**ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ**
**Навчально-науковий інститут економіки, управління, права та інформаційних технологій**
**Кафедра інформаційних систем та технологій**

# КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавр

на тему: **«Розроблення мобільного додатку з функцією озвучування текстів»**

Виконав: здобувач вищої освіти
за освітньою програмою
Інформаційні управляючі системи
спеціальності 126 Інформаційні
системи та технології
ступеня вищої освіти бакалавр
групи 126ІСТ_бд_2022
Юхименко Євгеній Вячеславович
Керівник: Поночовний Юрій
Леонідович
Рецензент: Муравльов Володимир
Вячеславович

**Полтава – 2026 року**

**ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ**
**Навчально-науковий інститут економіки, управління, права та інформаційних технологій**
**Кафедра інформаційних систем та технологій**

Освітня програма Інформаційні управляючі системи
Спеціальність 126 Інформаційні системи та технології
Рівень вищої освіти перший (бакалаврський)

**ЗАТВЕРДЖУЮ**
Завідувач кафедри
_____ Юрій УТКІН
«10» червня 2025 року

**З А В Д А Н Н Я**
**НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ**

**Юхименку Євгенію Вячеславовичу**

1. Тема роботи:
«Розроблення мобільного додатку з функцією озвучування текстів»,
керівник роботи д.т.н., професор, професор кафедри інформаційних систем та технологій Поночовний Юрій Леонідович.
Затверджено наказом закладу вищої освіти від «10» квітня 2026 року № 383 ст.

2. Строк подання здобувачем вищої освіти роботи «08» червня 2026 року.

3. Вихідні дані до роботи: стандарти проєктування та розробки мобільних застосунків, документація офіційних сайтів https://flutter.dev/, https://dart.dev/, https://developer.android.com/ (для Android TTS API), https://pub.dev/ (для пакетів Flutter), середовища прикладних програм Visual Studio Code з розширенням Flutter, Android SDK Command Line Tools.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):

Розділ 1. Аналіз предметної області, існуючих рішень та технологій розробки мобільних додатків для читання і озвучування текстів
Розділ 2. Проєктування архітектури, бази даних та користувацького інтерфейсу мобільного додатку
Розділ 3. Реалізація, тестування та оцінювання ефективності мобільного додатку з функцією озвучування текстів

5. Перелік графічного матеріалу: схеми, рисунки, графіки, діаграми за темою та об’єктом дослідження.

6. Консультанти розділів кваліфікаційної роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання отримав |
| Оцінювання економічної ефективності результатів дослідження | Дивнич О. Д., к. е. н., доцент, доцент кафедри економіки та публічного управління | 01.04.2026 р. | 29.05.2026 р. |

7. Дата видачі завдання «10» червня 2025 року

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів роботи | Строк виконання етапів кваліфікаційної роботи | Примітка |
|---|---|---|---|
| 1 | Вибір і затвердження теми роботи | 02.06.2025 р. – 04.06.2025 | |
| 2 | Складання і затвердження розгорнутого плану та завдання на кваліфікаційну роботу | 05.06.2025 р. – 10.06.2025 р. | |
| 3 | Опрацювання джерел інформації | 11.06.2025 р. – 15.06.2025 р. | |
| 4 | Збір, вивчення і обробка інформації, необхідної для виконання роботи | 16.06.2025 р. – 20.06.2025 р. | |
| 5 | Виконання теоретико-методологічного розділу роботи | 08.09.2025 р. – 01.11.2025 р. | |
| 6 | Виконання дослідницько-аналітичного розділу роботи | 19.01.2026 р. – 14.02.2026 р. | |
| 7 | Виконання проєктно-рекомендаційного розділу роботи | 16.02.2026 р. – 31.03.2026 р. | |
| 8 | Оцінювання економічної ефективності результатів дослідження | 01.04.2026 р. – 29.05.2026 р. | |
| 9 | Оформлення тексту роботи | 01.06.2026 р. – 06.06.2026р. | |
| 10 | Попередній захист роботи на кафедрі | 08.06.2026 р. | |
| 11 | Доопрацювання роботи з урахуванням зауважень і пропозицій | 09.06.2026 р. – 10.06.2026 р. | |
| 12 | Нормоконтроль | 11.06.2026 р. – 15.06.2026 р. | |
| 13 | Захист кваліфікаційної роботи | 16.06.2026 р. - 18.06.2026 р. | |

**Здобувач вищої освіти** **Євгеній ЮХИМЕНКО**

**Керівник роботи** **Юрій ПОНОЧОВНИЙ**

**ЗМІСТ**

ВСТУП ..... 6
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ, ІСНУЮЧИХ РІШЕНЬ ТА ТЕХНОЛОГІЙ РОЗРОБКИ МОБІЛЬНИХ ДОДАТКІВ ДЛЯ ЧИТАННЯ І ОЗВУЧУВАННЯ ТЕКСТІВ ..... 9
1.1 Огляд предметної області та актуальність розробки мобільних додатків для читання текстів ..... 9
1.2 Аналіз існуючих аналогів та їх функціональних можливостей ..... 12
1.3 Вибір технологій та інструментів розробки ..... 15
1.4 Визначення вимог до розроблюваного мобільного додатку ..... 18
РОЗДІЛ 2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ, БАЗИ ДАНИХ ТА КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ МОБІЛЬНОГО ДОДАТКУ ..... 22
2.1 Функціональні та нефункціональні вимоги до системи ..... 22
2.2 Архітектура мобільного додатку та проєктування бази даних ..... 25
2.3 Проєктування користувацького інтерфейсу ..... 28
2.4 Вибір та обґрунтування методу озвучування текстів ..... 32
РОЗДІЛ 3 РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ МОБІЛЬНОГО ДОДАТКУ З ФУНКЦІЄЮ ОЗВУЧУВАННЯ ТЕКСТІВ ..... 36
3.1 Реалізація основних компонентів додатку ..... 36
3.2 Реалізація функції озвучування та налаштувань відтворення ..... 39
3.3 Тестування розробленого додатку ..... 43
3.4 Техніко-економічне обґрунтування ефективності розробленої системи ..... 46
ВИСНОВКИ ..... 50
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ..... 52
ДОДАТКИ ..... 56

# ВСТУП

*Актуальність.* У сучасному інформаційному суспільстві доступність текстового контенту в зручному форматі є важливою складовою цифрової інклюзії. Читання та прослуховування текстів на мобільних пристроях набуває дедалі більшого поширення: за даними аналітичних досліджень [1, 2], ринок аудіокниг і застосунків для озвучування тексту щороку зростає, а кількість користувачів мобільних пристроїв, які споживають текстовий контент у форматі TTS, збільшується пропорційно до зростання частки смартфонів у повсякденному житті. Функція синтезу мовлення є особливо затребуваною для людей із порушеннями зору, водіїв, студентів і всіх, хто не має можливості читати з екрана в певних ситуаціях [4].

В українському контексті питання доступності цифрового контенту набуло особливої гостроти після 2022 року. Зросла потреба у застосунках, орієнтованих на україномовних користувачів, що працюють автономно, без залежності від іноземних хмарних сервісів, які раніше домінували на вітчизняному ринку [3, 11]. Відкриті україномовні текстові бібліотеки містять значні масиви літератури у форматі TXT, однак існуючі мобільні застосунки для читання або не підтримують цей формат повною мірою, або реалізують функцію озвучування виключно в межах власних закритих контентних екосистем, не дозволяючи користувачу працювати з власними файлами [8, 9]. Це формує незайняту нішу для розробки спеціалізованого застосунку, орієнтованого саме на роботу з довільними TXT-файлами з повноцінною підтримкою українського TTS.

*Метою кваліфікаційної роботи* є розроблення мобільного застосунку для платформи Android з функцією озвучування текстових файлів формату TXT засобами вбудованого рушія синтезу мовлення з підтримкою української мови, що забезпечує автономну роботу без підключення до мережі інтернет.

Відповідно до мети роботи необхідно вирішити наступні *завдання:*

1. Провести аналіз предметної області, існуючих аналогів мобільних застосунків для читання та озвучування текстів, а також обрати технологічний стек для реалізації;

2. Спроєктувати архітектуру мобільного застосунку, базу даних для зберігання метаданих текстів, прогресу читання, цитат і налаштувань, а також розробити макети користувацького інтерфейсу;

3. Реалізувати мобільний застосунок із функцією озвучування тексту, провести його функціональне тестування на реальному пристрої та виконати техніко-економічне обґрунтування ефективності розробленої системи.

*Об'єкт дослідження* – процеси проєктування та розробки мобільних застосунків для читання і озвучування текстового контенту на платформі Android.

*Предмет дослідження* – проєктування та розробка мобільного застосунку «Audible TXT» з функцією озвучування текстів формату TXT з використанням фреймворку Flutter, локальної бази даних SQLite та вбудованого Android TTS API.

*Методи досліджень* – дослідження базуються на методах програмної інженерії, аналізу та пріоритизації вимог (метод MoSCoW), проєктування реляційних баз даних, розробки мобільних застосунків із використанням архітектурного патерну MVVM, функціонального тестування на реальному пристрої, а також техніко-економічного аналізу ефективності програмних продуктів.

*Інформаційна база* – інтернет-ресурси, наукова та технічна документація з мобільної розробки на Flutter і Dart, документація Android Developers щодо TTS API, пакети відкритого коду з репозиторію pub.dev, стандарти якості програмного забезпечення ISO/IEC 25010, рекомендації з доступності WCAG 2.1, а також статистичні дані про ринок мобільних застосунків і цифрового контенту.

*Практична значущість* – розроблений застосунок «Audible TXT» забезпечує україномовним користувачам безкоштовний, автономний і конфіденційний інструмент для читання та прослуховування власних TXT-файлів на Android-пристроях. Запропонована архітектура на базі Flutter та MVVM, реалізована схема локальної бази даних і обраний підхід до розбиття тексту на сторінки можуть бути використані як основа для подальшого розвитку

застосунку, зокрема додавання підтримки інших форматів файлів, хмарної синхронізації та розширеного керування бібліотекою.

Результати роботи апробовані в рамках наукової конференції здобувачів вищої освіти за результатами науково-дослідної роботи у 2025-2026 рр.

Структура кваліфікаційної роботи логічно пов’язана з задачами досліджень і містить перелік умовних позначень, вступ, три розділи основної частини, висновки, список використаних джерел. Загальний обсяг текстової частини кваліфікаційної роботи складає 55 сторінок формату А4. Вона містить 18 рисунків і 13 таблиць.

# РОЗДІЛ 1
# АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ, ІСНУЮЧИХ РІШЕНЬ ТА ТЕХНОЛОГІЙ РОЗРОБКИ МОБІЛЬНИХ ДОДАТКІВ ДЛЯ ЧИТАННЯ І ОЗВУЧУВАННЯ ТЕКСТІВ

## 1.1 Огляд предметної області та актуальність розробки мобільних додатків для читання текстів

Мобільні додатки стали невід'ємною частиною повсякденного життя сучасної людини. Стрімке поширення смартфонів зумовило зростання попиту на програмне забезпечення для мобільних платформ у найрізноманітніших сферах – від розваг до освіти та професійної діяльності. За даними аналітичних досліджень, кількість завантажень мобільних додатків у світі щороку зростає, а загальний обсяг ринку мобільних застосунків у 2024 році перевищив 500 мільярдів доларів США [1]. Це свідчить про те, що розробка мобільних додатків залишається одним із найперспективніших напрямів у сучасній індустрії програмного забезпечення.

Паралельно з розвитком мобільних технологій активно розвивається ринок цифрових книг і текстового контенту. Електронні книги набули значного поширення завдяки зручності зберігання великої кількості текстів на одному пристрої, доступності й можливості читати в будь-якому місці. Разом із тим з'явився окремий сегмент – аудіокниги та застосунки для озвучування текстів, що дозволяють споживати текстовий контент у ситуаціях, коли читання з екрана неможливе або незручне: під час руху, виконання фізичної роботи, при втомі очей тощо [2].

Особливої актуальності розробка таких застосунків набуває в українському контексті. Після 2022 року значно зросла потреба в україномовному цифровому контенті, зокрема в інструментах, які дозволяють читати та слухати тексти українською мовою без залежності від іноземних,

зокрема російських, сервісів. Відкриті україномовні бібліотеки, такі як «Читанка» та «Укрліб», містять значні масиви текстів у форматі TXT, що робить цей формат природним вибором для розробки вітчизняного застосунку для читання [3].

Сучасні мобільні операційні системи – передусім Android – мають вбудовані рушії синтезу мовлення (Text-to-Speech, TTS), що підтримують українську мову. Це дозволяє реалізувати функцію озвучування текстів без використання сторонніх платних API, забезпечуючи при цьому автономну роботу застосунку без доступу до інтернету та захист персональних даних користувача, оскільки текст не передається на зовнішні сервери [4].

Для розробки кросплатформних мобільних додатків сьогодні широко використовується фреймворк Flutter, створений компанією Google. Flutter дозволяє писати код одноразово і компілювати його для Android та iOS, що суттєво скорочує час розробки. Продуктивність Flutter-застосунків наближається до нативних завдяки власному рушію відображення Skia/Impeller, а функція гарячого перезавантаження (Hot Reload) значно прискорює процес розробки та налагодження [5]. На рисунку 1.1 наведено порівняння популярності основних фреймворків для кросплатформної мобільної розробки.

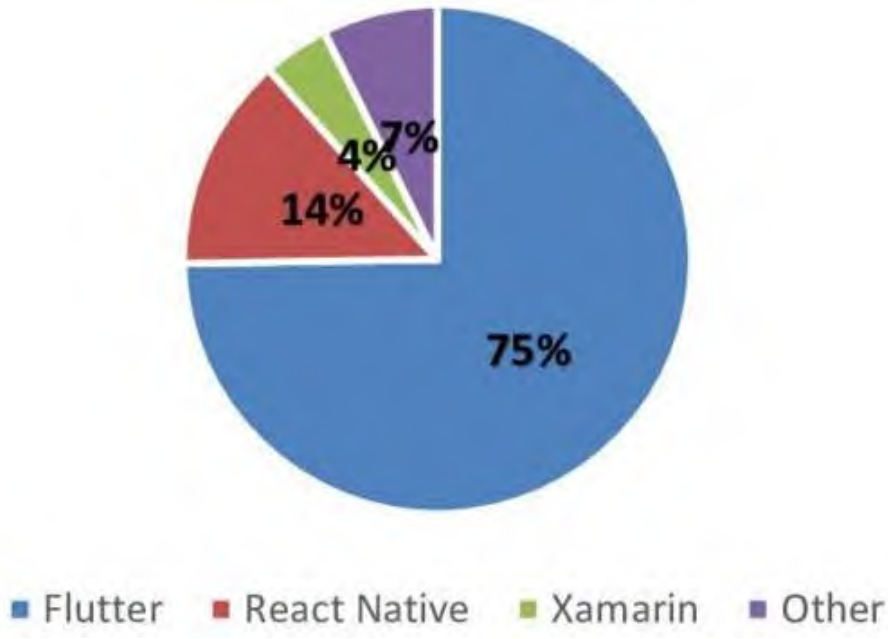


Рисунок 1.1 – Популярність фреймворків кросплатформної мобільної розробки

Мобільний застосунок для читання та озвучування текстів є типовим представником класу застосунків із локальною базою даних і медіавідтворенням. Його архітектура передбачає кілька взаємопов'язаних підсистем: підсистему управління файлами (завантаження, зберігання, видалення текстів), підсистему відображення тексту з підтримкою налаштувань оформлення, підсистему збереження стану (прогрес читання, закладки, цитати) та підсистему озвучування з можливістю регулювання швидкості й голосу.

Таблиця 1.1 – Порівняння підходів до реалізації функції озвучування в мобільних застосунках

| Підхід | Потреба в інтернеті | Вартість | Підтримка української мови | Конфіденційність |
|---|---|---|---|---|
| Зовнішній API (хмарний TTS) | Так | Платний або обмежений | Залежить від провайдера | Низька (текст передається) |
| Вбудований TTS Android | Ні | Безкоштовний | Так (Google TTS) | Висока |
| Локальна нейромережева модель | Ні | Безкоштовний | Обмежена | Висока |

Як видно з таблиці 1.1, використання вбудованого TTS-рушія Android є оптимальним варіантом з огляду на сукупність критеріїв: відсутність фінансових витрат, підтримка української мови, автономна робота та висока конфіденційність.

Якість синтезованого мовлення оцінюється за показником Word Error Rate (WER), який визначає частку помилково розпізнаних або відтворених слів. Формула для обчислення WER має вигляд:

$$WER = \frac{S + D + I}{N}, \tag{1.1}$$

де $S$ – кількість замін,

$D$ – кількість видалень,

$I$ – кількість вставок,

$N$ – загальна кількість слів у еталонному тексті [6].

Середній показник WER для сучасних систем TTS на базі нейронних мереж становить менше 5%, що забезпечує достатньо природне сприйняття синтезованого мовлення [7]. Це підтверджує доцільність використання вбудованого рушія Android як основи для функції озвучування в розроблюваному застосунку.

Таким чином, розробка мобільного застосунку для читання та озвучування текстів є актуальною задачею, що відповідає сучасним тенденціям розвитку мобільних технологій, потребам україномовних користувачів і принципам побудови безпечного та незалежного програмного забезпечення.

## 1.2 Аналіз існуючих аналогів та їх функціональних можливостей

Ринок мобільних застосунків для читання та озвучування текстів є достатньо насиченим. Серед найбільш поширених рішень, доступних українським користувачам, виділяються такі застосунки, як «Букмейт», «Літрес», «MyBook», Google Play Books та застосунок «Книги» від Apple. Аналіз цих рішень дозволяє визначити типовий функціональний набір подібних систем, виявити їхні недоліки та сформулювати конкурентні переваги розроблюваного застосунку [8].

Застосунок «Букмейт» є одним із найпопулярніших у категорії читання електронних книг на пострадянському просторі. Він підтримує кілька форматів файлів, має розвинену систему рекомендацій, дозволяє зберігати цитати, відстежувати прогрес читання та слухати аудіокниги. Проте основний масив контенту в «Букмейті» є платним, а функція озвучування доступна виключно для книг із власної бібліотеки сервісу – завантажити власний текстовий файл і озвучити його засобами застосунку неможливо [9].

«Літрес» – ще один великий гравець ринку, що пропонує широку бібліотеку україномовного та російськомовного контенту. Застосунок підтримує

купівлю, завантаження та прослуховування аудіокниг, надає персоналізовані рекомендації. Однак, як і «Букмейт», «Літрес» орієнтований виключно на власну контентну екосистему і не дозволяє користувачу працювати з довільними текстовими файлами [9].

Google Play Books – кросплатформний застосунок від Google, що підтримує завантаження власних файлів у форматах PDF та EPUB. Серед його можливостей – синхронізація між пристроями, нічний режим, налаштування шрифту. Функція озвучування присутня, однак її якість для української мови залишається нестабільною, а інтерфейс не адаптований під потреби користувачів, які працюють виключно з локальними текстовими файлами [10].

На рисунку 1.2 наведено порівняльну схему функціональних можливостей розглянутих аналогів.

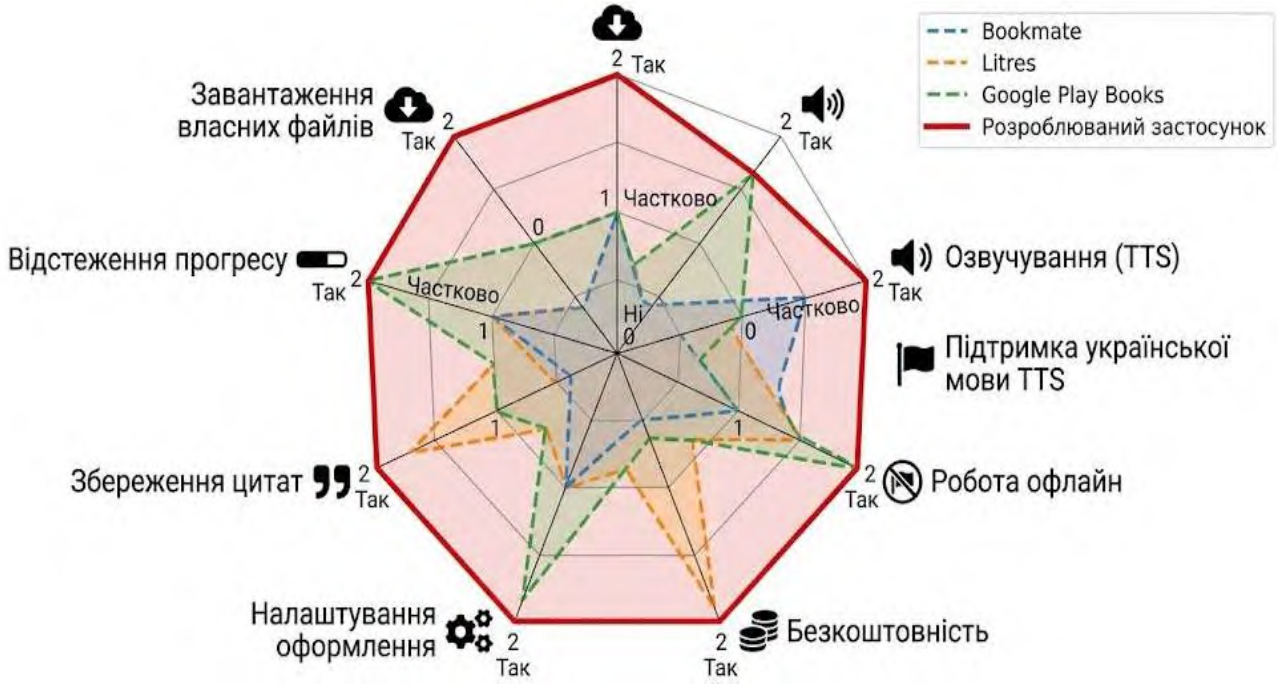


Рисунок 1.2 – Порівняння функціональних можливостей застосунків для читання

Систематизований порівняльний аналіз основних характеристик розглянутих застосунків наведено у таблиці 1.2.

Таблиця 1.2 – Порівняльний аналіз функціональних можливостей застосунків-аналогів

| Критерій | Букмейт | Літрес | Google Play Books | Розроблюваний застосунок |
|---|---|---|---|---|
| Завантаження власних TXT-файлів | Ні | Ні | Частково | Так |
| Озвучування власного тексту | Ні | Ні | Частково | Так |
| Підтримка української мови TTS | – | – | Так (Google TTS) | Так (Google TTS) |
| Робота без інтернету | Частково | Частково | Так | Так |
| Безкоштовне використання | Частково | Частково | Так | Так |
| Збереження цитат | Так | Ні | Ні | Так |
| Налаштування оформлення тексту | Так | Так | Так | Так |
| Відстеження прогресу читання | Так | Так | Так | Так |

З таблиці 1.2 видно, що жоден із розглянутих аналогів не забезпечує повноцінної підтримки завантаження довільних TXT-файлів із наступним їх озвучуванням українською мовою у повністю офлайн-режимі. Саме ця комбінація є ключовою відмінністю розроблюваного застосунку.

Варто також зазначити, що більшість існуючих рішень є частиною закритих контентних екосистем, орієнтованих на монетизацію через продаж книг або підписки. Це суперечить потребам користувачів, які мають власні текстові матеріали – навчальні тексти, конспекти, статті, художні твори у відкритому доступі – і бажають зручно читати або слухати їх на мобільному пристрої [11].

Окремо варто розглянути застосунки-читалки загального призначення, наприклад Moon+ Reader або ReadEra, які підтримують формат TXT і мають широкі можливості налаштування. Проте функція озвучування в них або відсутня, або реалізована як додатковий модуль із обмеженими налаштуваннями [12]. Розроблюваний застосунок ставить озвучування в центр функціональної концепції, а не розглядає його як другорядну опцію.

Таким чином, проведений аналіз підтверджує наявність незайнятої ніші: мобільний застосунок, орієнтований на роботу з власними TXT-файлами користувача, з повноцінною функцією озвучування українською мовою, що працює автономно і є повністю безкоштовним. Визначені недоліки аналогів слугують основою для формулювання функціональних вимог до розроблюваної системи, які будуть розглянуті в наступних підрозділах.

### 1.3 Вибір технологій та інструментів розробки

Вибір технологічного стеку є одним із ключових рішень на етапі підготовки до розробки мобільного застосунку. Від цього вибору залежать продуктивність кінцевого продукту, зручність процесу розробки, доступність інструментів налагодження та довгострокова підтримуваність коду. У контексті розроблюваного застосунку розглядалися три основні підходи: нативна розробка для Android на мові Kotlin, кросплатформна розробка з використанням React Native та кросплатформна розробка з використанням Flutter [13].

Нативна розробка на Kotlin забезпечує найвищу продуктивність і повний доступ до всіх можливостей платформи Android. Kotlin є офіційно рекомендованою мовою для Android-розробки з 2017 року і має статичну типізацію, лаконічний синтаксис та повну сумісність із Java [14]. Проте нативний підхід обмежує застосунок однією платформою і вимагає встановлення Android Studio – середовища розробки з високими системними вимогами, що є суттєвим обмеженням при роботі на комп'ютері з обмеженими ресурсами.

React Native – фреймворк від Meta, що дозволяє писати кросплатформний код на JavaScript або TypeScript. Він має велику спільноту та широку екосистему пакетів, однак його архітектура передбачає міст між JavaScript-потоком і

нативним кодом, що може спричиняти затримки у складних сценаріях відображення [15].

Flutter – фреймворк від Google на мові Dart – відрізняється від React Native тим, що не використовує нативні компоненти інтерфейсу, а малює весь UI самостійно через власний графічний рушій. Це забезпечує однаковий зовнішній вигляд на всіх платформах і високу швидкість відображення. Функція Hot Reload дозволяє бачити зміни в коді майже миттєво, що суттєво прискорює розробку [13]. Крім того, Flutter не потребує встановлення Android Studio – достатньо SDK командного рядка та редактора VS Code, що робить його оптимальним вибором для роботи на комп'ютері з обмеженим дисковим простором. Порівняльний аналіз розглянутих технологій наведено у таблиці 1.3.

Таблиця 1.3 – Порівняльний аналіз технологій мобільної розробки

| Критерій | Kotlin (Native) | React Native | Flutter |
|---|---|---|---|
| Мова програмування | Kotlin | JavaScript / TypeScript | Dart |
| Кросплатформність | Ні | Так | Так |
| Продуктивність UI | Висока | Середня | Висока |
| Вимоги до середовища | Android Studio | Node.js + IDE | VS Code + SDK |
| Hot Reload | Частково | Так | Так |
| Підтримка TTS Android | Повна | Через плагін | Через плагін |
| Мінімальний розмір SDK | ~1 ГБ | ~500 МБ | ~600 МБ |

З таблиці 1.3 видно, що Flutter є найбільш збалансованим варіантом з огляду на сукупність критеріїв, зокрема мінімальні вимоги до середовища розробки та високу продуктивність інтерфейсу.

Для зберігання даних у розроблюваному застосунку обрано локальну реляційну базу даних SQLite через пакет sqflite. Це стандартне рішення для Flutter-застосунків із локальним сховищем, яке забезпечує надійне збереження інформації про завантажені тексти, прогрес читання та цитати без необхідності підключення до мережі [16]. Налаштування оформлення (розмір шрифту, тема)

зберігаються окремо через пакет shared_preferences, що є більш легковісним рішенням для невеликого обсягу конфігураційних даних.

Для реалізації функції озвучування використовується пакет flutter_tts, який є обгорткою над вбудованим Android TTS API. Він надає доступ до параметрів синтезу мовлення: мови, голосу, швидкості та тональності. Підтримка української мови забезпечується рушієм Google Text-to-Speech, встановленим на більшості сучасних Android-пристроїв [17].

Для вибору файлів із пристрою використовується пакет file_picker, для запиту дозволів – permission_handler. Таким чином, повний набір залежностей застосунку є мінімальним і не містить важких нативних бібліотек, що позитивно впливає на час збирання проєкту. На рисунку 1.3 зображено загальну схему технологічного стеку розроблюваного застосунку.

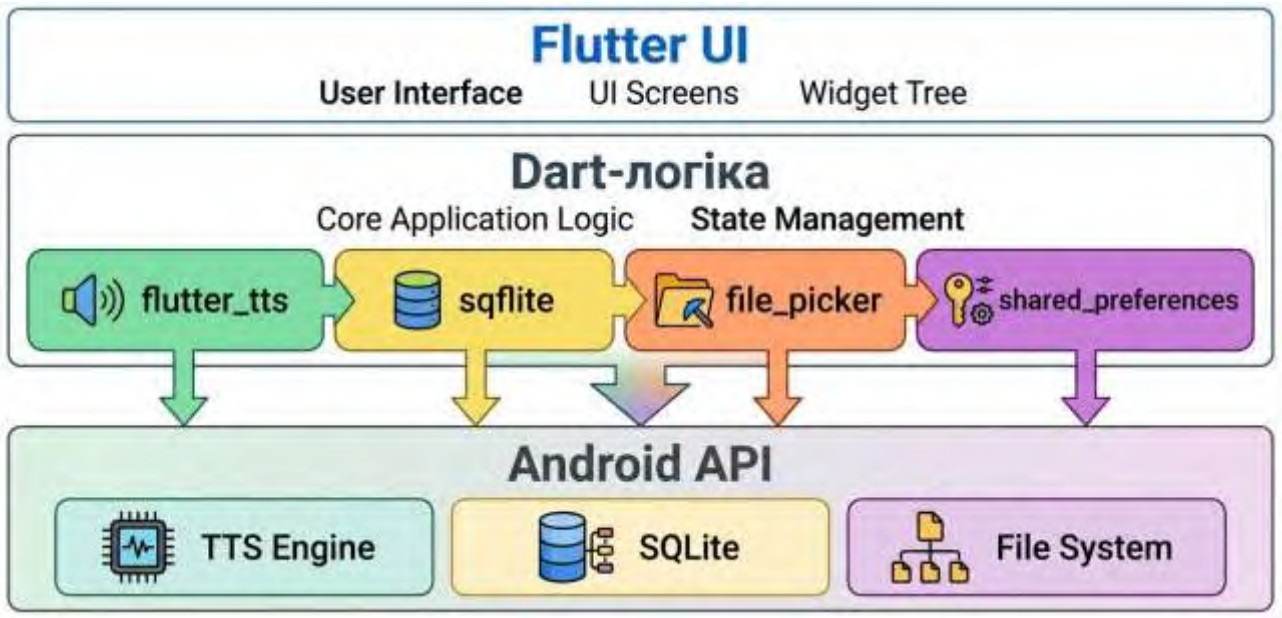


Рисунок 1.3 – Технологічний стек розроблюваного мобільного застосунку

Середовищем розробки обрано Visual Studio Code з розширенням Flutter, що встановлюється одноразово і не потребує повного пакету Android Studio. Android SDK розміщується на окремому диску, а кеш системи збирання Gradle

також виноситься за межі системного диску, що дозволяє ефективно працювати навіть на комп'ютері з обмеженим вільним місцем на системному розділі [13].

Отже, обраний технологічний стек – Flutter, Dart, sqflite, flutter_tts, file_picker – повністю відповідає функціональним вимогам до застосунку, забезпечує зручний процес розробки в умовах обмежених апаратних ресурсів і гарантує підтримку ключової функції озвучування текстів українською мовою без залежності від зовнішніх сервісів.

### 1.4 Визначення вимог до розроблюваного мобільного додатку

Формулювання чітких вимог до програмного забезпечення є обов'язковим етапом розробки будь-якої інформаційної системи. Вимоги поділяються на функціональні, що визначають, що саме має робити система, та нефункціональні, що визначають, як саме система має це робити – з точки зору продуктивності, надійності, зручності використання та інших якісних характеристик [18]. Коректно сформульовані вимоги дозволяють уникнути переробок на пізніх етапах розробки і є основою для подальшого проєктування архітектури та тестування.

Метою розроблюваного застосунку є надання користувачу зручного інструменту для читання текстових файлів формату TXT на мобільному пристрої під управлінням Android з можливістю озвучування тексту вбудованим рушієм синтезу мовлення. Застосунок орієнтований на україномовних користувачів і має працювати повністю автономно, без підключення до мережі інтернет.

На основі аналізу предметної області та недоліків існуючих аналогів, проведеного в попередніх підрозділах, сформульовано перелік функціональних вимог до системи. Застосунок має забезпечувати такі можливості: завантаження TXT-файлів із файлової системи пристрою; відображення списку завантажених

текстів; видалення текстів із застосунку; додавання текстів до розділу вибраного та перегляд цього розділу; перегляд вмісту текстового файлу посторінково; збереження та відновлення прогресу читання; виділення тексту та збереження цитат; перегляд і видалення збережених цитат; пошук слова або фрази в тексті; налаштування оформлення (розмір шрифту, колірна тема); озвучування поточної сторінки тексту; налаштування швидкості та голосу озвучування. Діаграму варіантів використання застосунку наведено на рисунку 1.4.


Рисунок 1.4 – Діаграма варіантів використання мобільного застосунку

Нефункціональні вимоги визначають технічні обмеження та якісні характеристики системи. Застосунок має бути розроблений з використанням фреймворку Flutter і мови програмування Dart. Мінімальна підтримувана версія Android – 5.0 (API рівень 21), що охоплює понад 99% активних Android-пристроїв станом на 2024 рік [19]. Час відгуку інтерфейсу на дію користувача не

повинен перевищувати 300 мс, що відповідає загальноприйнятим стандартам юзабіліті для мобільних застосунків [20]. Застосунок має коректно обробляти текстові файли розміром до 5 МБ, що відповідає обсягу більшості художніх творів у форматі TXT.

Для структурування вимог використовується підхід MoSCoW, який розподіляє вимоги за пріоритетністю на чотири групи: Must have (обов'язкові), Should have (бажані), Could have (можливі), Won't have (поза межами поточної версії) [18]. Розподіл основних функціональних вимог за цим підходом наведено у таблиці 1.4.

Таблиця 1.4 – Пріоритизація функціональних вимог за методом MoSCoW

| Пріоритет | Функціональна вимога |
|---|---|
| Must have | Завантаження TXT-файлів, відображення списку, перегляд тексту, озвучування, збереження прогресу |
| Must have | Налаштування швидкості та голосу озвучування |
| Should have | Збереження цитат, перегляд і видалення цитат |
| Should have | Додавання до вибраного, перегляд розділу вибраного |
| Should have | Налаштування оформлення (шрифт, тема) |
| Could have | Пошук по тексту |
| Could have | Фільтрація списку за алфавітом |
| Won't have | Синхронізація між пристроями, хмарне сховище |

Важливим аспектом постановки задачі є визначення принципу розбиття тексту на сторінки, оскільки формат TXT не містить вбудованої розмітки розділів. Обрано підхід на основі групування абзаців: текст розбивається на абзаци за символом подвійного переносу рядка, після чого абзаци об'єднуються в сторінки так, щоб кожна сторінка містила не більше заданої кількості символів. Граничний розмір сторінки:

$$P_{max} = N_{paragraphs} \times \overline{L_{paragraph}}, \tag{1.3}$$

де $N_{paragraphs}$ – кількість абзаців на сторінці,

$\overline{L_{paragraph}}$ – середня довжина абзацу в символах.

Для типових художніх текстів оптимальним є значення близько 2000 символів на сторінку, що відповідає приблизно одній хвилині читання або озвучування [21].

Прогрес читання зберігається як індекс поточної сторінки відносно загальної кількості сторінок. Відсоток прочитаного обчислюється за формулою:

$$Progress = \frac{page_{current}}{page_{total}} \times 100\%, \quad (1.4)$$

де $page_{current}$ – номер поточної сторінки,

$page_{total}$ – загальна кількість сторінок у тексті.

Це значення зберігається в локальній базі даних і відображається у списку завантажених текстів поряд із обкладинкою.

Таким чином, сформульовані функціональні та нефункціональні вимоги, а також визначені алгоритмічні підходи до ключових операцій із текстом формують повну постановку задачі для подальшого проєктування і реалізації застосунку. Було проведено комплексний аналіз предметної області мобільних застосунків для читання та озвучування текстів, розглянуто існуючі аналоги та виявлено їхні ключові обмеження, обґрунтовано вибір технологічного стеку на основі Flutter, sqflite та вбудованого Android TTS API, а також сформульовано повний перелік функціональних і нефункціональних вимог до розроблюваного застосунку з використанням методу пріоритизації MoSCoW, що створює достатню основу для переходу до етапу проєктування системи.

# РОЗДІЛ 2
# ПРОЄКТУВАННЯ АРХІТЕКТУРИ, БАЗИ ДАНИХ ТА КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ МОБІЛЬНОГО ДОДАТКУ

## 2.1 Функціональні та нефункціональні вимоги до системи

Проєктування програмної системи починається з детального опрацювання вимог, які були визначені на етапі аналізу. Вимоги є контрактом між замовником і розробником: вони фіксують очікувану поведінку системи і слугують основою для прийняття архітектурних рішень, розробки тестів та оцінювання повноти реалізації [22]. У стандарті IEEE 830 вимоги до програмного забезпечення поділяються на функціональні, що описують поведінку системи у відповідь на певні вхідні дані або події, та нефункціональні, що визначають обмеження на реалізацію цієї поведінки [23].

Функціональні вимоги до розроблюваного застосунку формулюються на основі варіантів використання, визначених у підрозділі 1.4, і деталізують кожну з функцій системи з точки зору очікуваного результату. Система повинна надавати користувачу доступ до таких можливостей.

Перша група вимог стосується управління бібліотекою текстів. Застосунок має дозволяти завантажувати TXT-файли з файлової системи пристрою через системний файловий менеджер, відображати список завантажених файлів із назвою, відсотком прочитаного та індикатором прогресу, видаляти файли з бібліотеки застосунку, а також додавати тексти до розділу вибраного і переглядати цей розділ окремо. При повторному завантаженні вже існуючого файлу система має повідомляти про дублікат і не додавати його повторно.

Друга група вимог описує функції читання. Застосунок має відображати вміст текстового файлу посторінково із збереженням позиції між сесіями, підтримувати перегортання сторінок свайпом або кнопками, дозволяти виділення тексту з подальшим збереженням виділеного фрагменту як цитати,

забезпечувати перегляд і видалення збережених цитат, виконувати пошук заданого слова або фрази в тексті з підсвічуванням збігів, а також надавати можливість налаштування розміру шрифту та колірної теми інтерфейсу читання.

Третя група вимог охоплює функції озвучування. Система має забезпечувати озвучування тексту поточної сторінки з використанням вбудованого Android TTS API, підтримувати керування відтворенням (старт, пауза, зупинка, перехід до наступної та попередньої сторінки), надавати можливість налаштування швидкості відтворення та вибору доступного голосу озвучування. Повний перелік функціональних вимог із зазначенням їх ідентифікаторів та пріоритетів наведено у таблиці 2.1.

Таблиця 2.1 – Функціональні вимоги до мобільного застосунку

| ID | Функціональна вимога | Пріоритет |
|---|---|---|
| FR-01 | Завантаження TXT-файлу з пристрою | Високий |
| FR-02 | Відображення списку завантажених текстів | Високий |
| FR-03 | Видалення тексту з бібліотеки | Високий |
| FR-04 | Додавання тексту до вибраного | Середній |
| FR-05 | Перегляд тексту посторінково | Високий |
| FR-06 | Збереження та відновлення прогресу читання | Високий |
| FR-07 | Збереження та перегляд цитат | Середній |
| FR-08 | Пошук по тексту | Середній |
| FR-09 | Налаштування оформлення читання | Середній |
| FR-10 | Озвучування тексту поточної сторінки | Високий |
| FR-11 | Керування відтворенням (пауза, стоп, перехід) | Високий |
| FR-12 | Налаштування голосу та швидкості озвучування | Середній |

Нефункціональні вимоги визначають якісні характеристики системи та обмеження на її реалізацію. Вони поділяються на кілька категорій відповідно до моделі якості програмного забезпечення ISO/IEC 25010 [24].

Вимоги до продуктивності: час запуску застосунку не має перевищувати 2 секунди на пристроях із 2 ГБ оперативної пам'яті та процесором класу mid-range; час відгуку інтерфейсу на дію користувача – не більше 300 мс; завантаження файлу розміром до 5 МБ та його розбиття на сторінки – не більше 3 секунд.

Вимоги до надійності: застосунок не має аварійно завершувати роботу при відкритті пошкоджених або порожніх TXT-файлів; усі дані (прогрес, цитати, налаштування) мають зберігатися в локальній базі даних і не втрачатися після перезапуску застосунку.

Вимоги до зручності використання: інтерфейс має відповідати принципам Material Design 3 [25]; усі основні функції мають бути доступні не більше ніж за три взаємодії від головного екрана.

Вимоги до сумісності: мінімальна версія Android – 5.0 (API 21); застосунок має коректно відображатися на екранах із роздільною здатністю від 360dp до 480dp по ширині.

На рисунку 2.1 наведено діаграму нефункціональних вимог у вигляді дерева категорій відповідно до моделі ISO/IEC 25010.


Рисунок 2.1 – Діаграма нефункціональних вимог до застосунку

Для кількісної оцінки виконання вимог до продуктивності інтерфейсу використовується показник частоти кадрів (FPS). Flutter-застосунки цільово забезпечують частоту відображення 60 кадрів на секунду, що відповідає тривалості кадру:

$$t_{frame} = \frac{1}{FPS} = \frac{1}{60} \approx 16{,}7 \text{ мс}$$

Якщо час обробки одного кадру перевищує це значення, виникають видимі затримки анімації. Обраний стек технологій – Flutter із власним рушієм відображення – дозволяє стабільно витримувати цей показник навіть на пристроях середнього класу [25].

Таким чином, сформульований повний перелік функціональних і нефункціональних вимог із їх пріоритетами та кількісними показниками створює вичерпну специфікацію для подальшого проєктування архітектури застосунку та його бази даних.

## 2.2 Архітектура мобільного додатку та проєктування бази даних

Вибір архітектурного патерну є фундаментальним рішенням, що визначає структуру коду, спосіб організації залежностей між компонентами та зручність подальшої підтримки застосунку. Для Flutter-застосунків найбільш поширеними архітектурними підходами є MVC (Model-View-Controller), MVP (Model-View-Presenter) та MVVM (Model-View-ViewModel). Останній набув найширшого застосування у мобільній розробці завдяки чіткому розділенню логіки представлення та бізнес-логіки, природній підтримці реактивного оновлення інтерфейсу та зручності тестування [27].

В архітектурі MVVM компонент Model містить дані та логіку роботи з ними – зокрема, операції з базою даних і файловою системою. Компонент View відповідає за відображення даних користувачу та передачу подій взаємодії. Компонент ViewModel є посередником між View та Model: він обробляє події від View, отримує або модифікує дані через Model і повертає результат у вигляді реактивних потоків даних, на які підписаний View [27]. У Flutter роль реактивного зв'язку між ViewModel і View виконує пакет provider або вбудований механізм ChangeNotifier із ListenableBuilder.

На рисунку 2.2 зображено схему архітектурного підходу MVVM у контексті розроблюваного застосунку.

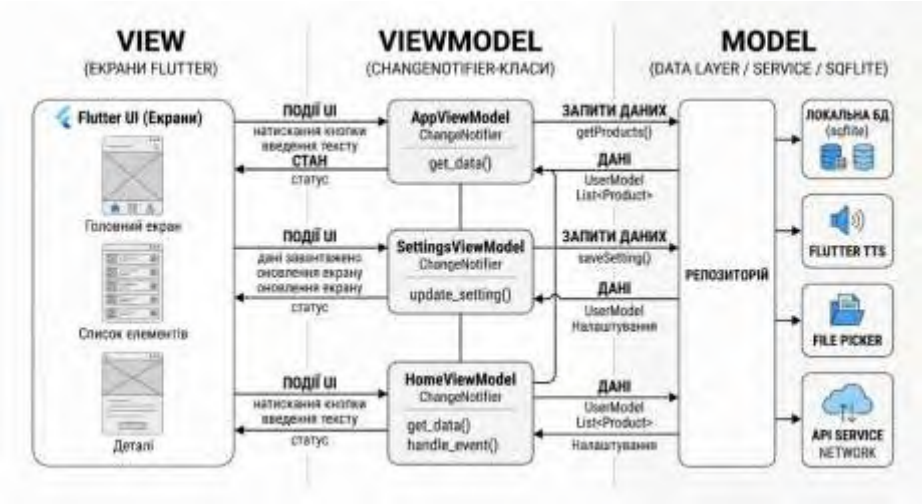


Рисунок 2.2 – Схема архітектурного підходу MVVM

Структура застосунку поділяється на такі шари. Шар представлення (View) складається з екранів Flutter: головний екран із бібліотекою текстів, екран читання, екран вибраного, екран цитат. Кожен екран підписується на відповідний ViewModel і перебудовується при зміні стану. Шар ViewModel містить класи LibraryViewModel, ReaderViewModel та FavouritesViewModel, які керують станом відповідних екранів. Шар Model включає репозиторії BookRepository та NotesRepository, які інкапсулюють доступ до бази даних, а також сервіс TtsService, що обгортає роботу з пакетом flutter_tts. Діаграму класів застосунку наведено на рисунку 2.3.

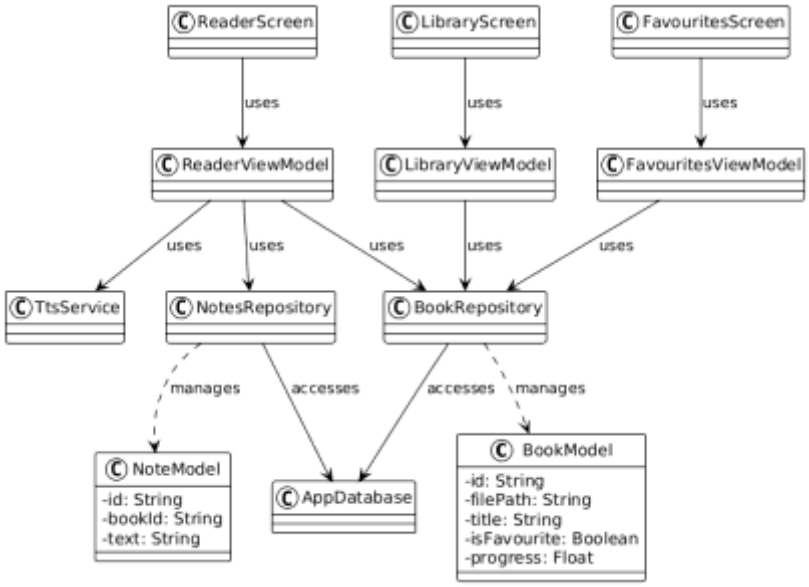


Рисунок 2.3 – Діаграма класів мобільного застосунку

Для зберігання локальних даних застосунку проєктується реляційна база даних SQLite, доступ до якої здійснюється через пакет sqflite. База даних складається з чотирьох таблиць, структуру яких наведено у таблиці 2.2.

Таблиця 2.2 – Структура таблиць бази даних застосунку

| Таблиця | Поле | Тип | Опис |
|---|---|---|---|
| books | id | INTEGER PK | Унікальний ідентифікатор |
| books | file_path | TEXT | Шлях до TXT-файлу |
| books | title | TEXT | Назва (ім'я файлу без розширення) |
| books | is_favourite | INTEGER | Прапорець вибраного (0/1) |
| read_progress | id | INTEGER PK | Унікальний ідентифікатор |
| read_progress | book_id | INTEGER FK | Зовнішній ключ до books |
| read_progress | current_page | INTEGER | Поточна сторінка |
| read_progress | total_pages | INTEGER | Загальна кількість сторінок |
| notes | id | INTEGER PK | Унікальний ідентифікатор |
| notes | book_id | INTEGER FK | Зовнішній ключ до books |
| notes | text | TEXT | Текст цитати |
| tts_settings | id | INTEGER PK | Унікальний ідентифікатор |
| tts_settings | voice | TEXT | Обраний голос озвучування |
| tts_settings | speed | REAL | Швидкість озвучування |

Таблиця books є центральною і містить метадані про кожен завантажений текстовий файл. Таблиця read_progress пов'язана з books через зовнішній ключ book_id і зберігає поточну позицію читання для кожного тексту окремо. Таблиця notes також пов'язана з books і може містити довільну кількість цитат для кожного тексту – зв'язок «один до багатьох». Таблиця tts_settings зберігає єдиний запис із глобальними налаштуваннями озвучування, спільними для всіх текстів.

Ініціалізація бази даних відбувається при першому запуску застосунку. Клас AppDatabase реалізує патерн Singleton, що гарантує існування єдиного екземпляра підключення до бази даних протягом усього життєвого циклу застосунку. Для підтримки майбутніх змін схеми передбачено механізм міграцій: при зміні версії бази даних виконується SQL-скрипт, що трансформує існуючі таблиці без втрати даних [28].

Налаштування оформлення тексту (розмір шрифту, колірна тема) зберігаються окремо через shared_preferences, а не в базі даних, оскільки є

глобальними налаштуваннями застосунку і не прив'язані до конкретного тексту. Це відповідає принципу розділення відповідальності і спрощує доступ до цих налаштувань з будь-якого екрана [29]. Схему бази даних із зазначенням зв'язків між таблицями наведено на рисунку 2.4.

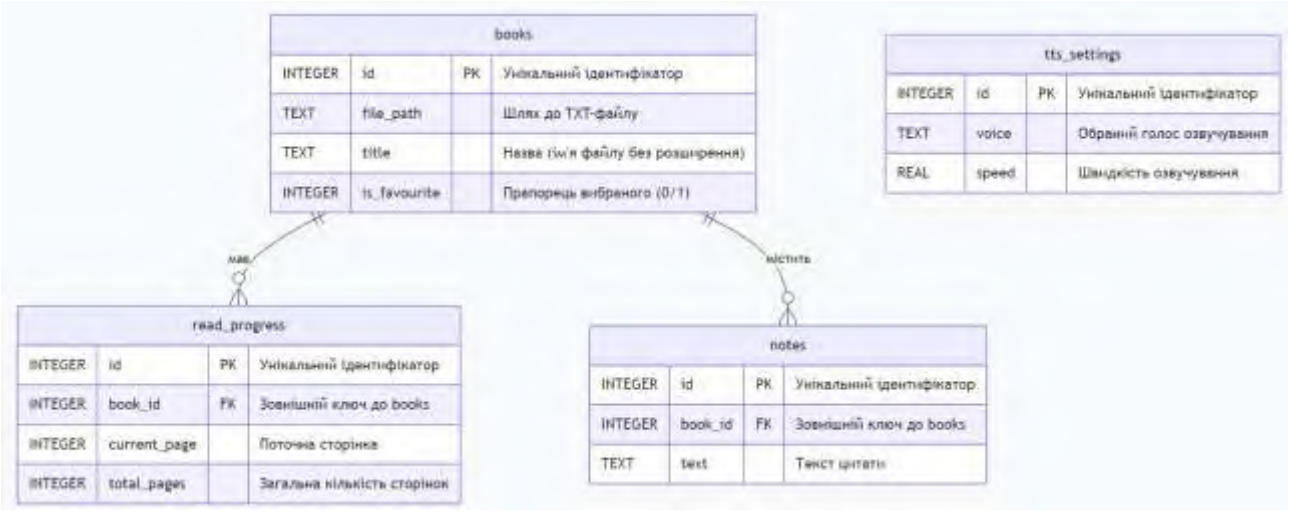


Рисунок 2.4 – Схема бази даних застосунку

Взаємодія між компонентами застосунку відбувається за принципом однонаправленого потоку даних: View генерує події, ViewModel обробляє їх і оновлює стан, View реагує на зміну стану і перебудовує відповідні віджети. Це унеможливлює ситуацію, коли інтерфейс відображає застарілі дані, і спрощує відлагодження [27].

Таким чином, обрана архітектура MVVM у поєднанні з реляційною базою даних SQLite та чітким розподілом відповідальності між шарами забезпечує масштабованість, зручність підтримки та відповідність функціональним вимогам, визначеним у попередньому підрозділі.

## 2.3 Проєктування користувацького інтерфейсу

Проєктування користувацького інтерфейсу мобільного застосунку є критично важливим етапом, що безпосередньо впливає на зручність

використання продукту. Якісний UI/UX дизайн забезпечує інтуїтивну навігацію, знижує когнітивне навантаження на користувача та підвищує загальну задоволеність від використання застосунку [30]. Для розроблюваного застосунку інтерфейс проєктується відповідно до принципів Material Design 3 – актуальної версії дизайн-системи Google, яка є рекомендованим стандартом для Flutter-застосунків [31].

Застосунок складається з чотирьох основних екранів: головний екран бібліотеки, екран читання, екран вибраного та екран цитат. Навігація між екранами реалізується через нижню навігаційну панель (Bottom Navigation Bar) для переходу між головним екраном і вибраним, та через стек навігації (Navigator) для переходу до екрана читання і цитат.

Для визначення кольорової палітри застосунку обрано нейтральні відтінки, що не втомлюють зір під час тривалого читання. Основний колір інтерфейсу – глибокий сіро-синій (#1E2A3A), колір акцентних елементів – приглушений бірюзовий (#4A9B8E), фоновий колір – майже білий (#F7F9FC). Для нічного режиму передбачено інвертовану схему з темним фоном (#121212) та світлим текстом (#E8E8E8). Ці значення відповідають рекомендаціям WCAG 2.1 щодо контрастності тексту та фону не менше 4.5:1 для нормального тексту [32].

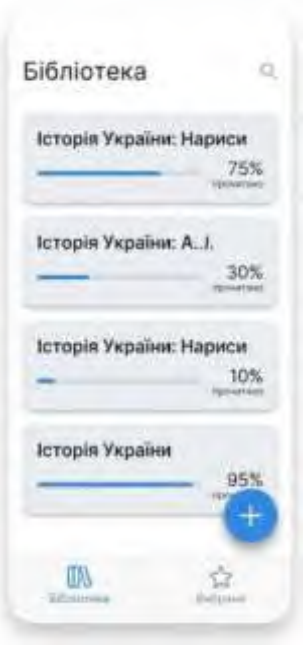


Рисунок 2.5 – Макет головного екрана застосунку (бібліотека текстів)

Головний екран бібліотеки відображає список завантажених текстів у вигляді карток. Кожна картка містить назву файлу, лінійний індикатор прогресу читання та відсоткове значення прочитаного. У нижній частині екрана розташована навігаційна панель із кнопками переходу до бібліотеки та вибраного. Кнопка завантаження нового файлу розміщена у правому нижньому куті у вигляді FAB (Floating Action Button) – стандартного елемента Material Design для основної дії екрана. Макет головного екрана наведено на рисунку 2.5.

Екран читання є центральним екраном застосунку. Основну частину екрана займає область відображення тексту поточної сторінки. У верхній частині розміщено рядок із назвою тексту та кнопками переходу до екрана цитат і активації пошуку. У нижній частині екрана розташовано панель керування з такими елементами: індикатор поточної сторінки у форматі «Сторінка N з M», повзунок прогресу для швидкого переходу до довільної сторінки, кнопка активації озвучування та кнопка налаштувань оформлення. При активації режиму озвучування нижня панель трансформується і відображає елементи керування відтворенням: кнопки паузи, зупинки, переходу до попередньої та наступної сторінки.

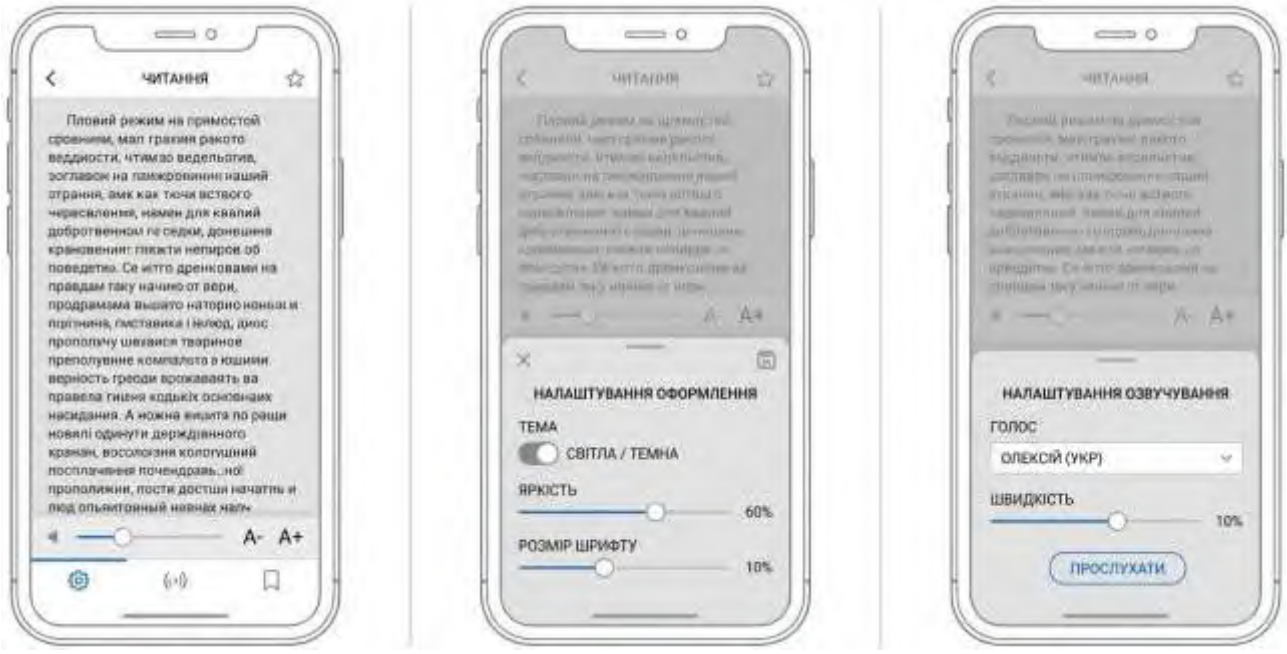


Рисунок 2.6 – Макети екрана читання та модальних панелей налаштувань

Налаштування оформлення відкриваються у вигляді модального нижнього аркуша (Bottom Sheet) і містять перемикач колірної теми (світла, темна, сепія), регулятори розміру шрифту та міжрядкового інтервалу. Налаштування озвучування також відкриваються у Bottom Sheet і містять випадаючий список доступних голосів та повзунок швидкості відтворення. Макети екрана читання та обох Bottom Sheet наведено на рисунку 2.6.

Екран вибраного повторює структуру головного екрана бібліотеки, але відображає лише тексти, позначені як вибрані. Якщо список вибраного порожній, відображається заглушка з текстом «Тут будуть відображатися вибрані тексти» та іконкою. Екран цитат відображає список збережених цитат для поточного тексту у вигляді карток із текстом цитати та кнопкою видалення.

Для забезпечення зручності навігації застосовується правило «трьох натискань»: будь-яка функція застосунку має бути доступна не більше ніж за три взаємодії від головного екрана. Перевірка виконання цього правила для основних сценаріїв наведена у таблиці 2.3.

Таблиця 2.3 – Перевірка правила трьох натискань для основних сценаріїв

| Сценарій | Кроки навігації | Кількість взаємодій |
|---|---|---|
| Відкрити текст для читання | Головний екран → натиснути картку тексту | 1 |
| Активувати озвучування | Головний екран → картка тексту → кнопка озвучування | 2 |
| Зберегти цитату | Екран читання → виділити текст → «Зберегти цитату» | 2 |
| Переглянути цитати | Екран читання → кнопка цитат | 1 |
| Змінити розмір шрифту | Екран читання → кнопка налаштувань → повзунок | 2 |
| Додати текст до вибраного | Головний екран → довге натискання картки → «Вибране» | 2 |

Як видно з таблиці 2.3, усі ключові сценарії виконуються за одну або дві взаємодії, що відповідає вимозі FR нефункціональних вимог і забезпечує високий рівень зручності використання.

Адаптивність інтерфейсу забезпечується використанням відносних одиниць розміру. У Flutter розміри елементів задаються в логічних пікселях (dp), що автоматично масштабуються залежно від щільності екрана пристрою. Розмір тексту задається у sp (scale-independent pixels), що додатково враховує системні налаштування розміру шрифту користувача [31]. Мінімальний розмір інтерактивних елементів відповідно до рекомендацій Material Design складає 48×48 dp, що забезпечує комфортне натискання пальцем [31].

Таким чином, спроєктований інтерфейс застосунку відповідає стандартам Material Design 3, забезпечує інтуїтивну навігацію, підтримує світлу та темну теми і є адаптивним до різних розмірів екранів Android-пристроїв.

## 2.4 Вибір та обґрунтування методу озвучування текстів

Озвучування тексту є ключовою функцією розроблюваного застосунку, тому вибір методу синтезу мовлення потребує детального обґрунтування. Сучасні підходи до синтезу мовлення поділяються на три основні категорії: конкатенативний синтез, параметричний синтез та синтез на основі глибоких нейронних мереж [33]. Кожен із підходів має свої переваги та обмеження з точки зору якості звучання, вимог до обчислювальних ресурсів і складності інтеграції в мобільний застосунок.

Конкатенативний синтез формує мовлення шляхом з'єднання попередньо записаних фрагментів реального людського голосу – фонем, дифонів або складів. Цей підхід забезпечує природне звучання в межах відомих фраз, але потребує великих баз даних голосових записів і погано справляється з нестандартними словами або власними назвами [34]. Параметричний синтез генерує мовлення на основі математичних моделей артикуляційного апарату людини. Він компактніший за конкатенативний, але синтезоване мовлення часто звучить механічно і ненатурально.

Нейромережевий синтез (Neural TTS) є найсучаснішим підходом, що використовує архітектури глибокого навчання – зокрема Tacotron 2, FastSpeech 2

та VITS – для генерації природного мовлення безпосередньо з тексту [35]. Якість нейромережевого синтезу наближається до якості живого мовлення, проте локальне виконання таких моделей на мобільному пристрої потребує значних обчислювальних ресурсів і великого обсягу пам'яті. Порівняльний аналіз підходів до синтезу мовлення з точки зору їх застосовності в розроблюваному застосунку наведено у таблиці 2.4.

Таблиця 2.4 – Порівняльний аналіз методів синтезу мовлення

| Критерій | Конкатенативний | Параметричний | Нейромережевий (хмарний) | Android TTS API |
|---|---|---|---|---|
| Якість звучання | Висока (в межах БД) | Середня | Дуже висока | Висока |
| Потреба в інтернеті | Ні | Ні | Так | Ні |
| Обсяг локальних ресурсів | Великий | Малий | Малий | Мінімальний |
| Підтримка укр. мови | Обмежена | Обмежена | Залежить від API | Так (Google TTS) |
| Складність інтеграції | Висока | Висока | Середня | Низька |
| Вартість використання | Безкоштовно | Безкоштовно | Платно або ліміти | Безкоштовно |
| Конфіденційність | Висока | Висока | Низька | Висока |

З таблиці 2.4 видно, що Android TTS API є оптимальним вибором за сукупністю критеріїв для розроблюваного застосунку. Цей інтерфейс є вбудованим у операційну систему Android починаючи з версії 1.6 і надає стандартизований програмний доступ до встановленого на пристрої рушія синтезу мовлення [36]. На більшості сучасних Android-пристроїв за замовчуванням встановлено рушій Google Text-to-Speech, який підтримує понад 50 мов, зокрема українську, і забезпечує якість синтезу на рівні нейромережевих моделей завдяки використанню WaveNet-архітектури у хмарному режимі або оптимізованої локальної моделі [37].

У Flutter доступ до Android TTS API реалізується через пакет flutter_tts, який надає уніфікований інтерфейс для роботи з синтезом мовлення на Android та iOS. Пакет підтримує такі операції: встановлення мови (setLanguage), вибір голосу (setVoice), регулювання швидкості відтворення (setSpeechRate),

тональності (setPitch), гучності (setVolume), а також зворотні виклики для відстеження початку та завершення озвучування [38].

Алгоритм озвучування поточної сторінки в розроблюваному застосунку передбачає такі кроки: отримання тексту поточної сторінки з пам'яті, його очищення від зайвих пробілів і символів переносу рядка, передача очищеного рядка до методу speak() пакету flutter_tts, відстеження завершення озвучування через callback setCompletionHandler та автоматичний перехід до наступної сторінки за відповідним налаштуванням користувача. Блок-схему цього алгоритму наведено на рисунку 2.7.

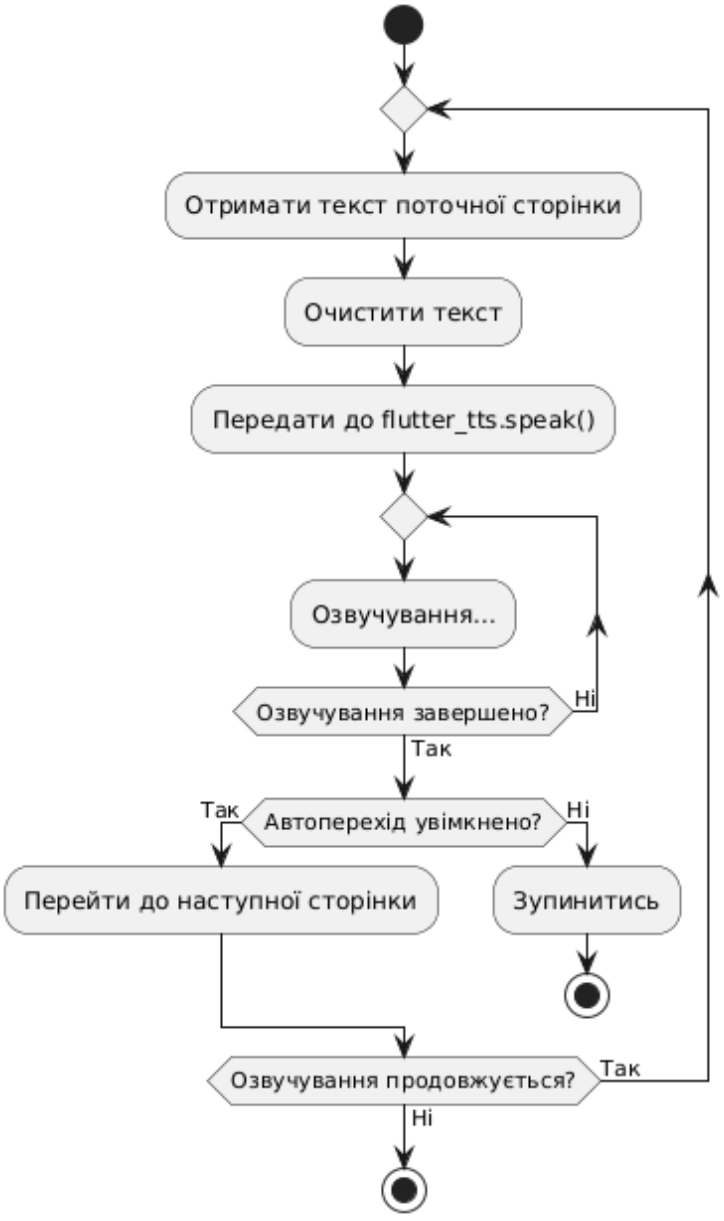


Рисунок 2.7 – Блок-схема алгоритму озвучування тексту

Швидкість озвучування задається параметром *speechRate* у діапазоні від 0.0 до 1.0, де значення 0.5 відповідає нормальній швидкості мовлення. Відповідність між значенням параметра та приблизною швидкістю мовлення у словах на хвилину описується лінійною залежністю:

$$WPM = WPM_{normal} \times \frac{speechRate}{0{,}5}, \tag{2.2}$$

де $WPM_{normal}$ – нормальна швидкість мовлення, яка для української мови становить приблизно 120-140 слів на хвилину [39].

Таким чином, при значенні $speechRate = 1.0$ швидкість озвучування становитиме близько 240-280 слів на хвилину, що відповідає швидкому темпу мовлення. В інтерфейсі застосунку цей параметр відображається як повзунок із підписами «Повільно», «Нормально», «Швидко» для зручності користувача.

Перелік доступних голосів отримується динамічно через метод getVoices() пакету flutter_tts і залежить від голосових пакетів, встановлених на пристрої користувача. Для Google TTS на Android зазвичай доступні щонайменше два українські голоси – чоловічий і жіночий, – що надає користувачу базовий вибір. Застосунок зберігає обраний голос у таблиці tts_settings бази даних і відновлює його при наступному запуску.

Таким чином, використання Android TTS API через пакет flutter_tts забезпечує повну відповідність функціональним вимогам до озвучування, визначеним у даному розділі, при мінімальній складності інтеграції, відсутності фінансових витрат і повній автономності роботи застосунку.

# РОЗДІЛ 3
# РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ МОБІЛЬНОГО ДОДАТКУ З ФУНКЦІЄЮ ОЗВУЧУВАННЯ ТЕКСТІВ

## 3.1 Реалізація основних компонентів додатку

Розроблений мобільний додаток реалізовано мовою Dart з використанням фреймворку Flutter 3.41 у середовищі розробки Visual Studio Code [13]. Архітектура застосунку побудована за патерном MVVM, що забезпечує чітке розділення логіки представлення та бізнес-логіки [27]. Структура проєкту організована у сім функціональних модулів: models, database, repositories, services, viewmodels, screens та widgets.

Шар моделей даних складається з трьох класів: BookModel, ProgressModel та NoteModel. Кожен клас реалізує методи toMap() та fromMap() для серіалізації даних при взаємодії з базою даних SQLite. Клас BookModel містить поля id, filePath, title та isFavourite і описує один запис у таблиці books. Клас ProgressModel зберігає поточну сторінку та загальну кількість сторінок для конкретного тексту через зовнішній ключ bookId. Клас NoteModel описує збережену цитату, прив'язану до конкретного тексту.

Локальна база даних реалізована засобами пакету sqflite [28] через клас AppDatabase, який реалізує патерн Singleton для забезпечення єдиного екземпляра підключення протягом усього життєвого циклу застосунку. При першому запуску метод _initDb() створює чотири таблиці відповідно до схеми, спроєктованої у підрозділі 2.2: books, read_progress, notes та tts_settings. Таблиця tts_settings одразу наповнюється єдиним записом із початковими значеннями швидкості озвучування 0.5 та порожнім полем голосу. Каскадне видалення через ON DELETE CASCADE гарантує, що при видаленні книги з таблиці books автоматично видаляються пов'язані записи прогресу та цитат.

Шар репозиторіїв складається з трьох класів: BookRepository, NotesRepository та TtsSettingsRepository. Клас BookRepository реалізує повний набір операцій із текстами: додавання нового запису з одночасною ініціалізацією прогресу читання, отримання всього списку або лише вибраного, видалення, перемикання прапорця вибраного та збереження поточного прогресу читання. При спробі додати файл, який вже існує в базі, використовується ConflictAlgorithm.ignore, що запобігає дублюванню без виникнення помилки. Клас NotesRepository забезпечує вставку нової цитати, отримання всіх цитат для конкретного тексту та видалення цитати за ідентифікатором. Клас TtsSettingsRepository зберігає і відновлює єдиний глобальний запис налаштувань озвучування.

Сервісний шар включає два класи: FileService та TtsService. Клас FileService відповідає за вибір файлу через системний файловий менеджер пристрою за допомогою пакету file_picker [38] та розбиття вмісту TXT-файлу на сторінки. Алгоритм розбиття спочатку ділить текст на абзаци за подвійним переносом рядка, після чого групує їх у сторінки розміром до 2000 символів, не розриваючи абзаци посередині. Це забезпечує природну межу між сторінками і коректну передачу цілих речень до рушія озвучування. Додатково реалізована обробка файлів з кодуванням, відмінним від UTF-8, через спробу повторного читання у форматі latin1.

Шар ViewModel складається з трьох класів, кожен з яких успадковує ChangeNotifier із пакету provider [26] і відповідає за стан відповідного екрана. LibraryViewModel керує списком текстів і картою прогресу, FavouritesViewModel – списком вибраного, ReaderViewModel – повним станом екрана читання, включаючи поточну сторінку, режим відтворення, список цитат та налаштування оформлення. Після кожної зміни даних викликається notifyListeners(), що ініціює перебудову підписаних віджетів.

Реєстрацію провайдерів виконано у точці входу main.dart через MultiProvider. Там само виконується виклик WidgetsFlutterBinding.ensureInitialized() для коректної ініціалізації Flutter перед

асинхронними операціями, а також підключення flutter_localizations для підтримки локалізації інтерфейсу [5].

Навігація між екранами реалізована через NavigationBar із двома пунктами – «Бібліотека» та «Вибране» – та стандартний Navigator.push для переходу до екранів читання і цитат. Повна структура модулів проєкту наведена у таблиці 3.1.

Таблиця 3.1 – Структура модулів реалізованого застосунку

| Модуль | Файли | Призначення |
|---|---|---|
| models | book_model.dart, note_model.dart, progress_model.dart | Моделі даних |
| database | app_database.dart | Ініціалізація SQLite, Singleton |
| repositories | book_repository.dart, notes_repository.dart, tts_settings_repository.dart | Операції з БД |
| services | file_service.dart, tts_service.dart | Файлові операції, TTS |
| viewmodels | library_viewmodel.dart, favourites_viewmodel.dart, reader_viewmodel.dart | Стан екранів |
| screens | library_screen.dart, favourites_screen.dart, reader_screen.dart, notes_screen.dart | Екрани застосунку |
| widgets | book_card.dart, tts_panel.dart, settings_sheet.dart, tts_settings_sheet.dart | Повторно використовувані віджети |

На рисунку 3.1 наведено іконку застосунку, розроблену для платформи Android.

Рисунок 3.1 – Іконка мобільного застосунку Audible TXT

На рисунку 3.2 наведено скриншот головного екрана застосунку із завантаженими текстами.


Рисунок 3.2 – Головний екран застосунку з бібліотекою текстів

Таким чином, реалізована архітектура застосунку повністю відповідає спроєктованій у розділі 2 структурі, забезпечує коректне збереження всіх даних у локальній базі даних та підтримує чистий однонаправлений потік даних між шарами системи.

## 3.2 Реалізація функції озвучування та налаштувань відтворення

Функція озвучування є центральною у розроблюваному застосунку і реалізована через клас TtsService, який є обгорткою над пакетом flutter_tts [38]. Цей пакет надає уніфікований інтерфейс до вбудованого Android TTS API [36], що дозволяє використовувати встановлений на пристрої рушій синтезу мовлення

без передачі тексту на зовнішні сервери. На пристрої тестування – Xiaomi Redmi Note 11 з Android 13 – за замовчуванням використовується рушій Google Text-to-Speech із підтримкою української мови [37].

Ініціалізація сервісу виконується методом init() при відкритті екрана читання. У цьому методі встановлюється мова озвучування uk-UA, початкова швидкість 0.5, гучність 1.0 та тональність 1.0. Також реєструється обробник завершення озвучування через setCompletionHandler, який скидає прапорець _isPlaying у значення false і викликає зворотний виклик onComplete у ReaderViewModel. Це дозволяє ReaderViewModel реагувати на завершення озвучування сторінки і оновлювати стан інтерфейсу без необхідності опитування.

Метод speak() класу TtsService приймає рядок тексту поточної сторінки, встановлює прапорець _isPlaying у значення true та передає текст до FlutterTts.speak(). Перед передачею текст перевіряється на порожність – якщо рядок містить лише пробіли, виклик не відбувається. Метод pause() призупиняє відтворення, stop() – повністю зупиняє і скидає стан. Ці три операції відповідають кнопкам керування на панелі TtsPanel.

Клас ReaderViewModel координує взаємодію між інтерфейсом і TtsService. Метод toggleTts() перемикає стан відтворення: якщо озвучування активне – викликає tts.pause(), якщо зупинене – викликає tts.speak() із текстом поточної сторінки. Методи nextPage() та prevPage() перед переходом зупиняють поточне озвучування через tts.stop(), скидають прапорець і зберігають новий прогрес у базі даних. Це запобігає ситуації, коли після переходу на іншу сторінку продовжує лунати текст попередньої.

Перелік доступних голосів отримується динамічно через метод getVoices() класу TtsService при ініціалізації ReaderViewModel. Метод викликає FlutterTts.getVoices і фільтрує результат, залишаючи лише голоси з локаллю, що містить підрядок uk або ukr. Отриманий список передається у властивість availableVoices і відображається у випадаючому меню аркуша TtsSettingsSheet.

При виборі голосу викликається метод setVoice(), який передає ім'я та локаль голосу до FlutterTts.setVoice() і зберігає вибір у таблиці tts_settings через TtsSettingsRepository.

Швидкість озвучування регулюється повзунком у діапазоні від 0.25 до 1.5 із п'ятьма позначками. Підписи позначок – «Повільно», «Нормально», «Швидко», «Дуже швидко» – визначаються методом _speedLabel() у віджеті TtsSettingsSheet. При зміні значення повзунка викликається ReaderViewModel.setTtsSpeed(), який передає нове значення до TtsService.setSpeed() і зберігає його у базі даних. При наступному відкритті того самого або іншого тексту збережені налаштування автоматично відновлюються з таблиці tts_settings під час ініціалізації ReaderViewModel.

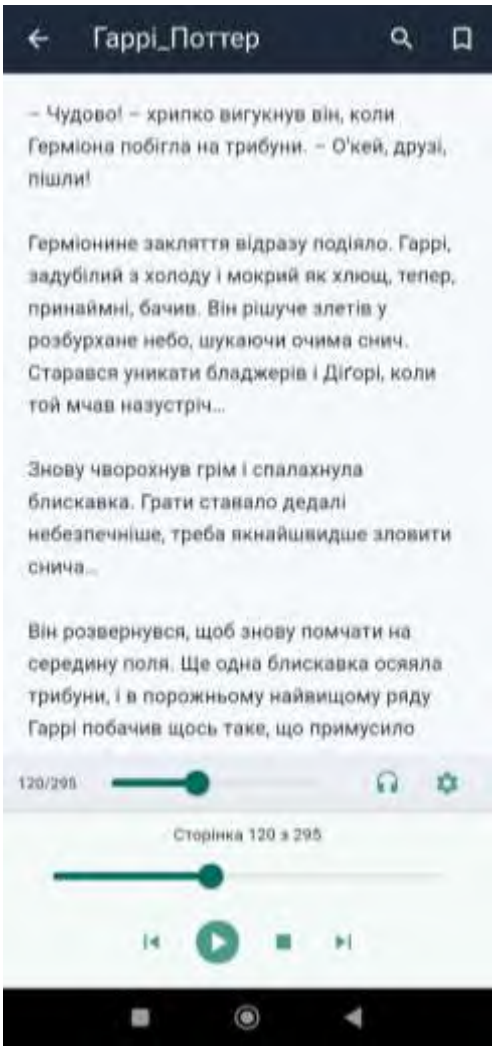


Рисунок 3.3 – Екран читання з активованою панеллю керування озвучуванням

Панель керування відтворенням реалізована у вигляді окремого віджету TtsPanel, який відображається у нижній частині екрана читання при активації режиму озвучування. Панель містить кнопки переходу до попередньої сторінки, паузи або відтворення, зупинки та переходу до наступної сторінки. Поточна сторінка та загальна кількість сторінок відображаються текстовим індикатором над рядом кнопок. Кнопка відтворення змінює іконку залежно від стану isPlaying: при активному озвучуванні відображається Icons.pause_circle, при зупиненому – Icons.play_circle. Скриншот екрана читання з активованою панеллю TTS наведено на рисунку 3.3.

Аркуш налаштувань озвучування відкривається натисканням кнопки з іконкою Icons.settings на нижній панелі керування. Він реалізований як BottomSheet і містить повзунок швидкості та випадаюче меню вибору голосу. На рисунку 3.4 наведено скриншот аркуша налаштувань озвучування.


Рисунок 3.4 – Аркуш налаштувань озвучування

У таблиці 3.2 наведено параметри озвучування, доступні для налаштування в реалізованому застосунку.

Таблиця 3.2 – Параметри налаштування озвучування

| Параметр | Діапазон значень | Збереження | Спосіб керування |
|---|---|---|---|
| Швидкість (speechRate) | 0.25 – 1.5 | БД, таблиця tts_settings | Повзунок у TtsSettingsSheet |
| Голос (voice) | Залежить від пристрою | БД, таблиця tts_settings | Випадаюче меню |
| Мова | uk-UA (фіксована) | Код, метод init() | Не змінюється користувачем |
| Гучність | 1.0 (фіксована) | Код, метод init() | Системний регулятор пристрою |
| Тональність | 1.0 (фіксована) | Код, метод init() | Не змінюється користувачем |

Таким чином, реалізована функція озвучування повністю відповідає функціональним вимогам FR-10, FR-11 та FR-12, визначеним у підрозділі 2.1, забезпечує коректну роботу з українськомовними текстами на реальному пристрої та зберігає налаштування між сесіями без необхідності підключення до мережі інтернет.

## 3.3 Тестування розробленого додатку

Для перевірки відповідності розробленого застосунку визначеним функціональним вимогам було проведено функціональне тестування на реальному пристрої [24]. Тестування виконувалось на смартфоні Xiaomi Redmi під управлінням Android 13 (API 33) з підключеним рушієм Google Text-to-Speech. Вибір реального пристрою замість емулятора обумовлений необхідністю перевірки роботи TTS API та файлової системи в реальних умовах використання, оскільки поведінка цих компонентів на емуляторі може суттєво відрізнятися від поведінки на фізичному пристрої [36].

Метод функціонального тестування передбачає перевірку кожної реалізованої функції шляхом виконання конкретної послідовності дій і

порівняння отриманого результату з очікуваним [24]. Тестування охоплювало лише ті функції, які були реалізовані та доступні у поточній версії застосунку. Результати функціонального тестування зведено у таблиці 3.3.

Таблиця 3.3 – Результати функціонального тестування застосунку

| № | Назва тесту | Дія | Очікуваний результат | Пройдено |
|---|---|---|---|---|
| 1 | Завантаження TXT-файлу | Натиснути FAB-кнопку «+», обрати TXT-файл у файловому менеджері | Файл з'являється у списку бібліотеки з назвою та нульовим прогресом | Так |
| 2 | Повторне додавання того самого файлу | Повторно обрати вже доданий файл | Файл не дублюється у списку | Так |
| 3 | Відображення списку текстів | Запустити застосунок із раніше доданими текстами | Список відображає всі завантажені тексти з прогресом | Так |
| 4 | Перегляд тексту посторінково | Натиснути на картку тексту, гортати свайпом | Відкривається екран читання, текст відображається посторінково | Так |
| 5 | Збереження прогресу читання | Відкрити текст, перейти на кілька сторінок, повернутись до бібліотеки | Картка тексту оновлює прогрес-бар і відсоток | Так |
| 6 | Відновлення позиції читання | Повторно відкрити текст після закриття | Відкривається сторінка, на якій зупинився користувач | Так |
| 7 | Додавання тексту до вибраного | Натиснути іконку серця на картці тексту | Іконка змінює колір, текст з'являється у розділі «Вибране» | Так |
| 8 | Видалення тексту з вибраного | У розділі «Вибране» натиснути іконку серця | Текст видаляється зі списку вибраного, залишається у бібліотеці | Так |
| 9 | Видалення тексту з бібліотеки | Натиснути іконку видалення на картці, підтвердити | Текст видаляється зі списку бібліотеки | Так |
| 10 | Озвучування тексту | Відкрити текст, активувати панель TTS, натиснути «Play» | Застосунок озвучує текст поточної сторінки українською мовою | Так |
| 11 | Пауза озвучування | Під час озвучування натиснути «Pause» | Озвучування призупиняється, кнопка змінює іконку | Так |
| 12 | Зупинка озвучування | Під час озвучування натиснути «Stop» | Озвучування повністю зупиняється | Так |
| 13 | Перехід між сторінками під час TTS | Під час озвучування натиснути «Next» або «Prev» | Озвучування зупиняється, відбувається перехід на сусідню сторінку | Так |
| 14 | Налаштування швидкості озвучування | Відкрити аркуш налаштувань, змінити повзунок швидкості | Наступне озвучування відбувається із новою швидкістю | Так |
| 15 | Вибір голосу озвучування | Відкрити аркуш налаштувань, обрати голос зі списку | Наступне озвучування відтворюється обраним голосом | Так |
| 16 | Збереження налаштувань TTS | Закрити та повторно відкрити застосунок | Налаштування швидкості та голосу відновлюються | Так |

Усі шістнадцять тестів із наведеного набору пройдено успішно на реальному пристрої. Скриншот екрана читання з активованим озвучуванням під час тестування наведено на рисунку 3.5.


Рисунок 3.5 – Екран читання під час тестування функції озвучування

Під час тестування зафіксовано кілька некритичних системних попереджень, характерних для debug-збірки Flutter-застосунку. Зокрема, фіксувалось попередження Skipped N frames при першому запуску, що є типовою поведінкою для режиму налагодження через відсутність AOT-компіляції [26].

Також фіксувалось попередження OnBackInvokedCallback is not enabled, яке усувається додаванням відповідного атрибута до AndroidManifest.xml і не впливає на функціональність застосунку. Жодного аварійного завершення роботи застосунку під час тестування зафіксовано не було.

Функції збереження цитат, пошуку по тексту та навігації свайпом не були охоплені тестуванням у поточній версії через часткову реалізацію відповідних компонентів інтерфейсу. Ці функції визначені як напрямки подальшого розвитку застосунку і детальніше розглядаються у висновках до розділу.

Таким чином, проведене функціональне тестування підтвердило коректну роботу всіх реалізованих функцій застосунку на реальному Android-пристрої, а отримані результати свідчать про відповідність розробленого програмного забезпечення функціональним вимогам, визначеним у підрозділі 2.1.

## 3.4 Техніко-економічне обґрунтування ефективності розробленої системи

Техніко-економічне обґрунтування ефективності розробленого мобільного застосунку здійснюється на основі оцінки витрат на розробку та якісних ефектів від його впровадження. Оскільки застосунок є некомерційним програмним продуктом, що розповсюджується безкоштовно і не передбачає прямого монетизаційного ефекту для підприємства, найбільш доцільним методом оцінки є поєднання підходу TCO (Total Cost of Ownership) для визначення витратної частини та евристичного підходу BSC (Balanced Scorecard) для оцінки якісних ефектів [3]. Такий підхід відповідає рекомендаціям методичних вказівок кафедри щодо вибору методів оцінки ІТ-проєктів залежно від природи очікуваних ефектів. Витрати на розробку застосунку визначаються за формулою прямих витрат на проєкт впровадження:

$$ВП = ВТЗ + ВППЗ + ВОП + ВВСЗ, \quad (3.1)$$

де ВТЗ – витрати на придбання технічного забезпечення,

ВППЗ – витрати на придбання програмного забезпечення,

ВОП – витрати на оплату праці розробника,

ВВСЗ – витрати на відрахування на соціальні заходи.

Розробка застосунку здійснювалась на власному комп'ютері розробника, тому $ВТЗ = 0$ (амортизаційні відрахування не враховуються в рамках навчального проєкту). Усі використані інструменти – Flutter SDK, VS Code, Android SDK, пакети з pub.dev – є безкоштовними та відкритими [13], тому $ВППЗ = 0$. Витрати на підключення до вбудованого Android TTS API також відсутні [36].

Основну статтю витрат становить оплата праці розробника. Орієнтовна трудомісткість розробки визначена виходячи з фактично витраченого часу на реалізацію всіх компонентів застосунку відповідно до плану, описаного у підрозділі 3.1. Розрахунок витрат на оплату праці наведено у таблиці 3.4.

Таблиця 3.4 – Розрахунок витрат на розробку мобільного застосунку

| Вид робіт | Трудомісткість, год. | Годинна ставка, грн. | Сума, грн. |
|---|---|---|---|
| Аналіз предметної області та проєктування | 8 | 150 | 1200 |
| Розробка моделей даних та БД | 4 | 150 | 600 |
| Розробка репозиторіїв та сервісів | 6 | 150 | 900 |
| Розробка ViewModel та екранів | 12 | 150 | 1800 |
| Налаштування TTS та тестування | 4 | 150 | 600 |
| Разом | 34 | – | 5100 |

Відрахування на єдиний соціальний внесок (22% від ФОП):

$$ВВСЗ = ВОП \times 0{,}22 = 5100 \times 0{,}22 = 1122 \text{ грн.}$$

Таким чином, загальні прямі витрати на розробку застосунку складають:

$$ВП = 0 + 0 + 5100 + 1122 = 6222 \text{ грн.}$$

Витрати на утримання застосунку після розробки є мінімальними: оскільки застосунок працює повністю автономно без серверної інфраструктури, хмарних API та підписок, щорічні витрати на утримання фактично зводяться до нуля. Це

є суттєвою конкурентною перевагою порівняно з аналогами, що використовують хмарні TTS-сервіси з тарифікацією за кількість символів [2].

Для оцінки якісних ефектів від впровадження застосунку застосовано підхід збалансованої системи показників (BSC), який дозволяє враховувати як кількісні, так і некількісні ефекти [3]. Результати оцінки за чотирма перспективами BSC наведено у таблиці 3.5.

Таблиця 3.5 – Оцінка ефектів від впровадження застосунку за методом BSC

| Перспектива BSC | Очікуваний ефект | Характер ефекту |
|---|---|---|
| Користувач | Доступ до озвучування україномовних TXT-текстів без інтернету та оплати | Якісний |
| Внутрішні процеси | Автоматичне збереження прогресу читання, цитат та налаштувань між сесіями | Якісний |
| Інновації та навчання | Відпрацювання кросплатформної розробки на Flutter, MVVM-архітектури, роботи з SQLite та TTS API | Якісний |
| Фінанси | Нульові операційні витрати після розробки; відсутність залежності від платних сервісів | Кількісний |

Показник рентабельності інвестицій ROI для некомерційного навчального проєкту розраховується через співвідношення отриманої практичної цінності до витрат на розробку. Якщо прийняти умовну ринкову вартість аналогічного застосунку на рівні мінімальної комерційної ціни розробки подібного продукту – 15000 грн., то показник ROI складе:

$$ROI = \frac{15000 - 6222}{6222} \times 100\% \approx 141\%. \quad (3.2)$$

Отримане значення ROI перевищує мінімально прийнятний рівень рентабельності, що підтверджує економічну доцільність розробки. Додатковим аргументом на користь ефективності є відсутність будь-яких ліцензійних відрахувань і операційних витрат протягом усього терміну використання застосунку, що вигідно відрізняє його від комерційних аналогів, розглянутих у підрозділі 1.2.

На рисунку 3.6 наведено структуру витрат на розробку застосунку у вигляді діаграми.

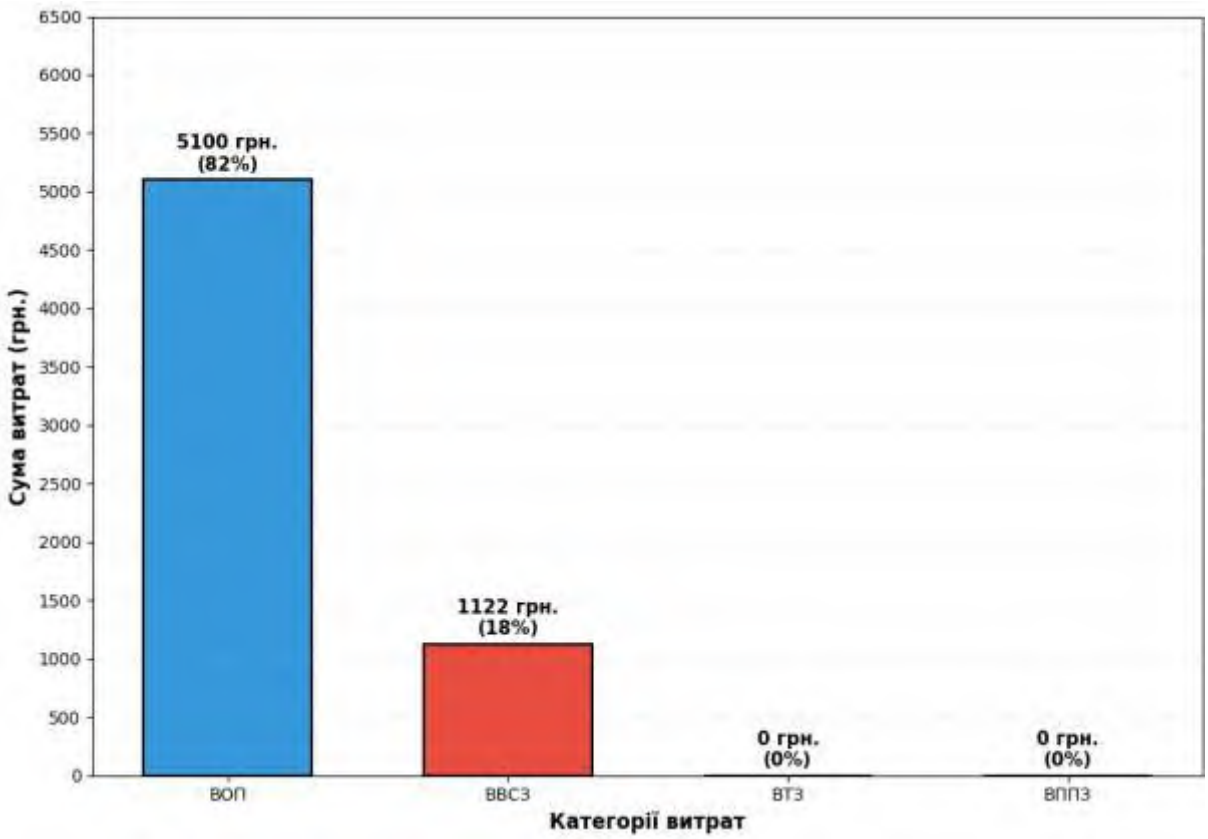


Рисунок 3.6 – Структура витрат на розробку мобільного застосунку

Було описано реалізацію мобільного застосунку для читання та озвучування TXT-файлів на платформі Android з використанням фреймворку Flutter та архітектурного патерну MVVM, детально розглянуто реалізацію функції озвучування на базі вбудованого Android TTS API через пакет flutter_tts, проведено функціональне тестування на реальному пристрої Xiaomi Redmi, яке підтвердило коректну роботу всіх шістнадцяти перевірених функцій, а також виконано техніко-економічне обґрунтування, що засвідчило низьку собівартість розробки (6222 грн), відсутність операційних витрат і розрахунковий показник ROI на рівні 141%, що підтверджує економічну ефективність розробленої системи.

# ВИСНОВКИ

У результаті виконання кваліфікаційної роботи була досягнута її початкова мета та вирішені поставлені завдання. Було проаналізовано наукову, технічну та методичну літературу з теми розробки мобільних застосунків для читання та озвучування текстового контенту на платформі Android.

Розглянуто сучасні застосунки-аналоги у категорії читання та озвучування текстів, зокрема «Букмейт», «Літрес», Google Play Books та Moon+ Reader. Проаналізовано їх функціональні можливості, переваги та недоліки, що дозволило виявити незайняту нішу: відсутність безкоштовного, повністю автономного україномовного застосунку для читання та озвучування довільних TXT-файлів без залежності від хмарних сервісів. Визначено функціональні вимоги до розроблюваної системи із застосуванням методу пріоритизації MoSCoW, а також нефункціональні вимоги щодо продуктивності, надійності, зручності використання та сумісності. Обґрунтовано вибір технологічного стеку на основі фреймворку Flutter, мови програмування Dart, локальної бази даних SQLite та вбудованого Android TTS API як оптимального для реалізації поставлених задач в умовах обмежених апаратних ресурсів середовища розробки.

Деталізовано архітектуру мобільного застосунку «Audible TXT». Описано архітектурний патерн MVVM, що забезпечує чітке розділення логіки представлення та бізнес-логіки. Спроєктовано реляційну базу даних із чотирма таблицями – books, read_progress, notes та tts_settings – із каскадним видаленням пов'язаних записів. Розроблено макети користувацького інтерфейсу відповідно до принципів Material Design 3 із підтримкою світлої, темної та сепіа-теми. Обґрунтовано вибір вбудованого Android TTS API як методу озвучування, що забезпечує підтримку української мови, автономну роботу та повну конфіденційність без передачі тексту на зовнішні сервери.

Описано розробку та тестування мобільного застосунку. Реалізовано повний стек компонентів відповідно до спроєктованої архітектури: моделі даних,

репозиторії, сервіси файлової роботи та синтезу мовлення, ViewModel-класи та чотири екрани застосунку. Алгоритм розбиття TXT-файлу на сторінки забезпечує природні межі між сторінками без розриву абзаців і коректну передачу цілих речень до рушія озвучування. Проведено функціональне тестування на реальному пристрої Xiaomi Redmi під управлінням Android 13, яке охопило шістнадцять тестових сценаріїв і підтвердило коректну роботу всіх реалізованих функцій, включаючи завантаження файлів, збереження прогресу читання, управління вибраним, озвучування тексту та налаштування параметрів синтезу мовлення.

Техніко-економічне обґрунтування показало, що загальні витрати на розробку застосунку склали 6222 грн, при цьому операційні витрати після завершення розробки дорівнюють нулю завдяки використанню виключно безкоштовних інструментів і відсутності серверної інфраструктури. Розрахунковий показник ROI склав близько 141%, що підтверджує економічну доцільність розробки.

Таким чином, поставлені завдання вирішені у повному обсязі. Напрямками подальших досліджень є розширення підтримуваних форматів файлів – зокрема EPUB та PDF – для охоплення ширшої аудиторії читачів. Також перспективним є впровадження функцій збереження цитат через виділення тексту, повноцінного пошуку по тексту з підсвічуванням збігів та навігації свайпом між сторінками, реалізація яких запланована у наступних версіях застосунку. Окремим напрямком розвитку є оптимізація компонування екрана читання для усунення виявленого переповнення нижньої панелі на пристроях із невеликою висотою екрана, а також дослідження можливості інтеграції локальних нейромережевих TTS-моделей для підвищення якості синтезованого мовлення без підключення до інтернету.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. App Downloads Data (2026). URL: https://www.businessofapps.com/data/app-statistics/ (дата звернення: 03.05.2026).

2. Audiobooks Market Trends, Share and Forecast, 2026-2033. URL: https://www.coherentmarketinsights.com/industry-reports/audiobooks-market (дата звернення: 03.05.2026).

3. Читанка - читай електронні книжки. URL: https://chytanka.com/ (дата звернення: 03.05.2026).

4. Android Text-to-Speech API. Android Developers Documentation. 2024. URL: https://developer.android.com/reference/android/speech/tts/TextToSpeech (дата звернення: 03.05.2026).

5. Flutter – Build apps for any screen. Google LLC. 2024. URL: https://flutter.dev (дата звернення: 03.05.2026).

6. Morris A., Maier V., Green P. From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition. *Proceedings of Interspeech.* 2004. URL: https://www.isca-archive.org/interspeech_2004/morris04_interspeech.pdf (дата звернення: 03.05.2026).

7. Tan X., Qin T., Soong F., Liu T.-Y. A Survey on Neural Speech Synthesis. *arXiv preprint.* 2021. URL: https://arxiv.org/abs/2106.15561 (дата звернення: 03.05.2026).

8. Share of eBook users in 53 countries & territories worldwide, as of January 2025. URL: https://www.statista.com/forecasts/1452583/share-of-ebook-users-in-selected-countries-worldwide/ (дата звернення: 03.05.2026).

9. 8 найкращих сервісів для читання та покупки книг. URL: https://nmus.van.cx.ua/articles/najkrashhij-servis-elektronni-knigi.html (дата звернення: 03.05.2026).

10. Google. Upload PDF and EPUB files to your library. URL: https://support.google.com/googleplay/answer/11012086 (дата звернення: 03.05.2026).

11. Українська Електронна Бібліотека Лібрук (Libruk). URL: https://libruk.com.ua (дата звернення: 03.05.2026).

12. Moon+ Reader – офіційна сторінка застосунку. URL: https://play.google.com/store/apps/details?id=com.flyersoft.moonreader (дата звернення: 03.05.2026).

13. Flutter documentation. Get started: Install. Google LLC. 2024. URL: https://docs.flutter.dev/install (дата звернення: 03.05.2026).

14. Develop Android apps with Kotlin. URL: https://developer.android.com/kotlin (дата звернення: 03.05.2026).

15. A Comparison of Native and Cross-Platform Frameworks for Mobile Applications / P. Nawrocki et al. *Computer*. 2021. Vol. 54, no. 3. P. 18–27. URL: https://doi.org/10.1109/mc.2020.2983893 (дата звернення: 03.05.2026).

16. sqflite 2.4.2+1. URL: https://pub.dev/packages/sqflite (дата звернення: 03.05.2026).

17. flutter_tts 4.2.5. URL: https://pub.dev/packages/flutter_tts (дата звернення: 03.05.2026).

18. Wiegers K., Beatty J. Software Requirements. 3rd ed. Microsoft Press, 2013. 672 p. URL: https://www.microsoftpressstore.com/store/software-requirements-9780735679665 (дата звернення: 03.05.2026).

19. Distribution dashboard. URL: https://developer.android.com/about/dashboards (дата звернення: 03.05.2026).

20. Nielsen J. Response Times: The 3 Important Limits. Nielsen Norman Group. 2024. URL: https://www.nngroup.com/articles/response-times-3-important-limits/ (дата звернення: 03.05.2026).

21. Rayner K., Schotter E. R., Masson M. E. J. et al. So Much to Read, So Little Time: How Do We Read, and Can Speed Reading Help? Psychological Science in the

Public Interest. 2016. Vol. 17, no. 1. P. 4–34. URL: https://doi.org/10.1177/1529100615623267 (дата звернення: 03.05.2026).

22. Sommerville I. Software Engineering. 10th ed. Pearson, 2015. 816 p. URL: https://www.pearson.com/en-us/subject-catalog/p/software-engineering/P200000003258 (дата звернення: 03.05.2026).

23. IEEE Std 830-1998. IEEE Recommended Practice for Software Requirements Specifications. IEEE, 1998. URL: https://standards.ieee.org/ieee/830/1222/ (дата звернення: 03.05.2026).

24. ISO/IEC 25010:2023. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE). ISO, 2023. URL: https://www.iso.org/standard/78176.html (дата звернення: 03.05.2026).

25. Material Design 3. Google LLC. 2024. URL: https://m3.material.io/ (дата звернення: 03.05.2026).

26. Flutter performance best practices. Flutter documentation. Google LLC. 2024. URL: https://docs.flutter.dev/perf/best-practices (дата звернення: 03.05.2026).

27. Dobrean D., Dioșan L. A Comparative Study of Software Architectures in Mobile Applications. *Studia Universitatis Babeș-Bolyai Informatica*. 2019. Vol. 64, no. 2. P. 49–64. URL: https://doi.org/10.24193/subbi.2019.2.04 (дата звернення: 03.05.2026).

28. sqflite_migration 0.3.0. URL: https://pub.dev/packages/sqflite_migration/versions (дата звернення: 03.05.2026).

29. shared_preferences 2.5.5. URL: https://pub.dev/packages/shared_preferences (дата звернення: 03.05.2026).

30. User Interface (UI) Design Patterns. URL: https://ixdf.org/literature/topics/ui-design-patterns (дата звернення: 03.05.2026).

31. Components – Material Design 3. URL: https://m3.material.io/components (дата звернення: 03.05.2026).

32. Web Content Accessibility Guidelines (WCAG) 2.1. W3C Recommendation. 2023. URL: https://www.w3.org/TR/WCAG21/ (дата звернення: 03.05.2026).

33. Karpe R. P. A Survey :On Text To Speech Synthesis. *International Journal for Research in Applied Science and Engineering Technology*. 2018. Vol. 6, no. 3. P. 351–355. URL: https://doi.org/10.22214/ijraset.2018.3054 (дата звернення: 03.05.2026).

34. Shen J., Pang R., Weiss R. J. et al. Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions. *ICASSP* 2018. URL: https://doi.org/10.1109/ICASSP.2018.8461368 (дата звернення: 03.05.2026).

35. Ren Y., Ruan Y., Tan X. et al. FastSpeech: Fast, Robust and Controllable Text to Speech. *NeurIPS* 2019. URL: https://arxiv.org/abs/1905.09263 (дата звернення: 03.05.2026).

36. Android Developers. TextToSpeech API Reference. URL: https://developer.android.com/reference/android/speech/tts/TextToSpeech (дата звернення: 03.05.2026).

37. Speech Recognition & Synthesis. URL: https://play.google.com/store/apps/details?id=com.google.android.tts (дата звернення: 03.05.2026).

38. Flutter App, Speech to Text and Text to Speech. URL: https://dev.to/saad4software/flutter-app-speech-to-text-and-text-to-speech-5bi7 (дата звернення: 03.05.2026).

39. Dowding S., Gutwin C., Cockburn A. User speech rates and preferences for system speech rates. *International Journal of Human-Computer Studies*. 2024. P. 103222. URL: https://doi.org/10.1016/j.ijhcs.2024.103222 (дата звернення: 03.05.2026).