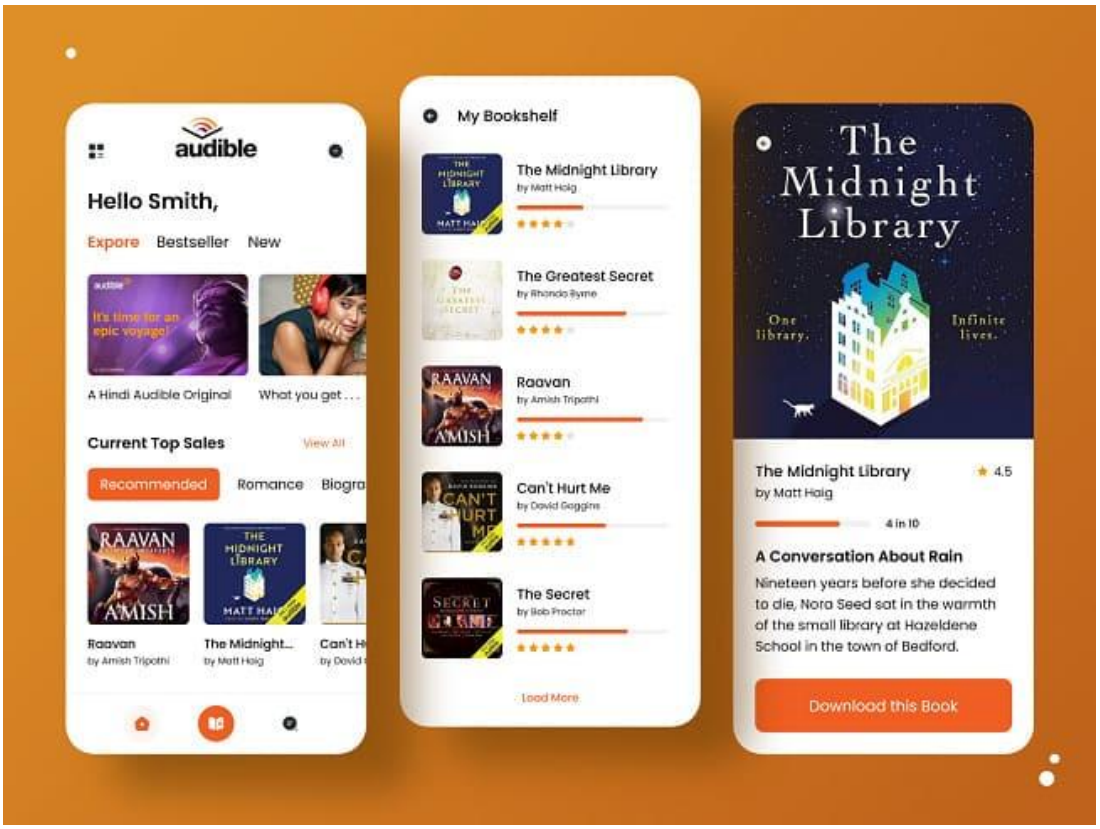

Рисунок 2 – Інтерфейс Audible (за даними [3])

Попри ці обмеження, Audible продовжує розширювати можливості персоналізованого прослуховування, зокрема шляхом інтеграції зі смарт-девайсами (наприклад, Alexa) та додаванням функцій, таких як регулювання швидкості відтворення, запам’ятовування останньої позиції або додавання закладок до важливих моментів. Проте платформа в основному орієнтована на попередньо записані професійні аудіоверсії, що обмежує її придатність для широкого спектру електронних книг, зокрема нових або рідкісних творів, які не були офіційно адаптовані до аудіоформату.

Крім того, Audible не підтримує можливості динамічного озвучення тексту користувачем (Text-to-Speech) або перекладу окремих фраз у процесі прослуховування. Для користувачів, які читають літературу іноземною мовою або вивчають мову, це становить істотне обмеження: відсутність перекладу «на льоту» унеможливлює повноцінну інтеграцію читання та навчання.

Ще одним недоліком є відсутність вільного доступу до контенту — більшість книг доступна лише за передплатою або окремою купівлею. Таким чином, використання Audible у якості платформи для щоденного навчального читання або озвучення текстів, не адаптованих до формату, є економічно менш доцільним для пересічного користувача.

Отже, незважаючи на високий рівень якості звуку та комфорт прослуховування, Audible є платформою з обмеженим функціоналом для інтерактивного читання. Її орієнтація на готові аудіокниги зумовлює відсутність гнучкості, яку можуть запропонувати інструменти на кшталт розроблюваного — із підтримкою онлайн- і офлайн-озвучення, перекладу, адаптивного TTS та інтегрованого словника.

2.2.3 Speechify: Інклюзивність та персоналізація [4].

Speechify позиціонується як TTS- додаток для користувачів із особливими потребами, наприклад, для людей із порушеннями зору або дислексією (рис. 3). Завдяки машинному навчанню програма забезпечує високоякісне перетворення тексту в мовлення, однак стикається з типовими для TTS-систем труднощами. Зокрема, проблеми виникають при обробці абревіатур, спеціалізованої термінології та символів, що часто трапляються в наукових або технічних текстах. Крім того, розширений функціонал (наприклад, доступ до більш природних голосів) доступний лише в платній версії, що обмежує її доступність. Попри персоналізацію голосу та темпу читання, поки що не реалізовано можливість створення індивідуальних голосів користувача.

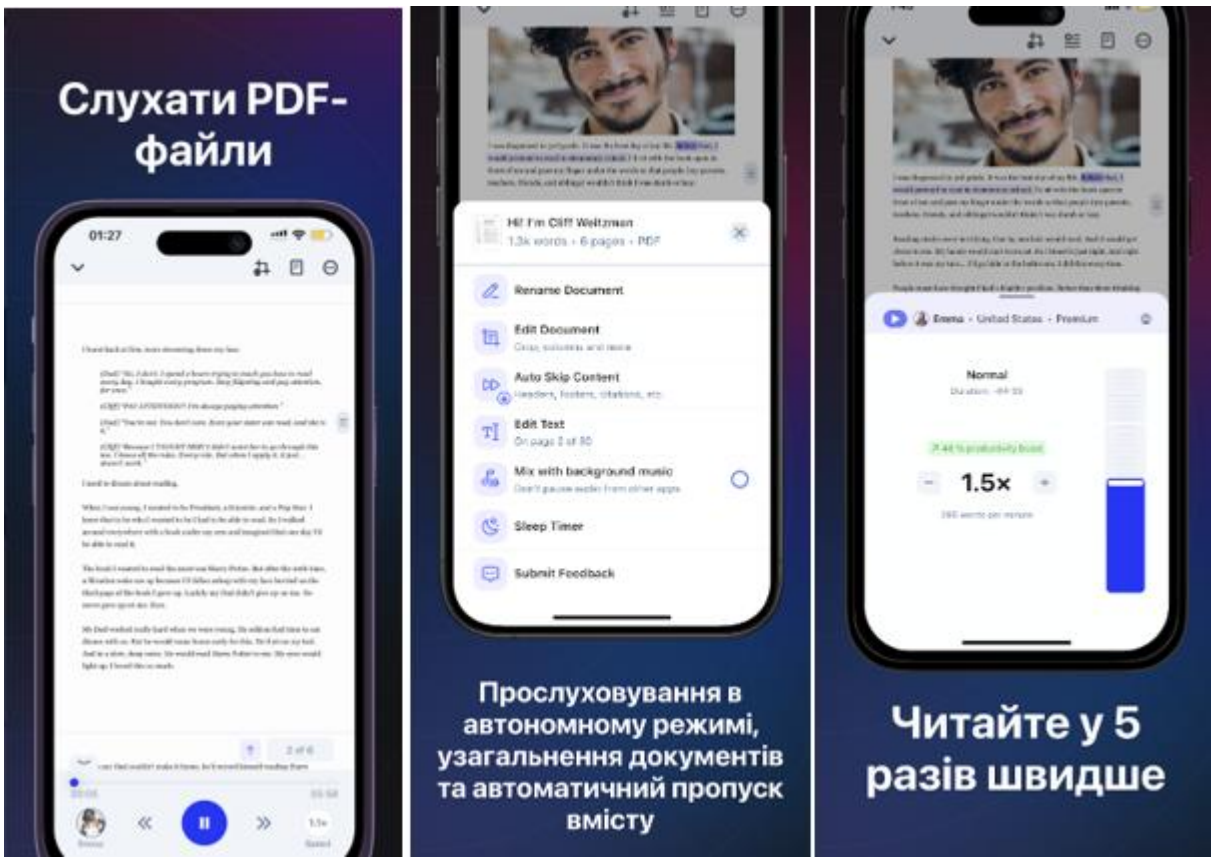

Рисунок 3 – Інтерфейс Speechify (за даними [4])

Ще однією особливістю Speechify є можливість імпортувати текст із різноманітних джерел, включно з PDF-файлами, вебсторінками та документами Google Docs. Ця функція робить додаток зручним для щоденного використання у навчанні, особливо серед студентів, які працюють із великими обсягами академічної інформації. Проте, на відміну від систем із повною інтеграцією TTS у середовище читання, у Speechify відсутня підтримка перегортання сторінок, вибіркового виділення фраз із миттєвим перекладом чи їхнього збереження у словник, що обмежує його функціональність як повноцінного інструменту для вивчення мов або глибокого опрацювання художнього тексту.

Додатковим недоліком є обмежена офлайн-функціональність. У безкоштовній версії більшість голосів працює тільки при активному з’єднанні з мережею, що не дозволяє використовувати додаток у ситуаціях з поганим або відсутнім інтернетом. Крім того, для роботи з іншими мовами, ніж англійська, потрібні додаткові мовні модулі, які часто недоступні без підписки.

Загалом, Speechify є прикладом потужного TTS-рішення, орієнтованого на конкретні аудиторії, проте не може повноцінно замінити універсальні платформи для читання й перекладу тексту. У цьому контексті розроблюваний додаток може пропонувати більш гнучку систему інтеграції озвучення, перекладу, читання та статистики в одному середовищі, що робить його ефективнішим для широкого кола користувачів із різними освітніми й комунікативними потребами.

Підсумовуючи, можна виділити декілька спільних технічних і концептуальних бар'єрів, з якими стикаються ML-інструменти в додатках для читання. По-перше, це велике навантаження на ресурси мобільних пристроїв — обробка озвучення та перекладу в реальному часі потребує значної обчислювальної потужності. По-друге, передача даних до хмарних сервісів викликає занепокоєння щодо конфіденційності, особливо при роботі з особистими або чутливими текстами. І нарешті, попри широку мовну підтримку, автоматизовані системи часто не враховують культурні й контекстуальні особливості мови, що є критичним для повного розуміння змісту.

1.3 Постановка задачі

Аналіз предметної галузі та дослідження сучасних мобільних додатків з функціями озвучування, перекладу та читання електронних книг дозволили виявити низку істотних проблем, які залишаються невирішеними або лише частково реалізованими у вже існуючих рішеннях. З огляду на ці обмеження, було сформульовано основну задачу кваліфікаційної роботи — створення мобільного додатка для читання електронних книг із підтримкою автоматичного озвучування тексту та перекладу окремих слів у реальному часі, що також забезпечує високу ступінь персоналізації, локалізації та інклюзивності.

Для досягнення поставленої мети необхідно вирішити такі ключові підзадачі:

– підтримка багатоформатного імпорту та читання електронних книг. Додаток повинен працювати з найбільш поширеними форматами електронних книг (EPUB, PDF, TXT тощо), а також забезпечувати можливість регулювання

параметрів відображення тексту: зміна шрифтів, кольорової теми, міжрядкових інтервалів тощо для покращення читабельності;

– реалізація автоматичного озвучування тексту з підтримкою кількох мов. Використання технологій синтезу мовлення (TTS) із застосуванням моделей машинного навчання та зовнішніх сервісів, що підтримують різні мови, з акцентом на правильну вимову, емоційність мовлення та налаштування голосу (швидкість, тембр, тональність). Озвучування має працювати в реальному часі без значних затримок;

– інтеграція функції перекладу окремих слів або фраз із застосуванням моделей машинного навчання та зовнішніх сервісів. Впровадження інтерфейсу, який дозволяє користувачеві перекладати вибрані фрагменти тексту «на льоту» без потреби перемикатися між програмами або вкладками. Важливо забезпечити точність перекладу та врахування контексту;

– інклюзивність для користувачів з особливими потребами. Забезпечення доступності інтерфейсу для людей із вадами зору або дислексією, наприклад, шляхом голосового управління, великих кнопок, читання вголос виділеного тексту тощо;

– персоналізація досвіду. Надання можливостей для індивідуального налаштування інтерфейсу, збереження історії прочитаного, налаштування мови та стилю озвучення, створення списку обраних книг, збереження улюблених слів тощо;

– оптимізація продуктивності та збереження приватності. Реалізація частини обробки (наприклад, переклад або озвучення) на пристрої користувача без відправлення тексту до сторонніх серверів, що дозволяє зменшити затримки та підвищити рівень конфіденційності.

Таким чином, задача кваліфікаційної роботи полягає у створенні кросплатформенного мобільного додатка нового покоління для роботи з електронними книгами, який не лише реалізує базові можливості читання, а й значно розширює їх за рахунок інтеграції технологій озвучення (TTS), автоматичного перекладу, персоналізованих налаштувань інтерфейсу та адаптації

до потреб різних користувачів. Розроблений додаток ReadEasy поєднує зручність класичного рідера з функціоналом мовного помічника, надаючи користувачеві повноцінний інструмент для взаємодії з текстом у динамічному, багатофункціональному середовищі.

Проєкт спрямований на підвищення доступності текстового контенту для людей із порушеннями зору, студентів, поліглотів та всіх, хто прагне ефективно працювати з літературою іноземними мовами. Завдяки гнучкій архітектурі, можливості офлайн-роботи та підтримці кастомізації зовнішнього вигляду, система охоплює широкий спектр сценаріїв використання — від навчання до дозвілля. Такий підхід не лише покращує користувацький досвід, а й сприяє розвитку інклюзивного цифрового середовища, що відповідає сучасним вимогам до універсального дизайну. У перспективі проєкт має потенціал до подальшої розробки, масштабування та комерційного впровадження.

1.4 Цільова аудиторія

Мобільний додаток буде орієнтований на широку аудиторію користувачів, що потребують зручного доступу до читання, перекладу та озвучення текстів. Основними цільовими групами є студенти, люди з вадами зору, поліглоти, викладачі та професіонали з технічних галузей. Найбільший акцент у першій версії буде зроблено на студентів, які вивчають іноземні мови, та користувачів із порушенням зору — для них забезпечено інтуїтивний інтерфейс, офлайн-режим, гнучке озвучення та адаптивний дизайн.

Для студентів і молоді додаток слугує інструментом для покращення словникового запасу, тренування вимови та зручного читання іноземних текстів. Користувачі з вадами зору отримують можливість повноцінно взаємодіяти з книжковим контентом завдяки підтримці TTS, великим шрифтам і контрастним темам. Поліглоти й мовні ентузіасти можуть використовувати додаток для читання оригінальних текстів та збереження перекладів до словника. Викладачі можуть застосовувати додаток для надання зручного представлення навчальним матеріалам. Професіонали, зокрема в галузі IT, можуть користуватись ним для

перекладу технічної документації або озвучення у фоновому режимі. Усі ці групи отримують переваги адаптивного, зручного та функціонального інструменту, який може розширюватися відповідно до потреб.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

На основі аналізу предметної області (розділ 1), сучасних тенденцій розвитку ринку електронних книг, технологій синтезу мовлення (TTS), а також функцій перекладу тексту в режимі реального часу було сформовано вимоги до функціональності, продуктивності та доступності мобільного додатка . Ці вимоги є результатом вивчення існуючих аналогів, виявлення їхніх обмежень та визначення потреб реальних користувачів, таких як студенти, викладачі, люди з порушеннями зору й мовні ентузіасти.

2.1 Цілі системи

Метою створення системи є розробка кросплатформенного мобільного додатка нового покоління, який об’єднує функції зручного читання електронних книг, озвучування тексту та перекладу слів і фраз у реальному часі. Основні цілі:

- забезпечити комфортне читання книг із широкими можливостями персоналізації (шрифт, інтервал, тема);
- реалізувати високоякісне синтезоване мовлення з вибором мови, тембру та швидкості;
- надати зручний інструмент для перекладу тексту без виходу з інтерфейсу читання;
- забезпечити підтримку офлайн-режиму та збереження прогресу користувача;
- створити доступний інтерфейс для людей із вадами зору.

2.2 Функціональні вимоги

Система повинна реалізовувати такі групи функціональностей:

- робота з книгами: імпорт із пам’яті або хмарних сховищ, підтримка форматів TXT, FB2, EPUB, MOBI, перегляд у вигляді сторінок;
- озвучення (TTS): читання окремих фрагментів або всього тексту голосом, налаштування параметрів озвучення;

– переклад тексту: переклад виділених слів чи фраз з можливістю прослуховування перекладу та збереження в словник;

– управління бібліотекою: перегляд, сортування, видалення та збереження книг;

– налаштування інтерфейсу: зміна теми, шрифту, кольорової схеми, міжрядкових інтервалів тощо;

– збереження прогресу: автоматичне збереження останньої прочитаної сторінки, налаштувань, статистики використання;

– офлайн-режим: можливість читати книги без підключення до Інтернету, з кешуванням попередніх результатів перекладу або TTS та використанням моделей ML для локальної реалізації функціоналу.

2.3 Нефункціональні вимоги

До нефункціональних вимог належать:

– продуктивність: швидкий запуск, плавне гортання сторінок, час очікування перекладу або озвучення — не більше 2 секунд;

– мобільність: підтримка Android-пристроїв з ОС версії 10.0 і вище та IOS з версією 14.0 і вище;

– адаптивність: підтримка різних розмірів екрана, масштабованість шрифтів, адаптивне компонування інтерфейсу;

– доступність: підтримка темної теми, великих шрифтів, елементів керування, що відповідають WCAG 2.1;

– безпека: зберігання персональних даних лише локально, шифрування конфіденційних налаштувань, використання HTTPS при роботі з API;

– надійність: автоматичне збереження стану, стійкість до обриву з’єднання, коректна робота в офлайн-режимі.

2.4 Обґрунтування технологічних рішень

Для розробки додатка обрано фреймворк React Native у поєднанні з мовою TypeScript, що дозволяє забезпечити кросплатформенність, швидкий час розробки

та зручність масштабування проєкту [5]. Збереження стану реалізовано через Zustand, що дозволяє гнучко організувати дані та легко управляти глобальним станом без перевантаження компонентів [6].

Для забезпечення функцій TTS і перекладу використовуються Google Text-to-Speech API та Google Translate API, що дозволяє досягти високої точності й якості мовного оброблення, для офлайн режиму використати локальні соделі цих сервісів.

Функції збереження прогресу, словника та налаштувань реалізовано через AsyncStorage, з попередньою фільтрацією контенту для зменшення навантаження на пристрій.

2.5 Висновки та посилання на специфікацію

Вимоги до системи формувалися на основі результатів аналізу предметної області, аналізу існуючих рішень і сценаріїв використання, а також очікувань цільових груп користувачів. Деталізований перелік функціональних, нефункціональних, інтерфейсних та системних вимог до додатка наведено у Додатку А — Специфікація вимог до програмного забезпечення (SRS).

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПЗ

3.1 UML проєктування ПЗ

У процесі розробки програмного забезпечення було створено UML-діаграму прецедентів (рис. 4), яка дозволяє формалізовано представити функціональні вимоги до системи та продемонструвати взаємодію користувача з основними компонентами додатку. Дана діаграма побудована відповідно до стандартів UML і виконує роль засобу візуалізації поведінки системи на ранніх етапах проєктування. Основним актором виступає користувач, який взаємодіє з додатком шляхом виконання різноманітних дій, таких як читання книги, керування бібліотекою, зміна налаштувань та інші функції, що забезпечують гнучке та інтуїтивне користування системою.

Центральним прецедентом є функція «Читати книгу», яка має декілька розширень і включень. Наприклад, користувач може перегортати сторінки, додавати закладки, імпортувати книги з пристрою, а також озвучувати текст або перекладати його. Можливість перекладу тексту, у свою чергу, розширюється додатковими функціями — користувач може додати перекладене слово в словник або прослухати переклад. Додатково передбачено керування відтворенням: перемотування, пауза, відтворення. Інший важливий прецедент — «Керувати бібліотекою», що включає в себе пошук, сортування, фільтрацію книг, а також можливість їх видалення.

Окремий блок функціональності пов’язаний з налаштуванням програми. Користувач має змогу змінювати загальні налаштування, шрифт, розмір тексту, тему інтерфейсу, параметри озвучення та мову перекладу. Передбачена також функція скидання до стандартних параметрів, яка розширює основний прецедент зміни налаштувань. Усі залежності між прецедентами оформлені відповідно до правил моделювання UML з використанням включення (include) та розширення (extend), що дозволяє відобразити умовну або обов’язкову природу взаємозв’язків між функціями. Діаграма прецедентів стала ключовим інструментом для

формування чіткої архітектури майбутньої системи та забезпечила базу для подальшої деталізації технічного проєктування.

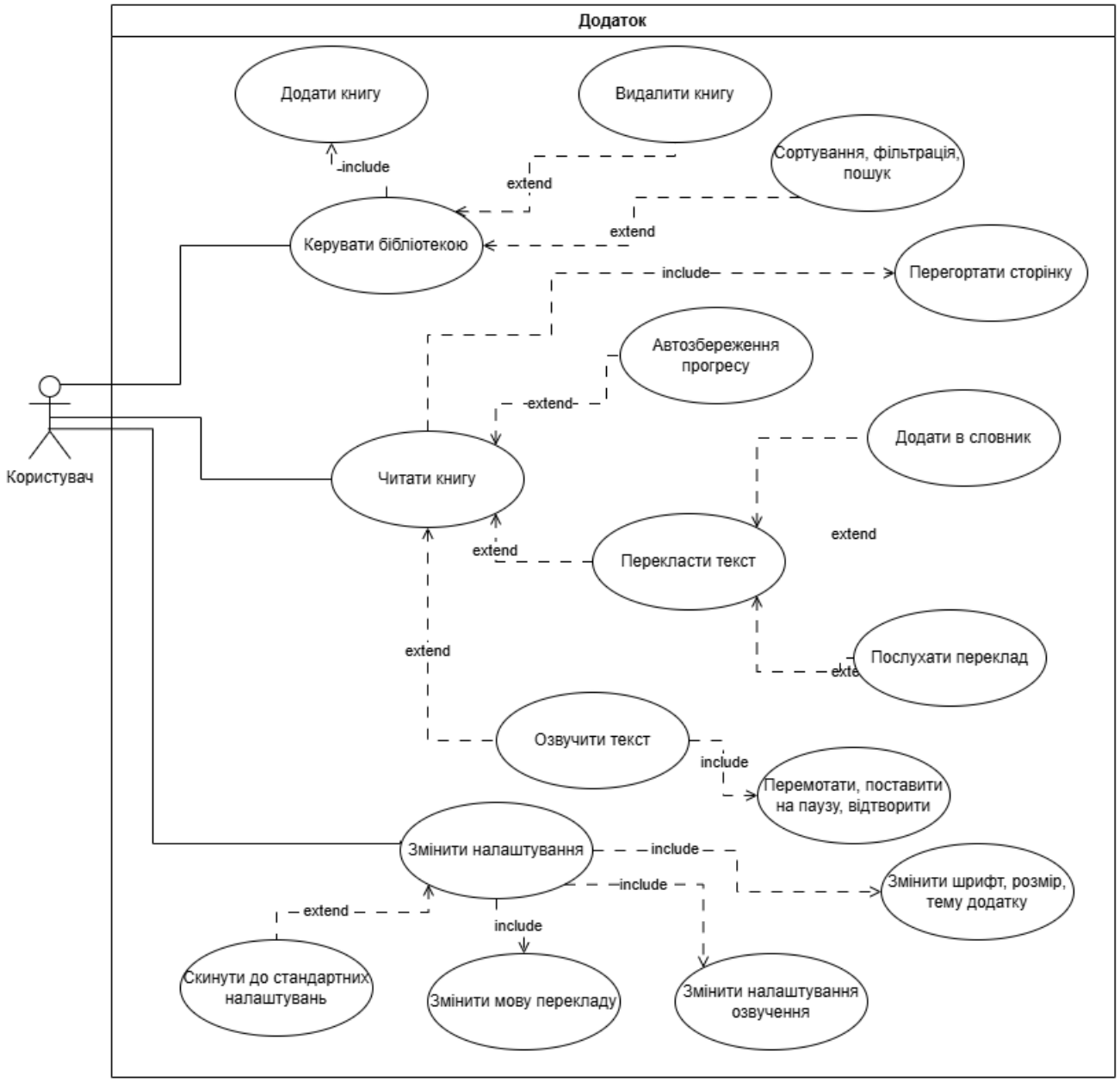

Рисунок 4 – UML діаграма прецедентів (рисунок виконаний самостійно)

У рамках мобільного додатка книга як об’єкт має чітко визначений життєвий цикл, що моделюється за допомогою діаграми станів (рис. 5). Початковим станом є Не відкрита, що означає, що книга ще не була обрана користувачем. Після вибору книги система переходить у стан Відкрита, де завантажується інтерфейс читання. У робочому режимі додаток перебуває у стані У процесі читання, що є основним.

Із цього стану можливі переходи до На паузі (наприклад, при згортанні додатка), до Озвучення (якщо активовано функцію TTS), або до Переклад (при виділенні слова чи фрази для перекладу). Після завершення відповідної дії або повернення користувача, система повертається до стану активного читання. Вихід з книги призводить до переходу у фінальний стан Закрита, після чого книга знову вважається Не відкритою. Така модель дозволяє чітко визначити логіку обробки дій користувача та впровадити адаптивну поведінку інтерфейсу.

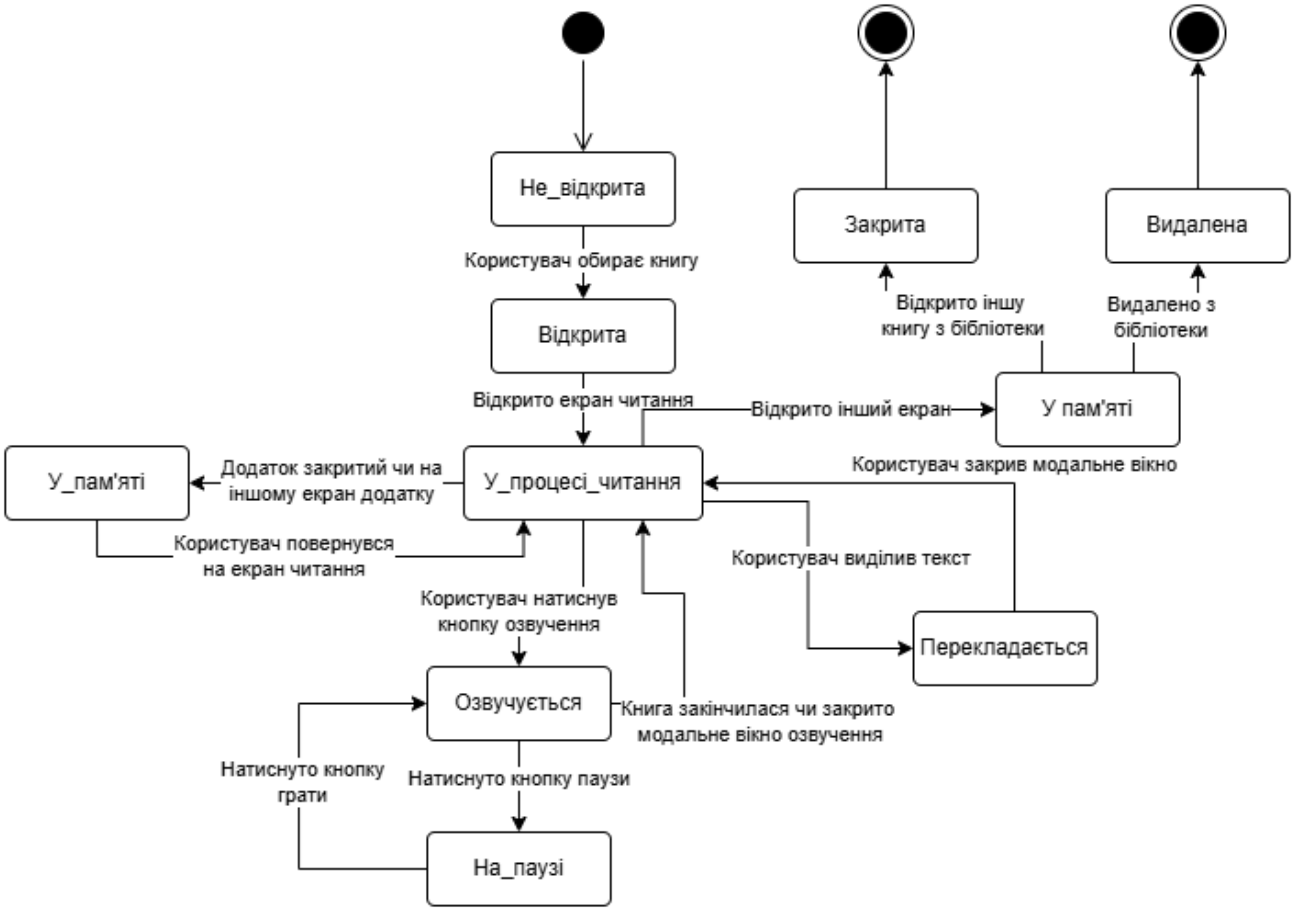

Рисунок 5 – Діаграма станів книги (рисунок виконаний самостійно)

У процесі проєктування програмної системи також було створено UML-діаграму розгортання (рис. 6), яка відображає фізичну структуру програмного забезпечення та середовищ, у яких воно виконується. Основною платформою є Android-пристрій користувача, де розгортається мобільний додаток ReadEasy, створений за допомогою React Native у середовищі Expo. Усі логічні компоненти

(читання, керування бібліотекою, налаштування, збереження прогресу тощо) виконуються локально, а дані зберігаються в пам’яті пристрою через AsyncStorage.

У випадку онлайн-режиму додаток звертається до зовнішніх API-сервісів: Google Cloud TTS для озвучення та DeepL API для перекладу виділених слів і фраз. Також є нативна можливість завантажувати файли з Google Диску. Для забезпечення автономної роботи без підключення до Інтернету інтегровано Google ML Kit Translation, що функціонує як нативний Android-модуль а також використовуються вбудовані можливості пристрою для TTS. Таким чином, система підтримує гібридну архітектуру, де основна логіка працює локально, а мережеві функції використовуються лише за необхідності.

Діаграма розгортання дозволяє наочно уявити, як саме реалізована взаємодія між локальними та хмарними компонентами, яким чином забезпечується масштабованість, доступність функцій офлайн, та які саме залежності існують між модулями. Такий підхід дозволяє спроєктувати стійку до збоїв систему з високим рівнем автономності.

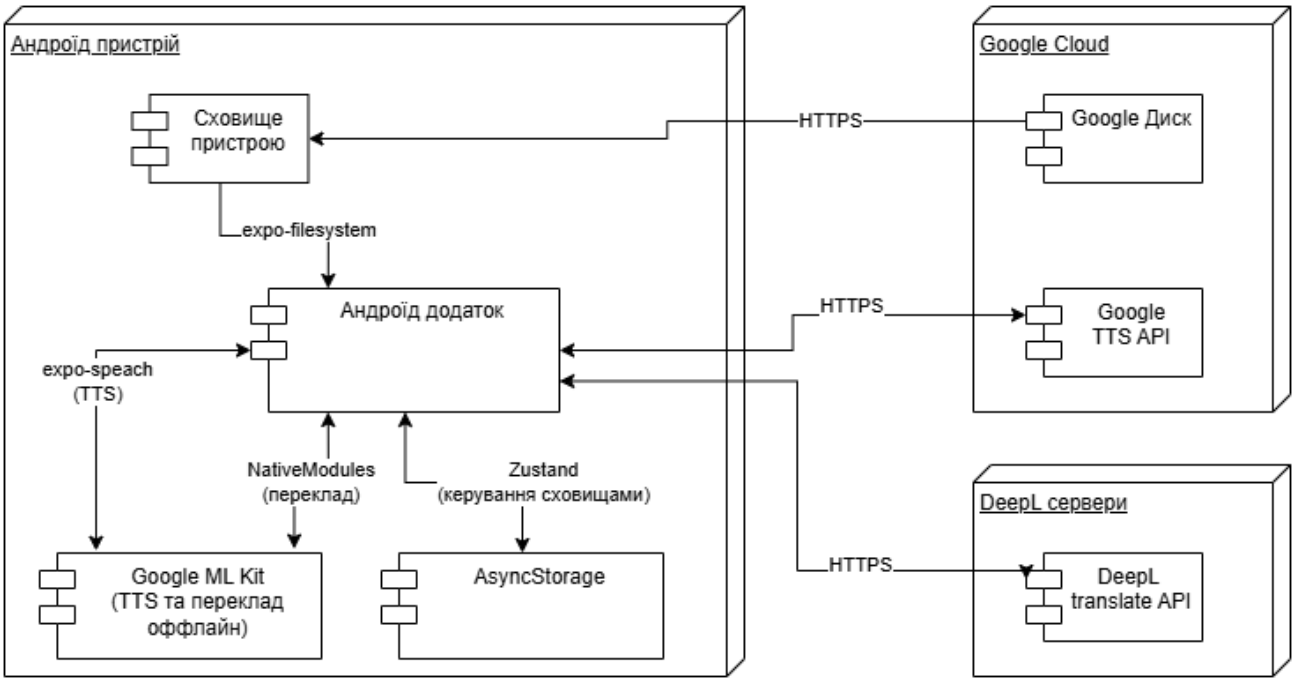

Рисунок 6 – UML діаграма розгортання (рисунок виконаний самостійно)

3.2 Проектування архітектури ПЗ

Архітектура програмного забезпечення мобільного додатка побудована з урахуванням вимог до масштабованості, модульності та зручності розробки функціонального інтерфейсу для читання, перекладу й озвучення електронних книг. Оскільки клієнтська частина розробляється на базі React Native з використанням TypeScript, особливу увагу приділено організації коду відповідно до підходу Feature-Sliced Design (FSD) — сучасної архітектурної методології, що дозволяє логічно розділити код за функціональністю, а не за технічною роллю. Такий підхід забезпечує кращу підтримуваність, масштабованість і ізоляцію модулів при зростанні проєкту [7].

У структурі проєкту всі модулі поділяються на шари: app, widgets, features, entities, shared (рис. 7). Це дозволяє чітко відокремити глобальну логіку (наприклад, навігацію чи типізацію) від бізнес-функцій (як-от озвучення тексту, управління книгами або переклад) [8]. Зокрема, функції озвучення реалізовані у вигляді незалежного feature-модуля, що використовує стороннє API Google Text-to-Speech, а переклад — через Google Translate API. Також передбачено використання моделей ML для цих функцій у офлайн режима. Обидва сервіси взаємодіють із зовнішнім середовищем через асинхронні entity-модулі, що відповідають за обробку запитів, кешування результатів і обробку помилок.

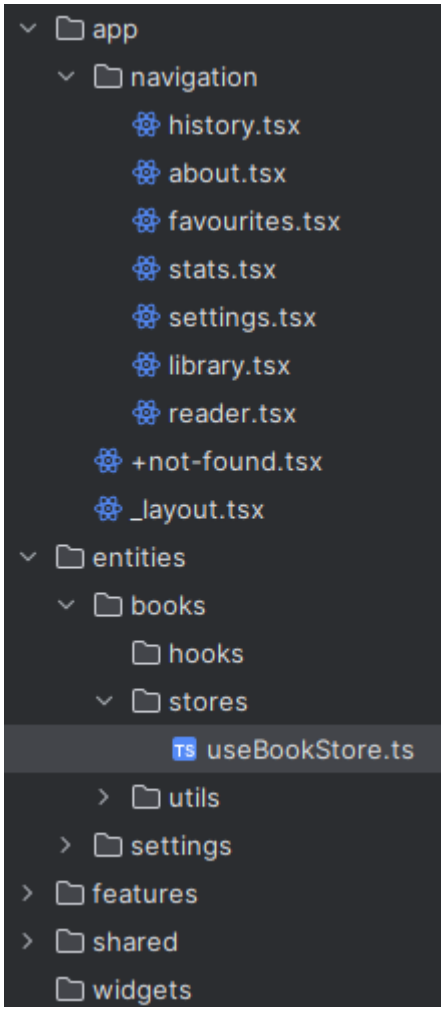

Рисунок 7 – Приклад структури FSD у проєкті (рисунок виконаний самостійно)

Інтерфейс користувача реалізовано у вигляді ізольованих компонентів, згрупованих за сторінками — наприклад, сторінка читання книги, словник, налаштування, бібліотека. Всі екрани проектуються із врахуванням принципів доступності та підтримки великого шрифту, темної теми та адаптивності. Для реалізації таких можливостей використовуються стилі, побудовані на основі React Native Style [9].

Збереження даних реалізовано з використанням AsyncStorage для локального кешу: зокрема, зберігається остання прочитана сторінка, список перекладених слів, мовні налаштування та параметри озвучення. У перспективі можливе доповнення локальної бази даних SQLite для складніших сценаріїв.

Для обробки стану застосовуються контекстні провайдери та локалізоване управління станом у межах features, відповідно до принципів FSD. Такий підхід

дозволяє уникнути глобального стану, зменшити зв’язність між модулями та спростити тестування. Якщо виникає потреба в глобальній синхронізації (наприклад, для теми або поточного користувача), використовується Zustand.

Для інтеграції з API використовуються стандартні HTTP-клієнти (зокрема, axios), які інкапсулюються в окремі сервіси у шарі shared/api. Кожен сервіс відповідає за обробку запитів до конкретного провайдера, включаючи авторизацію, таймаути, повтори запитів та обробку помилок.

Особливу увагу приділено роботі в умовах нестабільного інтернету. Додаток підтримує офлайн-режим, у якому можливо читати попередньо завантажені книги, зберігати переклади та використовувати локальну модель для перекладу та працювати з локальною моделлю озвучення. Усі функції перевіряють наявність мережі перед виконанням запитів, а критичні дані кешуються для подальшої синхронізації.

Таким чином, архітектура мобільного додатка ґрунтується на принципах модульності, ізоляції функцій та адаптивного інтерфейсу, реалізованого через Feature-Sliced Design. Такий підхід дозволяє ефективно впроваджувати та масштабувати ключові функції додатку — читання, переклад, озвучення — забезпечуючи при цьому стабільну роботу на Android-пристроях навіть у складних умовах використання.

3.3 Проектування структури зберігання даних

У мобільному додатка зберігання даних реалізовано на основі асинхронного ключ-значення сховища AsyncStorage, що є стандартним API для React Native. Для зручної та централізованої роботи зі сховищем створено утиліту storageUtils, яка інкапсулює всі базові операції: зчитування (get), запис (set), видалення (remove) та повне очищення (clear) даних. Завдяки цьому додаток може ефективно керувати станом між сесіями, зберігати прогрес читання, налаштування інтерфейсу та метадані книг.

Особливістю реалізації є селективне зберігання інформації про книги. Щоб уникнути зайвого дублювання та зменшити обсяг даних у AsyncStorage, під час

виклику set() система автоматично фільтрує вхідний об’єкт книги. Якщо книга містить поле content (тобто повний текст), у сховище зберігаються лише її метадані: ідентифікатор (id), назва (name), тип (type), поточна сторінка (currentPage), загальна кількість сторінок (totalPages), шлях до локального файлу (filePath) та дата останнього відкриття (lastRead). Це дозволяє значно зменшити навантаження на пристрій і мінімізувати ризик перевищення ліміту на розмір сховища.

Уся логіка управління станом, включно з викликами до storageUtils, організована в рамках глобального сховища стану за допомогою бібліотеки Zustand. Це забезпечує реактивність компонентів інтерфейсу: зміни, що відбуваються у збережених об’єктах (наприклад, оновлення сторінки або налаштувань), автоматично відображаються у відповідних екранах без необхідності ручного оновлення.

Ключові типи даних, що зберігаються:

```
// Зберігає дані перекладів користувача
export interface TranslationData {
  translations: Translation[]; // Масив усіх збережених перекладів
}

// Окремий переклад тексту з книги
export interface Translation {
  id: string; // Унікальний ідентифікатор перекладу
  bookName: string; // Назва книги, з якої зроблено переклад
  originalText: string; // Оригінальний текст для перекладу
  translatedText: string; // Перекладений текст
  timestamp: number; // Час створення перекладу (Unix timestamp)
}

// Статистика читання для конкретної книги
export interface BookStatistics {
  bookId: string; // Унікальний ідентифікатор книги
  bookName: string; // Назва книги
  totalReadingTime: number; // Загальний час читання в секундах
  lastReadDate: number; // Дата останнього читання (Unix timestamp)
  pagesRead: number; // Кількість прочитаних сторінок
  totalPages: number; // Загальна кількість сторінок у книзі
  readingSessions: { // Масив сесій читання
    startTime: number; // Час початку сесії (Unix timestamp)
    endTime: number; // Час закінчення сесії (Unix timestamp)
    pagesRead: number; // Кількість сторінок, прочитаних за сесію
  }[];
}

// Зберігає статистику читання для всіх книг
export interface BookStatisticsData {
```

```
  statistics: Record<string, BookStatistics>; // Об'єкт з статистикою, де
ключ - ID книги
}

// Налаштування додатка
export interface Settings {
  theme: Theme; // Тема оформлення (світла/темна)
  fontSize: number; // Розмір шрифту тексту
  fontFamily: FontFamily; // Сімейство шрифту
  lineSpacing: number; // Міжрядковий інтервал
  marginWidth: number; // Ширина полів сторінки
  interfaceLanguage: Language; // Мова інтерфейсу додатка
  voiceLanguage: Language; // Мова для озвучування тексту
  translationLanguage: Language; // Мова для перекладу тексту
  voiceSpeed: number; // Швидкість озвучування (множник)
  voicePitch: number; // Висота тону голосу
}

// Дані про книги в додатку
export interface BookData {
  books: Book[]; // Масив усіх доступних книг (в store зберігається як
BookMetadata[])
  currentBook: Book | null; // Поточно відкрита книга
}

// Повна інформація про книгу
export interface Book {
  id: string; // Унікальний ідентифікатор книги
  name: string; // Назва книги
  type: string; // Тип файлу (mobi, fb2, epub, txt)
  content?: string; // Текстовий вміст книги (не зберігається персистивно)
  fileSize: number; // Розмір файлу в байтах
  currentPage?: number; // Поточна сторінка читання
  totalPages?: number; // Загальна кількість сторінок
  pages?: string[]; // Масив сторінок книги (не зберігається персистивно)
  filePath?: string; // Шлях до файлу книги
  lastRead?: string; // Дата останнього читання (рядок)
  isFavorite?: boolean; // Чи додана книга до улюблених
}

// Метадані книги без контенту (зберігається персистивно в useBookStore)
export type BookMetadata = Omit<Book, 'content' | 'pages'>;
```

Таким чином, архітектура зберігання в додатку відповідає принципам локальної ефективності, безпечної серіалізації та розділення даних на ключові категорії. Завдяки цьому додаток здатен швидко відновлювати сесію після перезапуску, працювати в офлайн-режимі та забезпечувати безперервну взаємодію користувача з контентом.

Код логіки управління зберіганням стану:

```
export const storageUtils = {
  async set(key: string, value: any): Promise<void> {
    try {
```

```
      // For books, only save metadata and file path
      if (value.content) {
        const bookMetadata = {
          id: value.id,
          name: value.name,
          type: value.type,
          currentPage: value.currentPage || 0,
          totalPages: value.totalPages || 0,
          filePath: value.filePath, // Path to the book file on device
          lastRead: new Date().toISOString(),
        };
        await AsyncStorage.setItem(key, JSON.stringify(bookMetadata));
      } else {
        await AsyncStorage.setItem(key, JSON.stringify(value));
      }
    } catch (error) {
      console.error('Error saving data:', error);
    }
  },

  async get(key: string): Promise<any> {
    try {
      const jsonValue = await AsyncStorage.getItem(key);
      return jsonValue != null ? JSON.parse(jsonValue) : null;
    } catch (error) {
      console.error('Error reading data:', error);
      return null;
    }
  },

  async remove(key: string): Promise<void> {
    try {
      await AsyncStorage.removeItem(key);
    } catch (error) {
      console.error('Error removing data:', error);
    }
  },

  async clear(): Promise<void> {
    try {
      await AsyncStorage.clear();
    } catch (error) {
      console.error('Error clearing data:', error);
    }
  },
};
```

3.4 Приклади найцікавіших алгоритмів та методів

Одним із ключових алгоритмів, реалізованих у мобільному додатка , є алгоритм пагінації тексту книги відповідно до параметрів візуалізації: розміру шрифту, міжрядкового інтервалу, ширини відступів та фактичних розмірів екрана

пристрою. Завдяки цьому користувач бачить текст, який точно заповнює екран, незалежно від пристрою, роздільної здатності чи обраних налаштувань читання.

Першим етапом є вимірювання доступної площі для тексту. Це виконується з урахуванням висоти службових елементів інтерфейсу (заголовок, нижній колонтитул, поля) та розрахунку кількості рядків, які можуть поміститися на екрані. Кожен рядок має висоту, яка обчислюється як добуток розміру шрифту на міжрядковий інтервал, а ширина кожного рядка визначається з урахуванням горизонтальних відступів (marginWidth).

Далі виконується перенесення тексту по ширині. Алгоритм обробляє текст параграфами та розбиває його на слова. Кожне слово вимірюється на основі його символів, з використанням емпіричних коефіцієнтів ширини (для стандартних та великих літер). Якщо додавання чергового слова перевищує максимальну ширину рядка, створюється новий рядок. Особливої уваги заслуговує ситуація, коли слово не вміщується повністю — воно розбивається на частини, що поміщаються в рядки, без втрати змісту. Також реалізовано автоматичне додавання абзацного відступу (три пробіли) на початку нового абзацу.

На завершальному етапі обробки сформовані рядки об'єднуються у сторінки. Коли кількість рядків перевищує допустиму висоту екрана (linesPerPage), створюється нова сторінка. Це дозволяє сформувати послідовний масив текстових сторінок, який зручно відображати та зберігати для подальшого навігаційного рендерингу. Усі розрахунки враховують реальні налаштування користувача, зчитані з локального сховища, що дозволяє відображати контент відповідно до індивідуальних уподобань. Опрацьований результат зберігається у локальному кеші (AsyncStorage) разом з кількістю сторінок, останньою прочитаною позицією та повним текстом, що дозволяє продовжити читання з будь-якого моменту без повторного оброблення тексту.

Такий підхід забезпечує гнучку, адаптивну й ефективну систему виводу тексту, зберігаючи при цьому високу продуктивність навіть на пристроях із обмеженими ресурсами. Алгоритм є основою зручного інтерфейсу читання,

дозволяє підтримувати різні мови, абзацну структуру та особливості форматування електронних книг.

Код алгоритму наведено у Додатку Б.

Функціональність екрана читання в додатка реалізована за допомогою кастомного React-хука, що інтегрує керування станом книги, обробку користувацьких налаштувань та анімацію жестів для гортання сторінок. Центральним елементом є взаємодія з глобальним сховищем стану useBookStore, з якого отримуються поточна книга, методи завантаження контенту та оновлення сторінок, а також функція для зміни активної сторінки. Налаштування візуалізації (розмір шрифту, міжрядковий інтервал, ширина відступів) підключаються з useSettingsStore, що дозволяє автоматично адаптувати текст при зміні будь-якого параметра [10].

Після першого монтування компонента викликається loadLastBook() — функція, що ініціалізує останню відкриту книгу з локального сховища. Далі, при наявності шляху до файлу (book.filePath), запускається loadBookContent() — асинхронна функція, яка завантажує текст книги для подальшої обробки. При зміні налаштувань тексту виконується updatePages(), який повторно запускає алгоритм пагінації та формує новий набір сторінок відповідно до обраних користувачем параметрів.

Стан поточної сторінки (currentPage) зберігається локально у компоненті та синхронізується із глобальним станом при кожному переході між сторінками. Логіка зміни сторінки інкапсульована в методі changePage(direction), що враховує допустимий діапазон сторінок та виконує оновлення тільки у разі наявності книги й коректного значення індексу.

Особливістю реалізації є підтримка жестів свайпу для гортання сторінок. Використовуючи бібліотеку react-native-gesture-handler, створено обробник gesture, який реагує на горизонтальні свайпи [11]. Якщо жести перевищують певний поріг (чверть ширини екрана), викликається відповідна зміна сторінки за допомогою runOnJS(changePage). Для плавності використовується анімація withSpring() з

бібліотеки react-native-reanimated, яка повертає текст у вихідне положення при скасуванні дії.

Уся ця логіка забезпечує інтерактивний, плавний і контекстно-залежний досвід читання, при якому зміни вмісту, налаштувань та навігації відбуваються автоматично, без перезавантажень або затримок. Це дозволяє створити ефективний і гнучкий інтерфейс, що відповідає сучасним вимогам до мобільного UX для електронних читалок.

Код хука наведено у Додатку В.

3.5 Створення UI/UX

Дизайн інтерфейсу мобільного додатка розроблений відповідно до принципів сучасного UI/UX-дизайну, з фокусом на зручність, доступність і персоналізацію досвіду читання. Оскільки головною цільовою аудиторією є студенти, викладачі, мовні ентузіасти та користувачі з порушенням зору, особливу увагу було приділено адаптивності інтерфейсу, простоті взаємодії та підтримці жестів.

Процес проєктування починався з аналізу контекстних сценаріїв використання (читання з перекладом, слухання книги, інтерактивна робота зі словником) та побудови портретів користувачів, що дозволило сформувати набір ключових вимог до інтерфейсу.

Інтерфейс реалізований як набір екранів (бібліотека, сторінка читання, історія перекладів, налаштування), кожен із яких має чітко визначену мету та набір доступних дій. Екран читання підтримує горизонтальну навігацію між сторінками свайпом, мінімалістичний вигляд без зайвих елементів, а також можливість перекладу виділеного тексту або його озвучення одним натисканням (рис. 8). Дизайн враховує адаптивність під різні розміри екранів — на менших пристроях елементи інтерфейсу автоматично масштабуються, а всі основні дії доступні жестами однією рукою.

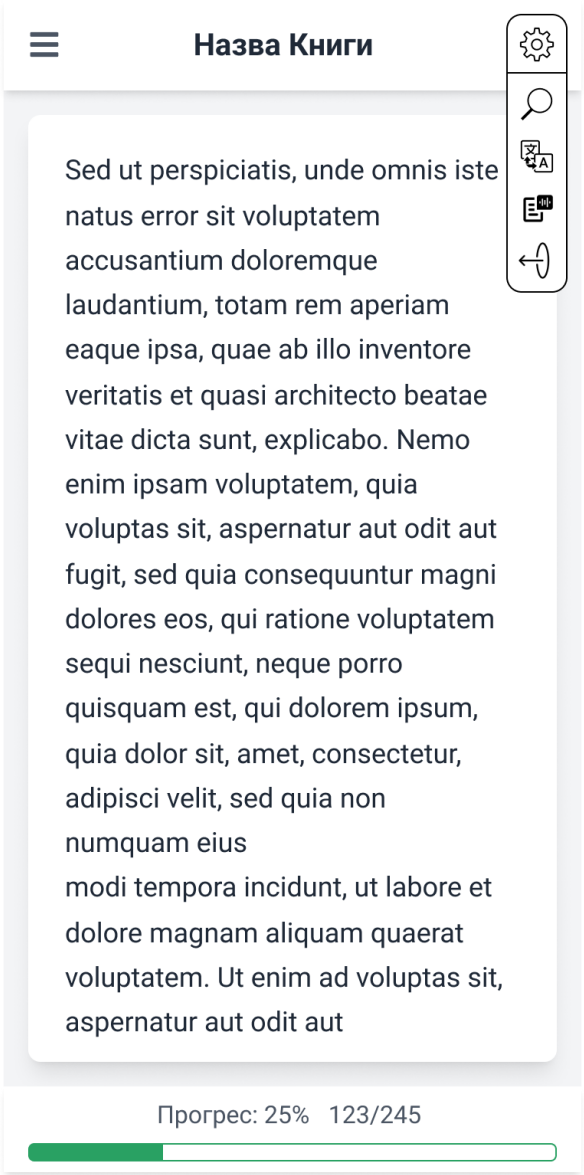

Рисунок 8 – Дизайн вікна читання (рисунок виконаний самостійно)

Завдяки реалізації темної та світлої теми, користувачі можуть налаштовувати інтерфейс під власні потреби, зменшуючи навантаження на очі при читанні в темряві. Додатково реалізовано індивідуальні налаштування шрифту, міжрядкового інтервалу, кольору фону й полів, що робить досвід максимально персоналізованим.

При створенні інтерфейсу були враховані вимоги доступності (accessibility) — великі натискні області, підтримка масштабованого шрифту, контрастні кольори відповідно до WCAG 2.1, можливість озвучення інтерфейсу за допомогою служб Android Accessibility [12]. Всі візуальні елементи мають чітку ієрархію та не перевантажують користувача інформацією.

Для реалізації адаптивного компонування інтерфейсу використано Flexbox і модульну сітку, що дозволяє гнучко розташовувати елементи незалежно від орієнтації екрана чи вибраних параметрів шрифту (рис. 9). Основна навігація в додатку реалізована через бокову панель, яка відкривається жестом або кнопкою меню та містить основні пункти: «Бібліотека», «Читання», «Про книгу», «Історія», «Статистика», «Налаштування», «Про додаток». Такий підхід дозволяє зберігати максимальний простір для читання, а також забезпечує швидкий доступ до функціональності без перевантаження головного інтерфейсу. Усі основні дії доступні в межах одного-двох натискань, що відповідає принципам сучасного мобільного UX.

Таким чином, дизайн поєднує функціональну простоту, адаптивність і гнучкість. Він враховує специфіку читання з мобільного пристрою, підтримує інклюзивність та дає змогу кожному користувачу налаштувати додаток під свої індивідуальні потреби.

Рисунок 9 – Адаптивне компонування інтерфейсу на екрані бібліотеки (рисунок виконаний самостійно)

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Вибір середовища та засобів розробки

Для реалізації мобільного додатка було обрано кросплатформену технологію розробки React Native, яка дає змогу створювати додатки для Android з єдиною кодовою базою, використовуючи сучасні веб-технології. Також реактивність React дозволяє використовувати нативно шаблон «Observer», що значно полегшує взаємодію між компонентами додатку. Основною мовою програмування є TypeScript, що дозволяє отримати всі переваги статичної типізації: кращу підтримку під час розробки, вищу надійність коду, попередження помилок ще до виконання програми.

Для швидкого запуску та полегшеної конфігурації середовища було використано платформу Expo, яка спрощує збірку, тестування та розгортання додатків на React Native. Expo дозволяє розробнику уникнути складної нативної конфігурації Android-проєкту на перших етапах, забезпечуючи одночасно доступ до апаратних функцій, таких як файлове сховище, доступ до мережі або розміри екрану.

Стан додатку управляється через zustand — легку та продуктивну бібліотеку для керування глобальним і локальним станом без складного шаблонного коду. Бібліотека дозволяє використати шаблон проєктування «Singleton», який забезпечує глобальний доступ до стану. На відміну від Redux, zustand не потребує окремої конфігурації сторів або додаткових обгорток. Це дозволяє зберігати та оновлювати стан, пов’язаний із книгою, сторінками, поточною позицією користувача, налаштуваннями та словником у компактному, декларативному вигляді.

Для реалізації жестів користувача, зокрема горизонтального гортання сторінок, використано бібліотеку react-native-gesture-handler, яка забезпечує продуктивну обробку свайпів і жестів, включно з кастомною анімацією та підтримкою подій дотику з високою точністю. Це критично важливо для UX при читанні, де очікується плавна й чутлива навігація між сторінками.

Загалом, вибір інструментів розробки продиктований потребою в гнучкості, адаптивності та можливості легко масштабувати додаток у майбутньому. Завдяки використанню сучасного технологічного стеку вдалося досягти швидкої інтерактивності інтерфейсу, надійності при збереженні даних та гарного користувацького досвіду при читанні й роботі з контентом.

4.2 Керування логікою екрана читання

Одним із центральних елементів додатка є екран читання, який реалізує складну інтерактивну логіку для відображення тексту, гортання сторінок, озвучення вмісту, перекладу виділених фрагментів, підсвічування слів і адаптації до користувацьких налаштувань. Його реалізація ґрунтується на поєднанні хуків React, керування станом через Zustand, а також gesture-обробки з використанням react-native-gesture-handler та анімації з react-native-reanimated.

Під час монтування компонента ініціалізується завантаження останньої прочитаної книги та її вмісту. Якщо вміст присутній, виконується динамічне розбиття тексту на сторінки відповідно до актуальних налаштувань користувача (розмір шрифту, міжрядковий інтервал, поля). Цей процес використовує попередньо розроблений алгоритм пагінації, що визначає кількість рядків на сторінці та ширину для обтікання тексту.

Гортання сторінок реалізується за допомогою горизонтального свайпу — розпізнавання жесту здійснюється за допомогою Pan-жесту (Gesture.Pan()), а зміна сторінки анімовано виконується з ефектом повернення (withSpring). Навігація між сторінками синхронізується зі станом через Zustand, що дозволяє зберігати поточну позицію навіть після перезапуску додатка .

Особливу увагу приділено логіці озвучення: система підтримує як онлайн TTS (через Google Text-to-Speech API[13] з попереднім кешуванням озвучення у .mp3), так і офлайн TTS (через expo-speech). Після натискання кнопки «play» визначається доступність мережі, і залежно від цього обирається відповідна стратегія. У разі офлайн-режиму, текст озвучується по словах із можливістю зупинки, відновлення і переходу на наступну сторінку. В процесі озвучення слово,

що зараз читається, підсвічується кольором, синхронізовано з обчисленням charIndex та onBoundary.

Також реалізовано можливість перекладу тексту. При виділенні окремих слів або групи слів (через tap або жест виділення) виконується, у відповідності до доступності мережі, виклик запиту до API DeepL[14], або кастомного JavaScript-модуля Translator на нативному рівні, який повертає переклад обраного фрагмента за допомогою потужностей моделі машинного перекладу від Google. Переклад відображається в нижній частині екрана, яка динамічно відкривається в режимі перекладу або озвучення.

Відображення тексту реалізується через сегментацію вмісту на окремі слова, які рендеряться як окремі компоненти ThemedText. Це дозволяє гнучко керувати стилями: підсвічування виділених слів, фоновий колір слова, що озвучується, та підтримка переносу між абзацами. Система підтримує вставку маркерів нового абзацу (\newLine) з відступом, а також забезпечує належне масштабування при зміні розміру шрифту або орієнтації екрана.

Таким чином, логіка екрана читання в додатку поєднує інтерактивність, персоналізацію, багатофункціональність і доступність, забезпечуючи глибоку інтеграцію між інтерфейсом користувача, налаштуваннями, мовними сервісами та локальним зберіганням прогресу.

4.3 Створення UI/UX або іншого дизайну системи

Інтерфейс мобільного додатка ReadEasy проєктувався відповідно до принципів сучасного UI/UX-дизайну: мінімалізм, зручність, адаптивність і доступність. Оскільки основна мета — забезпечити комфортне читання та взаємодію з текстом, ключовим пріоритетом стало фокусування на контенті, а не на інтерфейсних елементах.

Глобальна навігація реалізована через Drawer-навігацію (бічне меню) на базі expo-router/drawer. Меню відкривається за допомогою кнопки, і містить ключові пункти: «Бібліотека», «Читання», «Про книгу», «Переклади», «Статистика», «Налаштування», «Про додаток»

Drawer-інтерфейс дозволяє зберегти максимум екранного простору для читання, а також забезпечує швидкий доступ до функцій озвучення та перекладу через кнопки в заголовку екрана «Читання». Це відповідає принципу скорочення кількості дій до 1–2 натискань для основних сценаріїв.

Інтерфейс адаптується до кольорової теми користувача (світла або темна), яку реалізовано через власний контекст ThemeProvider і хук useTheme. Візуальні компоненти автоматично перебарвлюються залежно від обраної теми, зокрема:

- колір фону, тексту, активних та неактивних пунктів меню;
- стилі заголовків і кнопок;
- активні підсвічування в навігації.

Це підвищує комфорт читання в різних умовах освітлення та відповідає стандартам доступності.

Компонування побудовано за допомогою Flexbox (GestureHandlerRootView, View) з автоматичною адаптацією до розміру екрана. Використовується система шрифтів, підключених через expo-font, а також автоматичне масштабування через useWindowDimensions.

Шрифт, інтервал, поля та кольорова гама сторінки повністю налаштовуються у розділі «Налаштування». Завдяки цьому інтерфейс персоналізується під потреби людей із порушенням зору, дислексією або індивідуальними читацькими звичками.

4.4 Реалізація алгоритмів озвучення та перекладу

Для забезпечення озвучення тексту та перекладу в реальному часі в мобільному додатка реалізовано гібридну модель роботи з мовними сервісами, яка дозволяє перемикатися між онлайн- і офлайн-режимами в залежності від наявності інтернету. Це забезпечує високу доступність і стабільність додатка навіть за умов обмеженого підключення.

4.4.1 Онлайн-озвучення (Google Cloud Text-to-Speech API)

Для створення високоякісного озвучення в онлайн-режимі додаток звертається до Google Cloud Text-to-Speech API, який генерує синтезоване мовлення на базі нейромереж. Особливості реалізації:

– запит: надсилається HTTP POST-запит на адресу https://texttospeech.googleapis.com/v1/text:synthesize, із передачею тексту, параметрів голосу та мови (наприклад: uk-UA-Chirp3-HD-Achenar), а також конфігурації озвучення (MP3, швидкість, висота тону тощо);

– авторизація: запит використовує API-ключ Google Cloud із доступом до texttospeech.googleapis.com;

– результат: API повертає поле audioContent у форматі Base64, яке зберігається на пристрої як .mp3-файл за допомогою expo-file-system, а потім відтворюється через expo-audio;

– переваги: висока якість озвучення, підтримка десятків мов і діалектів, можливість кастомізації голосів;

– обмеження: вимагає активне з’єднання з Інтернетом і має ліміти на кількість запитів у безплатному тарифі.

4.4.2. Офлайн-озвучення (expo-speech)

Коли інтернет відсутній або API дає збій, додаток переходить у режим локального озвучення за допомогою бібліотеки expo-speech, яка використовує вбудовані можливості Android:

– Інтерфейс: Speech.speak(text, options) з callback-подіями onDone, onStopped, onBoundary.

– Мова: вибір мови голосу залежить від доступних голосів ОС (наприклад, український TTS не завжди попередньо встановлений).

– Контроль: система підтримує паузу, зупинку, продовження та вказівник на поточне слово, що дозволяє підсвічувати слова під час читання.

– Переваги: працює без інтернету, миттєвий старт, нативна інтеграція.

– Обмеження: якість голосів може бути нижчою, обмежена кількість голосів і параметрів на деяких пристроях.

4.4.3. Онлайн-переклад (DeepL API)

Для перекладу виділених слів або фраз у режимі онлайн застосовується API DeepL — одна з найточніших ML-систем перекладу:

- Запит: HTTP POST-запит із передачею тексту та мовних параметрів (наприклад, source=auto, target=uk) на https://api-free.deepl.com/v2/translate.
- Авторизація: через токен у заголовку Authorization: DeepL-Auth-Key <ключ>.
- Результат: повертається JSON з масивом translations, де зберігається переклад.
- Особливості: переклад точніший, ніж у Google Translate, краще зберігає граматичну структуру та контекст.
- Обмеження: безплатна версія підтримує обмежений список мов тому для розширення підтримуваних мов можуть знадобитися витрати, вимагає мережевий доступ.

4.4.4. Офлайн-переклад (Google ML Kit Translation)

Для підтримки перекладу без Інтернету додаток використовує com.google.mlkit.nl.translate.Translation із бібліотеки Google ML Kit[15], що дозволяє завантажувати мовні моделі та виконувати переклад локально:

- Інтеграція: реалізовано через нативний Android-модуль та викликається з JavaScript-коду через NativeModules.Translator.translateWord(...).
- Підтримка мов: моделі потрібно попередньо завантажити — ML Kit самостійно зберігає їх на пристрої.
- Переваги: працює без мережі, висока швидкість, немає витрат на API.
- Обмеження: підтримується обмежена кількість мов, відсутні елементи контекстного перекладу для складних фраз.

4.5 Інтерфейс додатку

Інтерфейс мобільного додатку ReadEasy було розроблено з урахуванням принципів зручності, доступності та функціональності для широкої аудиторії користувачів. Для реалізації графічного інтерфейсу застосовано фреймворк React Native, а також бібліотеки expo-router, що забезпечила кросплатформенність та підтримку адаптивної навігації.

Основна структура навігації побудована на основі бокового drawer-меню, де розміщено всі ключові екрани додатка . Це дозволяє легко орієнтуватися в інтерфейсі незалежно від кількості доступного контенту.

Екран «Бібліотека» (рис. 10). Це головний екран, який відкривається при запуску додатка . Він містить список доданих користувачем книг з інформацією про назву, тип формату, дату останнього читання та збережену сторінку. Кожну книгу можна відкрити для читання, видалити або позначити як улюблену через swipe-жести. Доступна також кнопка для додавання нових книг через файловий менеджер або хмарне сховище.

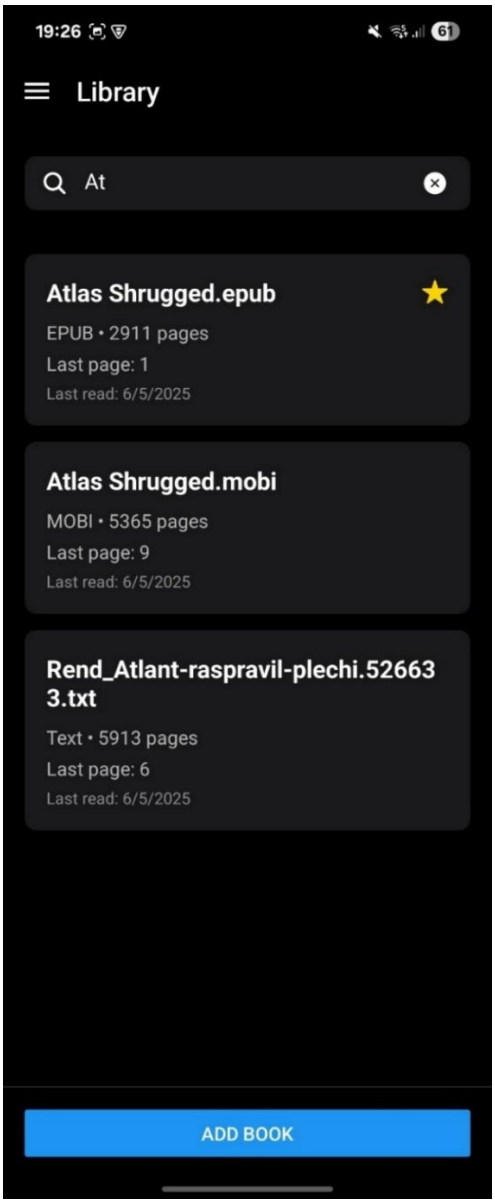

Рисунок 10 – Екран «Бібліотека» (рисунок виконаний самостійно)

Екран «Читання» (рис. 11). Цей екран забезпечує безпосередній перегляд контенту книги. Текст відображається сторінками, з можливістю свайпу вліво/вправо для перемикання. Реалізовано функції виділення тексту, озвучування (TTS) та перекладу виділених фрагментів, переходу на сторінку та пошук по ключовим словам. Є можливість регулювання зовнішнього вигляду тексту — шрифт, міжрядковий інтервал, відступи.

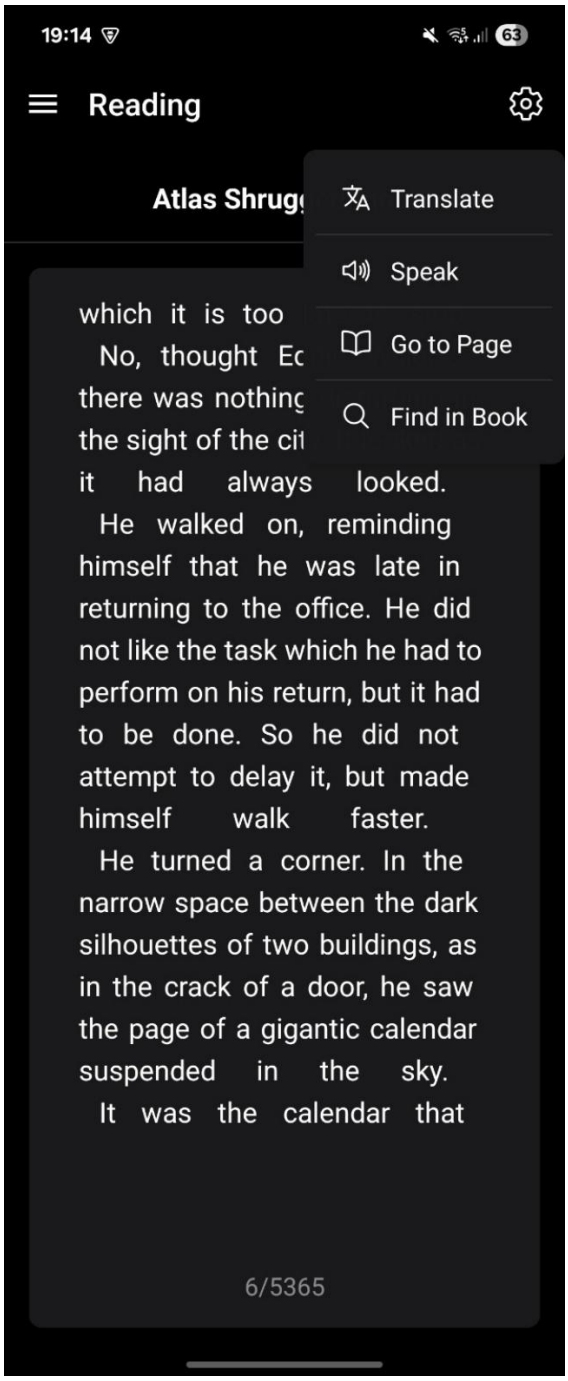

Рисунок 11 – Екран «Читання» (рисунок виконаний самостійно)

При натисканні на кнопку, нижній частині екрану відображається відповідна панель керування з кнопками «Озвучити», «Перекласти», «Знайти» та «Перейти», які відповідають за функціональність. Також при перекладі можна зберегти його до історії перекладів для подальшого використання у вивченні мови (рис. 12).

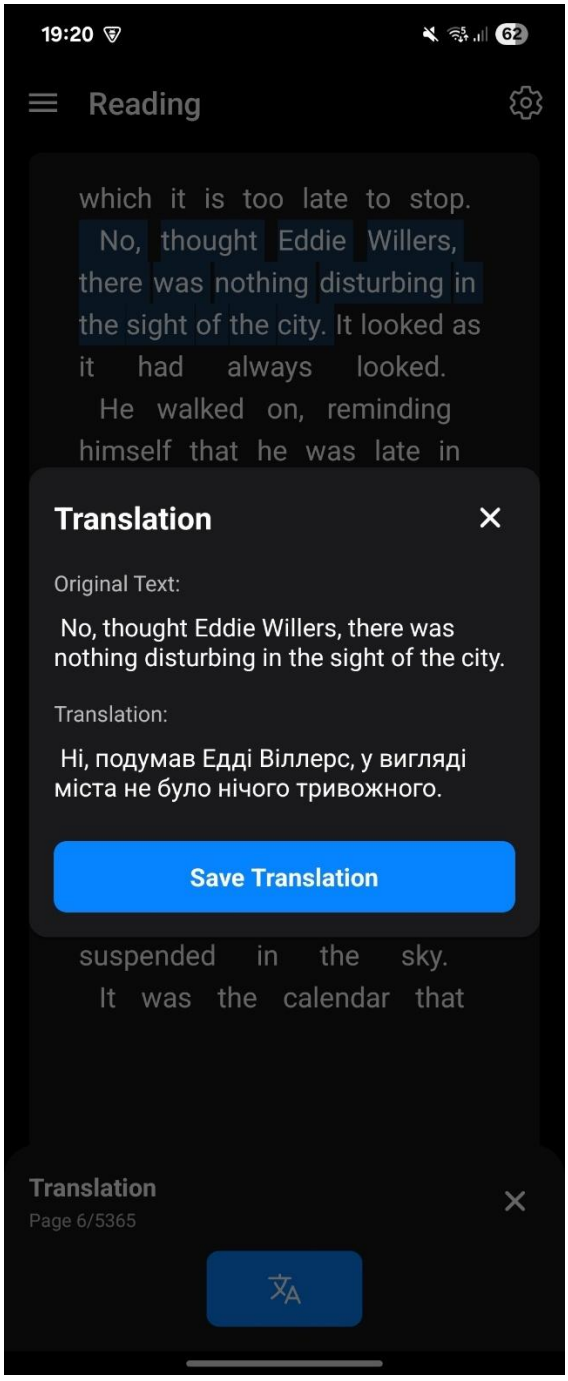

Рисунок 12 – Модальне вікно перекладу (рисунок виконаний самостійно)

Екран «Про книгу» (рис. 13). Містить детальну інформацію про вибрану книгу: назву, кількість сторінок, розмір файлу, дату останнього читання. Доступні

функції поширення (share) та видалення з додатку книги. Прогрес читання відображається у вигляді прогрес-бара.

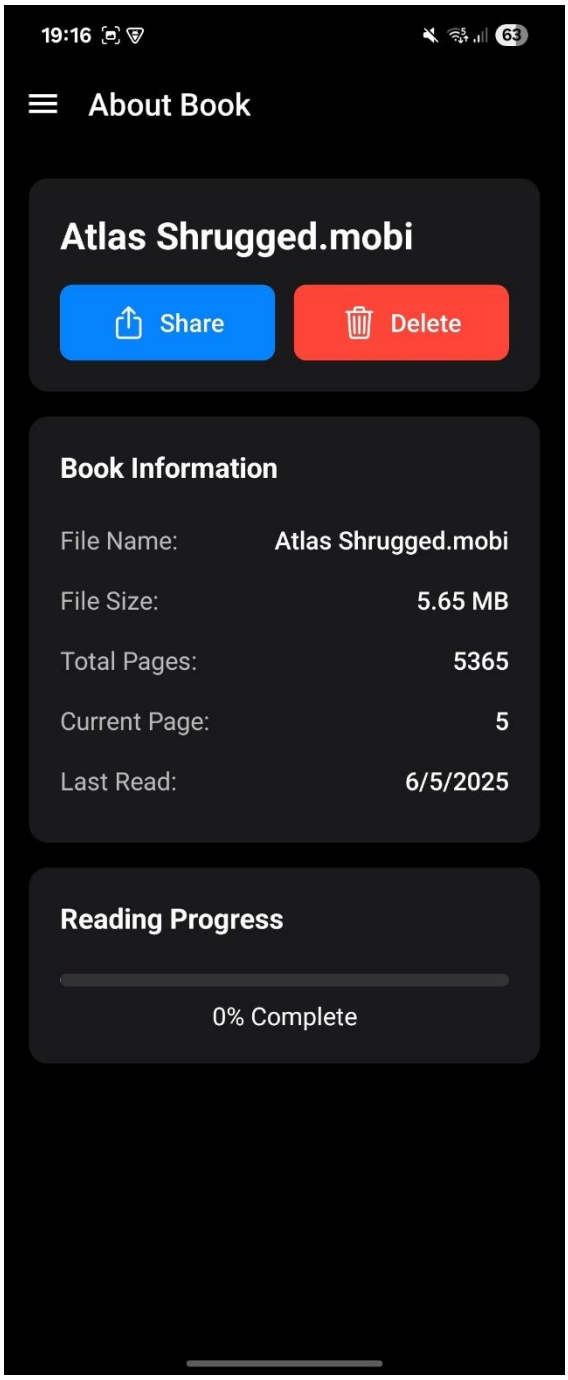

Рисунок 13 – Екран «Про книгу» (рисунок виконаний самостійно)

Екран «Історія перекладів» (рис. 14). Цей екран містить список усіх збережених перекладів, які користувач отримував у процесі читання. Для кожного запису вказано оригінальний текст, переклад, назву книги та дату. Користувач може видаляти окремі записи.

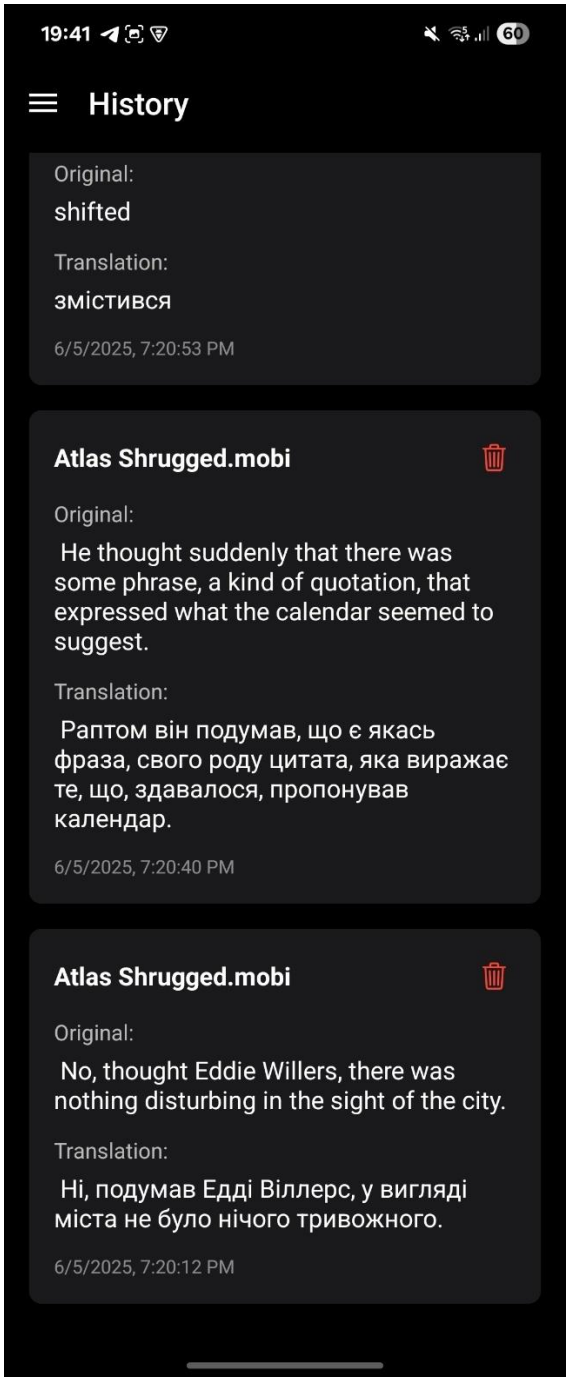

Рисунок 14 - Екран «Історія перекладів» (рисунок виконаний самостійно)

Екран «Статистика» (рис. 15). Містить узагальнену інформацію про активність користувача: кількість прочитаних сторінок, час читання, кількість книг тощо. Дані відображаються у зручному форматі та можуть слугувати мотиваційним інструментом для користувача.

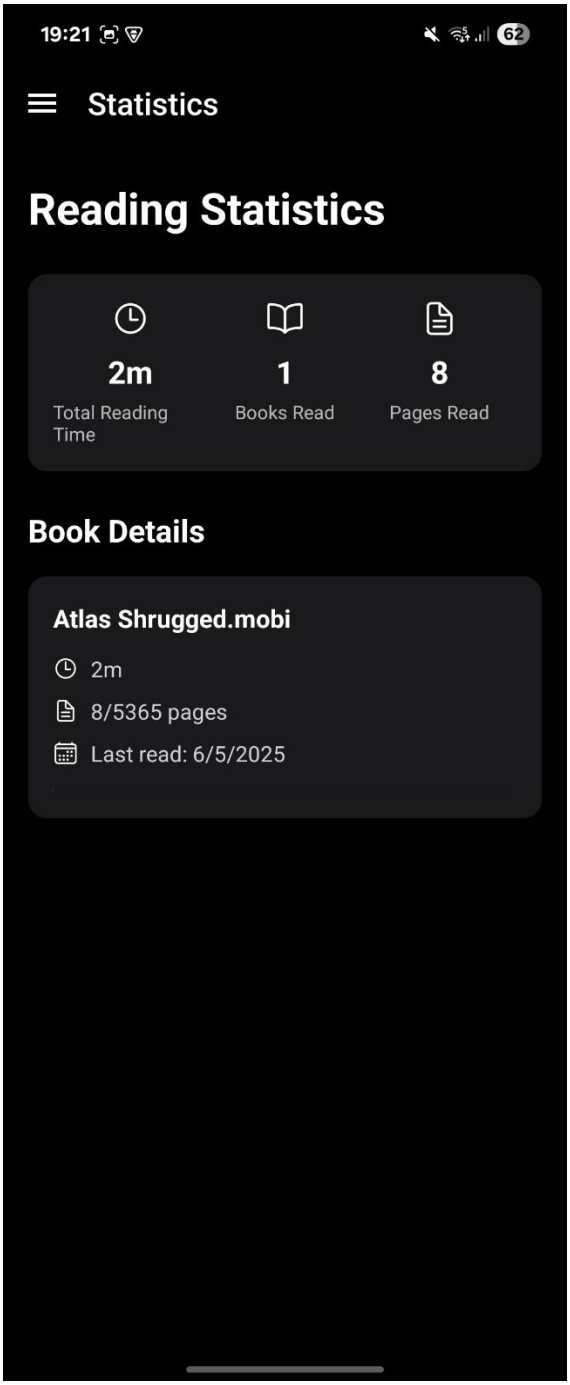

Рисунок 15 - Екран «Статистика» (рисунок виконаний самостійно)

Екран «Налаштування» (рис. 16). Користувач може налаштовувати вигляд інтерфейсу (темна/світла тема, шрифт, інтервал), а також параметри TTS і перекладу (мова, швидкість, голос). Зміни зберігаються локально та застосовуються негайно.

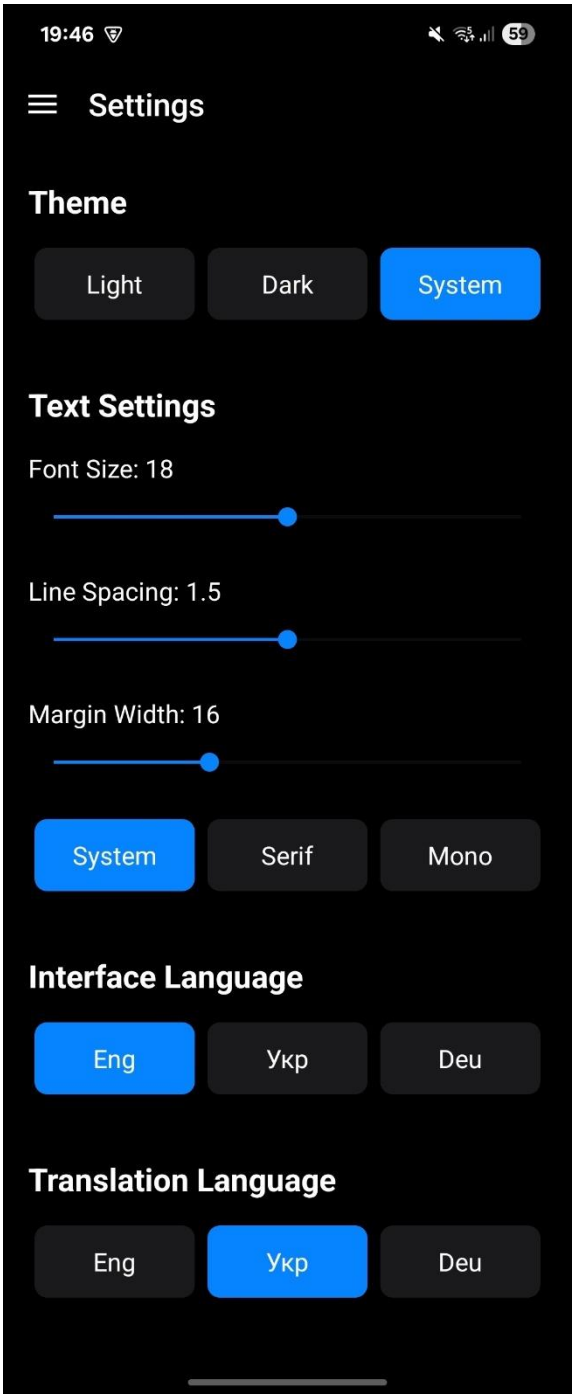

Рисунок 16 - Екран «Налаштування» (рисунок виконаний самостійно)

Екран «Про додаток» (рис. 17). Надає інформацію про версію додатку, розробника, а також опис основних функцій.

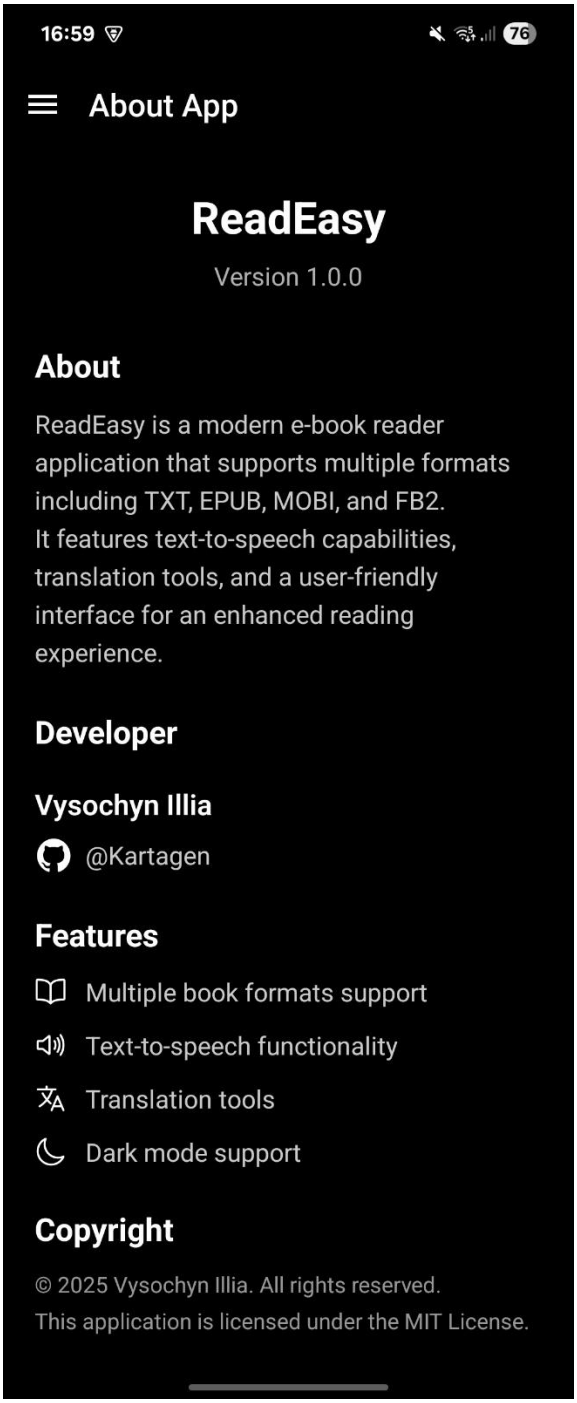

Рисунок 17 - Екран «Про додаток» (рисунок виконаний самостійно)

Цей інтерфейс відповідає сучасним стандартам UX-дизайну, зокрема WCAG 2.1, і забезпечує комфортне використання навіть для людей з вадами зору або користувачів, які вивчають іноземні мови.

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування програмного забезпечення є ключовим етапом життєвого циклу розробки мобільного додатка , який гарантує надійність, стабільність і якість взаємодії користувача з функціоналом. У рамках розробки мобільного додатка було проведено комплексне тестування, що охоплює ручне (мануальне) тестування інтерфейсу й функціональних сценаріїв, а також автоматизовані модульні тести для перевірки бізнес-логіки та цілісності збереження станів.

Метою тестування було виявлення потенційних помилок на ранніх етапах, перевірка коректності виконання ключових функцій у типових та граничних ситуаціях, а також забезпечення відповідності реальної роботи додатка очікуванням користувачів. Особливу увагу було приділено стабільності додатка в умовах обмеженого підключення до Інтернету, коректній роботі функцій озвучення, перекладу та збереження даних.

5.1 Мануальне тестування

Мануальне тестування проводилось на фізичних Android-пристроях із різними версіями операційної системи (від Android 10 до Android 16), що дало змогу оцінити стабільність поведінки додатка в реальних умовах. Було розроблено тест-план, що охоплював базові та розширені сценарії використання додатку, відповідно до передбаченої функціональності в технічному завданні та специфікації вимог. Перевірялися такі сценарії:

- інтерфейс користувача (UI): перевірка відповідності елементів інтерфейсу очікуваному дизайну, зручності навігації, реакції на дії користувача, коректності роботи у світлій та темній темах;
- навігація між екранами: тестування переходів через Drawer-меню, перевірка коректної роботи кнопок «Назад», зміна стану при поверненні до попередніх екранів;

– функціональність бібліотеки: імпорт книг різних форматів (TXT, EPUB, FB2, MOBI), видалення книг, позначення як улюблених, відображення детальної інформації;

– екран читання: відображення вмісту тексту сторінками з належним форматуванням; тестування жестів свайпу для перегортання сторінок вліво/вправо; коректне масштабування та зміна шрифту, міжрядкового інтервалу, відступів при зміні налаштувань;

– озвучення тексту: відтворення з використанням онлайн та офлайн режиму через, коректне озвучення виділених фрагментів, зупинка/відновлення, відображення поточного слова;

– переклад: виділення слів, відображення перекладу онлайн та офлайн; правильна обробка відсутності мережі; можливість збереження перекладів;

– збереження та відновлення прогресу: тестування автозбереження сторінки після виходу з книги, повторне відкриття книги з тієї ж сторінки;

– історія перекладів: перевірка додавання, перегляду та видалення перекладених фраз; перевірка збереження назв книг і дати перекладу;

– статистика: облік часу читання, підрахунок прочитаних сторінок, відображення даних для окремих книжок у форматі списку.

Особливі випадки тестування:

– тестування на стабільність: багаторазове відкриття/закриття книги, швидке перемикання між екранами, робота з великими текстами (>5000 мобільних сторінок);

– тестування виняткових ситуацій: перевірка поведінки при видаленні книги під час читання, при розриві інтернет-з’єднання під час перекладу або озвучення, при імпорті порожніх або пошкоджених файлів.

Усі основні функції додатку пройшли ручне тестування успішно. Збоїв у роботі не було зафіксовано. Виявлені незначні UI-неточності були оперативно усунені до завершення етапу розробки. Результати мануального тестування повністю підтвердили відповідність реалізованого функціоналу функціональним та нефункціональним вимогам, що описані у SRS-документі, зокрема в частині

стабільності роботи, швидкості відгуку інтерфейсу, підтримки офлайн-функцій озвучення та перекладу, а також інтерфейсної адаптації під темну й світлу теми. Всі передбачені обмеження, такі як залежність деяких функцій від наявності інтернет-з’єднання (Google TTS API, DeepL), а також необхідність попереднього завантаження моделей ML Kit для офлайн-перекладу, були враховані під час тестування й не призвели до критичних збоїв. Таким чином, тестування підтвердило, що застосунок стабільно працює в межах заявлених технічних характеристик та відповідає очікуванням кінцевих користувачів.

5.2 Автоматизоване тестування

Було написано та виконано модульні тести для основних логічних модулів програми на базі бібліотеки Jest [16]. Метою цих тестів було перевірити коректність роботи внутрішніх станів у Zustand-сховищах (stores), які відповідають за збереження даних додатку: книг, перекладів, статистики тощо. Це критично важливо для забезпечення стабільності додатка , особливо в умовах асинхронних подій, переходів між екранами та динамічних змін контенту користувачем. Зокрема, були протестовані такі аспекти роботи станів:

Сховище перекладів (useTranslationStore):

- додавання перекладу;
- видалення перекладу.

Сховище статистики (useStatisticsStore):

- запуск та завершення сесії читання;
- оновлення кількості прочитаних сторінок.

Сховище книг (useBookStore):

- додавання книги;
- видалення книги;
- позначення книги як улюбленої.

В результаті тести було пройдено (рис. 18). Це свідчить про повну працездатність базових механізмів роботи із внутрішнім станом додатка . Всі тести проходять без помилок.

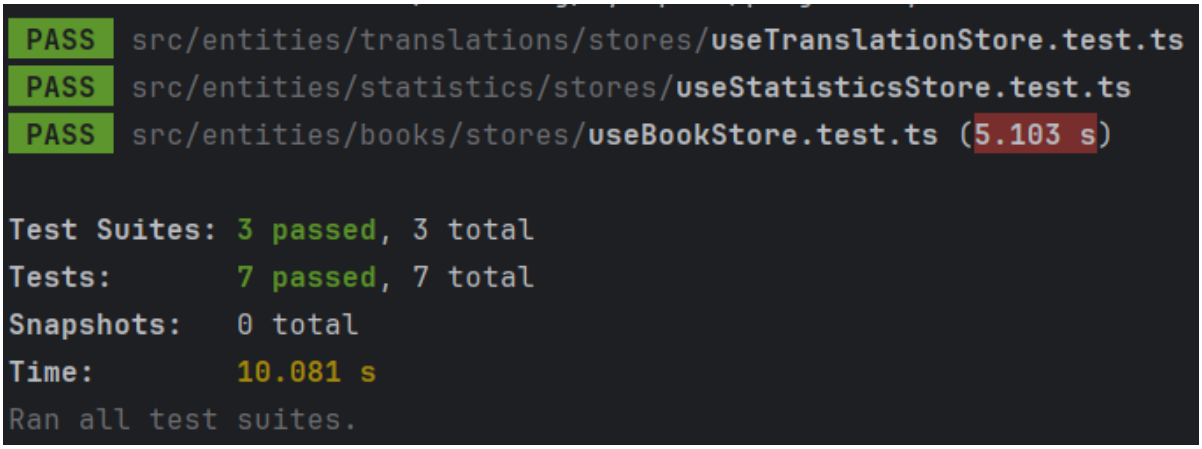

Рисунок 18 – Тестування роботи сховищ станів (рисунок виконаний самостійно)

Таким чином, комбінація мануального та автоматизованого тестування дозволила виявити та усунути потенційні помилки ще до етапу релізу, забезпечивши стабільність, надійність і високу якість користувацького досвіду.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було створено мобільний додаток ReadEasy — сучасний інструмент для читання електронних книг із функціоналом перекладу та автоматичного озвучення тексту. Проєкт охопив повний цикл створення програмного продукту: від постановки задачі й формування вимог до системи до реалізації функціоналу, тестування й підготовки до впровадження. Основною метою розробки було забезпечення доступності книжкового контенту для широкої аудиторії, зокрема людей із порушеннями зору та користувачів, які вивчають іноземні мови.

Програмна система є додатком із підтримкою багатьох форматів, а також інтеграцію з хмарними сервісами для імпорту книг. Особливу увагу приділено зручності користувача: реалізовано адаптивний інтерфейс, підтримку тем оформлення, персоналізацію елементів UI та механізми збереження прогресу читання. Завдяки використанню React Native і TypeScript додаток працює стабільно на більшості сучасних Android та IOS пристроїв.

Функціональні можливості ReadEasy включають миттєвий переклад виділеного тексту, озвучення окремих фрагментів і повних книг, підтримку багатьох мов і збереження перекладів до словника користувача. Інтеграція з DeepL API забезпечує точний та оперативний переклад, а система статистики дозволяє відстежувати динаміку користування.

У розробці системи було враховано потреби різних цільових груп, включно зі студентами, викладачами, професіоналами, поліглотами та людьми з порушенням зору. Для останніх реалізовано контрастні теми та збільшений шрифт. Також у додатка передбачено офлайн-режим, що дозволяє працювати з уже завантаженими книгами без доступу до Інтернету, що критично важливо в мобільному користуванні.

Було реалізовано адаптивний інтерфейс із Drawer-навігацією, гнучкими стилями для світлої та темної теми, а також підтримкою персоналізації читання — зміна шрифту, інтервалів, кольорових тем і налаштувань мовлення. Важливим є те,

що інтерфейс пройшов перевірку на відповідність вимогам доступності (WCAG), що підвищує його інклюзивність.

З технічного погляду, під час реалізації було застосовано сучасні принципи проектування: модульну архітектуру, використання Zustand для управління станом, підтримку асинхронної обробки запитів до зовнішніх API та збереження даних у локальному сховищі. Проведене тестування підтвердило відповідність продуктивності заданим вимогам — запуск додатка , відкриття книг, відтворення озвучення та переклад виконуються із мінімальними затримками.

Систему тестувалось як вручну, так і автоматизовано. Було створено набір модульних тестів для основних функціональних блоків: управління книгами, перекладами та статистикою. Результати тестування засвідчили стабільну роботу всіх критичних механізмів. Таким чином, розробка охопила повний цикл життєвого циклу ПЗ — від постановки задачі до перевірки якості.

Загалом, ReadEasy демонструє приклад збалансованого, адаптивного та інклюзивного мобільного програмного продукту, який вирішує актуальні задачі цифрової грамотності, полегшує доступ до літератури та сприяє вивченню іноземних мов. Розроблений додаток готовий до розгортання на ринку, має потенціал до розширення функціональності та є конкурентоздатним серед сучасних додатків для читання електронних книг та інструментів для вивчення літератури.

В ході навчання було проведено апробацію результатів дослідження у ряді конференцій [1, 7], зокрема:

- Держава, регіони, підприємництво: інформаційні, суспільно-правові, соціально-економічні аспекти розвитку: збірник матеріалів VI Міжнародної конференції (5 грудня 2024 р., м. Київ): Університет "КРОК", 2024;
- 29-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті»(16–19 квітня 2025 р.): Харківський національний університет радіоелектроніки (ХНУРЕ).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Vysochyn Illya, Michkivskyy Sergiy USING MACHINE LEARNING FOR TRANSLATION AND SPEECH GENERATION IN E-BOOK READING APPLICATIONS // Держава, регіони, підприємництво: інформаційні, суспільно-правові, соціально-економічні аспекти розвитку: збірник матеріалів VI Міжнародної конференції (5 грудня 2024 р., м. Київ): Університет "КРОК", 2024 – URL: https://conf.krok.edu.ua/SRE/SRE-2024/paper/view/2661

2. Google Play Книги: офіційна сторінка у Google Play Market [Електронний ресурс]. – URL: https://play.google.com/store/apps/details?id=com.google.android.apps.books. (дата звернення: 04.04.2021).

3. Audible – the leading producer and provider of audio storytelling [Електронний ресурс]. – URL: https://www.audible.com/about/. (дата звернення: 04.04.2025).

4. Speechify – Text to Speech Online [Електронний ресурс]. – URL: https://speechify.com/text-to-speech-online/. (дата звернення 04.04.2025).

5. Adam Boduch. React and React Native / Packt Publishing Ltd, 2017 – ISBN 9781786469571. 210 с.

6. Zustand: документація – Introduction / PMND [Електронний ресурс]. – URL: https://zustand.docs.pmnd.rs/getting-started/introduction. (дата звернення 12.04.2025).

7. Височин І. М. Модульна архітектура у React Native-додатках / І. М. Височин, Н. С. Кравець // Радіоелектроніка та молодь у XXI столітті : матеріали 29-го Міжнар. молодіж. форуму, 16–19 квітня 2025 р. – Харків : ХНУРЕ, 2025. – Т. 6 – С. 241-243.

8. Feature-Sliced Design: офіційна документація [Електронний ресурс]. – URL: https://feature-sliced.github.io/documentation/docs. (дата звернення 10.04.2025).

9. Richard Kho. React Native By Example / Packt Publishing Ltd, 2017 – ISBN 9781786465641. 219 с.

10. React Native Documentation [Електронний ресурс]. – URL: https://reactnative.dev/docs/getting-started. (дата звернення: 10.04.2025).

11. React Native Gesture Handler Documentation [Електронний ресурс]. – URL: https://docs.swmansion.com/react-native-gesture-handler/docs/. (дата звернення: 21.04.2025).

12. World Wide Web Consortium (W3C). Керівництво з доступності вебконтенту (WCAG) 2.1 [Електронний ресурс]. – URL: https://www.w3.org/Translations/WCAG21-ua/. (дата звернення: 17.04.2025).

13. Google Cloud Text-to-Speech API Documentation [Електронний ресурс]. – URL: https://cloud.google.com/text-to-speech/docs/reference/rest (дата звернення: 28.04.2025).

14. DeepL API Documentation [Електронний ресурс]. – URL: https://developers.deepl.com/docs/api-reference/translate (дата звернення: 29.04.2025).

15. Google ML Kit Translation Documentation [Електронний ресурс]. – URL: https://developers.google.com/ml-kit/language/translation/android (дата звернення: 26.04.2025).

16. Jest – Testing React Native Apps [Електронний ресурс]. – URL: https://jestjs.io/docs/tutorial-react-native (дата звернення: 15.05.2025).

17. Посилання на GitHub – URL: https://github.com/NureVysochynIllia/2025_B_PI_PZPI-21-7_Vysochyn_I_M/.